UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

MESTRADO ACADÊMICO EM COMPUTAÇÃO

ANTONIO SANTOS DE SOUSA

CONTROLE DE MOVIMENTO DE PERSONAGENS FISICAMENTE SIMULADOS USANDO FUNÇÕES DE RECOMPENSA

QUIXADÁ

2022

ANTONIO SANTOS DE SOUSA

CONTROLE DE MOVIMENTO DE PERSONAGENS FISICAMENTE SIMULADOS USANDO FUNÇÕES DE RECOMPENSA

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em computação. Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Rubens Fernandes Nunes

QUIXADÁ

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S696c Sousa, Antonio Santos de.
Controle de movimento de personagens fisicamente simulados usando funções de recompensa / Antonio Santos de Sousa. – 2022.
76 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Campus de Quixadá, Programa de Pós-Graduação em Computação, Quixadá, 2022.
Orientação: Prof. Dr. Rubens Fernandes Nunes.

1. Aprendizagem profunda. 2. Movimento. 3. Simulação. 4. Rede Neural Artificial. I. Título.
CDD 005

ANTONIO SANTOS DE SOUSA

CONTROLE DE MOVIMENTO DE PERSONAGENS FISICAMENTE SIMULADOS USANDO FUNÇÕES DE RECOMPENSA

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em computação. Área de Concentração: Ciência da Computação

Aprovada em: __/__/___.

BANCA EXAMINADORA

Prof. Dr. Rubens Fernandes Nunes (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Regis Pires Magalhães
Universidade Federal do Ceará (UFC)

Prof. Dr. Joaquim Bento Cavalcante Neto
Universidade Federal do Ceará (UFC)

Prof. Dr. Creto Augusto Vidal
Universidade Federal do Ceará (UFC)

Dedico este trabalho a José Izaquiel e Maria do Remédios, os melhores pais que alguém poderia ter. Vocês me ensinaram o que significa o amor, com vocês eu sempre me senti amado e seguro.

AGRADECIMENTOS

Agradeço em primeiro lugar aos meus pais por todas as lições que vocês me deram e dão todos os dias sobre amor, amizade, respeito, caridade, doação, altruísmo, compreensão e perdão. Estou orgulhoso e honrado por ter pais tão especiais. E meus irmãos que sempre me serviram de incentivo e inspiração para continuar e não desistir.

Ao meu orientador, Dr. Rubens Nunes, por sua orientação atenciosa, incentivos sinceros, paciência, dedicação e compartilhar comigo seu conhecimento, me guiando na minha jornada e acreditando no meu trabalho, mesmo quando eu duvidava de mim. Sendo um pilar de apoio para mim, direcionando com maestria as nossas ideias para que consigamos finalizar o nosso trabalho. Tenho certeza que não teria chegado tão longe sem o seu apoio. Você não foi apenas um mentor para mim, você sempre será um mestre e amigo.

Aos membros da Banca Dr. Creto Vidal e Dr. Joaquim Bento pela disponibilidade e pelas orientações e contribuições valiosas durante a escrita do artigo *Physics-Based Motion Control Through DRL's Reward Functions*. Ao membro Dr. Régis Pires, que me acolheu na cidade e me apresentou os primeiros passos no aprendizado de máquina. A todos vocês avaliadores que gentilmente compartilharam seus tempos e conhecimentos revisando minha dissertação e pelas indispensáveis contribuições que aprimoraram e engrandeceram esta pesquisa. Vocês foram mais que avaliadores são coorientadores desse trabalho.

Ao Danilo Borges que compartilhou comigo seus conhecimentos, materiais, ajuda nos experimentos e revisão do texto. À Tatiane Costa que esteve nessa jornada comigo não deixando eu desistir sempre puxando minha orelha quando eu precisava. À Renan Vieira que mesmo na distância foi um porto seguro para mim, compartilhando as dores de também fazer uma pós-graduação e com palavras sábias, alegres e gentis muitas gentes diminuindo a dor que eu sentia.

Aos professores e ao programa de Mestrado em Ciência da Computação da Universidade Federal do Ceará - Campus Quixadá que possibilitaram ainda mais conhecimentos para minha formação. Aos funcionários desta instituição, que sempre se mostraram disponíveis para ajudar no que fosse preciso.

Ao Instituto Federal de Educação, Ciência e Tecnologia do Piauí pelo apoio institucional, imprescindível para a realização deste trabalho, e, especialmente, ao diretor do Campus Parnaíba, Luis Fernando e meus colegas professores de informática.

Aos amigos do programa de Pós-Graduação em Computação da UFC-Campus

Quixadá, cujo o apoio, ensinamentos e colaboração foram fundamentais para esta conquista, em especial ao José Neto, Warlles, Caetano, Wellington, Rahel, Rogério, Gilberto e Júnior que tornaram os momentos de solidão mais leves e descontraídos.

A todos os professores que tive a oportunidade de ter, em especial aos meus professores de Computação da UESPI - Parnaíba, onde passei a admirar essa área e a profissão, servindo de inspiração para me tornar o profissional que sou.

Aos meus amigos, a família "*Khronus/Banzai*", pelos momentos descontraídos, pelas palavras de apoio e por enriquecerem minha vida com significado. Aos meus filhos de coração Pedro Felipe, Letícia Gabrielly e Augusto Sousa que me servem de inspiração para ser sempre uma pessoa melhor. Aos meus alunos, que são o meu *ikigai*, minha força motivadora que me dá um senso de propósito e razão para viver.

À cidade de Quixadá e às pessoas maravilhosas e acolhedoras que encontrei, em especial a Sra. Lúcia por todas as nossas conversas e pelo carinho. Aos meus ex-alunos João Henrique e Bruno Carvalho que me deram abrigo e apoio nos primeiros momentos na cidade.

E por fim, agradeço a todos os autores citados neste trabalho e que certamente contribuíram direta ou indiretamente para a existência dessa dissertação. Em especial, Arthur Juliani, que dedicou um tempo para me orientar nos experimentos. Parafraseando Isaac Newton, se cheguei tão longe é porque tive a ajuda de gigantes.

Para estas e todas as outras pessoas que contribuíram na minha vida, um muito obrigado.

“Existem três coisas que não podem ser interrompidas: o sonho dos homens, o fluxo do tempo e a vontade herdada, enquanto as pessoas continuarem buscando o sentido da liberdade tudo isso jamais deixará de existir. O sonho de um homem nunca tem fim.”

(Eichiro Oda)

RESUMO

Gerar movimentos naturais para personagens articulados fisicamente simulados é desafiador principalmente porque ajustar a qualidade visual do movimento muitas vezes compromete o funcionamento básico do controlador, pois os parâmetros de controle geralmente possuem um relacionamento não intuitivo com a animação resultante. O Aprendizado por reforço profundo (*Deep Reinforcement Learning* (DRL)) tem sido uma abordagem bastante explorada recentemente para tratar o problema de controle de tais estruturas, em que uma rede neural é usada para lidar com esse relacionamento entre as entradas de controle e as informações de saída a serem usadas pelos atuadores. Além disso, a definição de uma recompensa apropriada é essencial para guiar o progresso da aprendizagem. Para problemas de controle contínuo, o processo de aprendizado pode levar de horas a dias de treinamento, dependendo da tarefa, mas após essa etapa, a rede treinada funciona em tempo real. Este trabalho propõe, portanto, ajustes na função de recompensa e nas informações de entrada e de saída da rede para fornecer ao animador um maior grau de controle sobre os movimentos resultantes. Esse controle é explorado tanto na fase de treinamento, quanto durante a execução da simulação em tempo real. Os termos de recompensa propostos foram adaptados para dois personagens com morfologias distintas e se mostraram capazes de nitidamente diferenciar diversos modos de locomoção, como corridas e saltos. Os ajustes nas informações de entrada e de saída da rede permitiram o controle de velocidade em tempo real e a possível imposição de simetria ao movimento do personagem. Experimentos simulando interações com o ambiente, tais como objetos sendo lançados contra o personagem e modificações causando irregularidades no terreno, mostraram a robustez do controle obtido usando DRL.

Palavras-chave: Animação de personagens fisicamente simulados; Controle de movimento; Aprendizado por reforço profundo.

ABSTRACT

Generating natural movements for physics-based articulated characters is challenging mainly because adjusting the visual quality of the movement often compromises the basic functioning of the controller, as the control parameters generally have a non-intuitive relationship with the resulting animation. Deep Reinforcement Learning (DRL) has been a recently explored approach to treat the control problem in such structures, in which a neural network is used to deal with this relationship between control inputs and information outputs to be used by the actuators. Further, the definition of an appropriate reward is necessary to guide the learning process. Although the learning process takes a long time, a major advantage of the DRL method is that the trained network works in real-time. This work, therefore, proposes adjustments in the reward function and in the network input and output information to provide the animator with a greater degree of control over the resulting movements. That control is explored both in the training phase and during the simulation in real-time. The proposed reward terms were adapted for two characters with different morphologies and proved to be able to clearly differentiate interesting types of locomotion, such as running and jumping. Adjustments to network input and output information allowed real-time speed control and the possible imposition of symmetry on the character's movement. Experiments simulating interactions with the environment, such as objects being thrown at the character and modifications causing irregularities in the terrain, showed the robustness of the control obtained using DRL.

Keywords: Animation of Physics-based Characters; Motion Control; Deep Reinforcement Learning.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

DRL	*Deep Reinforcement Learning*
SVR	*Symposium on Virtual and Augmented Reality*
DOF	*Degrees of Freedom*
PD	Proporcional-Derivativo
MDP	*Markov Decision Process*
SAC	*Soft Actor-Critic*
PPO	*Proximal Policy Optimization*
CL	*Curriculum Learning*
PCA	*Principal Component Analysis*
NPC	*Non-Player Character*

SUMÁRIO

1 INTRODUÇÃO

A animação de personagens articulados fisicamente simulados é um desafio há muito tempo na área de animação. Normalmente personagens virtuais são animados utilizando métodos cinemáticos, em que as poses dos movimentos são geradas diretamente pela ação do animador ou obtidas a partir de dados de captura de movimentos. A dificuldade em adaptar esses movimentos a interações com o ambiente limita o uso desses métodos em algumas aplicações em tempo real. No caso da captura de movimentos, movimentos correspondentes a cada tipo de interação precisam ser capturados antecipadamente. Além disso, animar manualmente aspectos físicos, tais como a influência do chão e reações a possíveis impactos no corpo do personagem, mesmo para animadores dotados de dons artísticos, é muito trabalhoso e pode se tornar inviável dependendo do contexto.

O uso de uma simulação física de acordo com as equações de movimento do personagem pode melhorar o realismo da animação, pois permite que as interações físicas com o ambiente sejam geradas automaticamente. Esse tipo de animação é empregado principalmente em simulações passivas, onde os corpos sofrem os efeitos de forças externas, como corpos em queda, fluídos, cabelos e roupas. Ao animar personagens articulados usando uma simulação física, eles precisam ser capazes de gerar os torques necessários para sua movimentação, além de sofrer influência de forças externas como atrito, resistência do ar, gravidade e interações com outros corpos. Assim, durante o seu movimento no ambiente, as interações acontecem de forma automática pois as leis do movimento estão sendo seguidas por todos os corpos na simulação física.

Em jogos, essa automaticidade é essencial. Entretanto, ela vem com o preço de controlar os atuadores do personagem, que precisam aplicar os torques internos adequados para executar os movimentos desejados. Além disso, o personagem não possui atuadores para todos os seus graus de liberdade, ou seja, sua disposição no ambiente depende das suas interações com o mesmo, além dos torques que ele é capaz de gerar. Um torque gerado em uma junta também influencia as outras juntas do personagem e um controlador se faz necessário para gerenciar todas essas variáveis e gerar o movimento desejado.

Gerar movimentos naturais para tais personagens é desafiador, principalmente, porque ajustar a qualidade visual do movimento muitas vezes compromete o funcionamento básico do controlador, pois os parâmetros de controle geralmente possuem um relacionamento não linear e não intuitivo com a animação resultante. O aprendizado por reforço profundo (*Deep*

Reinforcement Learning - DRL) tem sido uma abordagem bastante explorada recentemente para tratar o problema de controle de estruturas articuladas fisicamente simuladas, principalmente para gerar movimentos de locomoção (BERGAMIN *et al.*, 2019; BOOTH; IVANOV, 2020; PENG *et al.*, 2017; CHENTANEZ *et al.*, 2018). Uma rede neural é usada para lidar com esse relacionamento entre as entradas de controle, que consistem basicamente de informações sensoriais do personagem e do ambiente, e as informações de saída, ângulos desejados ou torques, a serem usadas pelos atuadores.

Embora o processo de aprendizagem, correspondente a definir os pesos das conexões da rede, demore algumas horas ou dias de treinamento, uma grande vantagem do método de DRL é que, terminado esse processo, a rede treinada funciona em tempo real. A definição de uma recompensa é necessária para guiar o progresso, mas mesmo o simples deslocamento horizontal usado como premiação tem o potencial de resultar em diferentes e complexos tipos de locomoção, inclusive capazes de se adaptar ao ambiente de treinamento (HEESS *et al.*, 2017), dependendo das estratégias combinadas com o método de DRL.

De fato, DRL tem se mostrado robusto, sendo capaz de controlar o equilíbrio de personagens diversificados de maneira bastante estável, por exemplo demonstrando habilidades de corrida bípede em direção a locais definidos em tempo real dentro de ambientes tridimensionais, sem cair (JULIANI *et al.*, 2020). Entretanto, o uso isolado de DRL em personagens, tais como humanoides, comumente resulta em movimentos nada parecidos com os de um humano real, apresentando, por exemplo, movimentos convulsivos de alta frequência nos braços, nem sempre numa tentativa de ajudar no equilíbrio. O uso de modelos de movimentos capturados (PENG *et al.*, 2018) ou a inclusão de termos e ajustes bastante cuidadosos na função de recompensa podem melhorar os resultados (YU *et al.*, 2018).

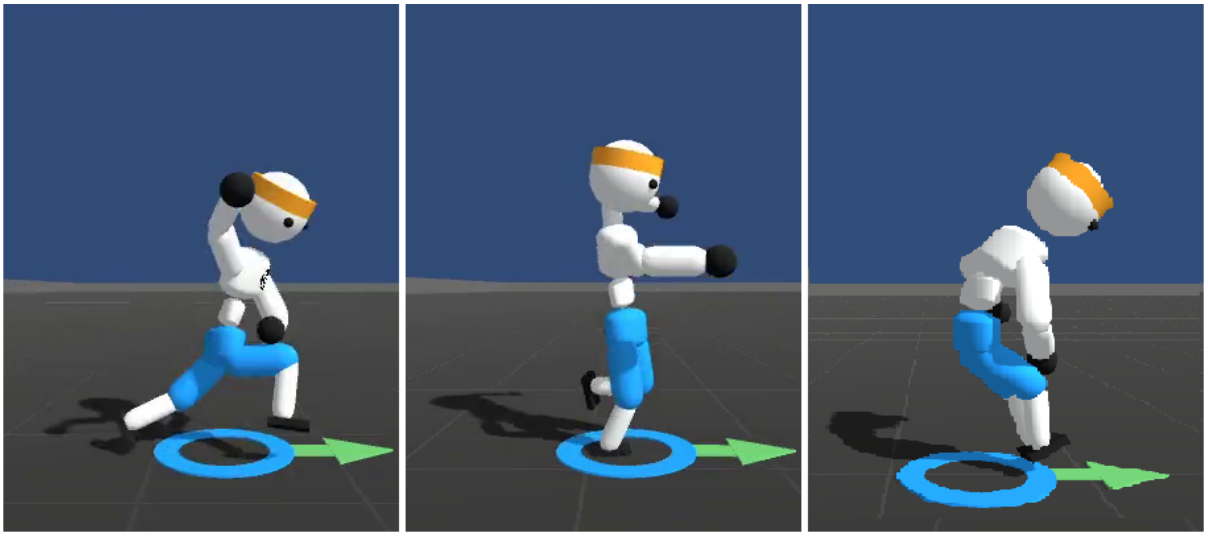
Diferente de tarefas com domínio discreto de ações e recompensas bem definidas, tais como as pontuações em jogos do tipo *arcade* ou em alguns jogos de tabuleiro como o xadrez, o desafio na definição da recompensa de tarefas de controle contínuo, tais como animação de personagens, geralmente consiste em identificar características específicas dos possíveis comportamentos da estrutura, de acordo com a animação desejada. Essas características ainda precisam ser traduzidas no formato de uma expressão matemática com o objetivo de guiar a aprendizagem.

Por mais que alguns artigos busquem manter as recompensas o mais simples possível e foquem em incorporar estratégias que facilitem a aprendizagem por outros meios (HEESS *et*

al., 2017; YU *et al.*, 2018), a escolha de como definir a função de recompensa nesse contexto de animação de personagens continua um problema em aberto quando se deseja um maior controle sobre o resultado, e a maioria dos trabalhos acaba adaptando os critérios usados para melhor se encaixar no seu respectivo cenário. O desafio se torna ainda maior devido ao grau de sensibilidade causado por ajustes na recompensa usada.

Uma das contribuições deste trabalho é mostrar o quanto de controle que a definição da função de recompensa pode proporcionar ao animador, sobre o movimento resultante, no sentido de diferenciar, por exemplo, uma corrida bípede normal, uma corrida saltitando em um pé só ou uma corrida usando os dois pés juntos para saltar como um canguru (Figura 1).

Figura 1 – Diferentes tipos de locomoção controlados através da função de recompensa. À esquerda, uma corrida bípede. No centro, uma corrida saltitando em um pé só. À direita, uma corrida saltando usando os dois pés juntos.



Fonte: elaborado pelo autor (2022).

Lembrando que uma recomendação importante ao se usar DRL é manter os critérios de progresso fáceis de generalizar. Embora um trabalhoso processo de tentativa e erro seja geralmente necessário para escolher os termos de recompensa adequados, neste trabalho, os termos usados nos experimentos foram propostos mantendo em mente a possibilidade de serem fáceis de adaptar para diferentes personagens e se mostraram capazes de nitidamente diferenciar diversos tipos de locomoção. Adaptados para dois personagens com morfologias distintas, os termos propostos também consistem em uma das formas de contribuição deste estudo.

Ainda na direção de fornecer ao animador um maior controle sobre aspectos relacionados ao tipo de locomoção, este trabalho propõe que algumas informações de controle específicas possam fazer parte da entrada da rede neural, como velocidade, direção ou altura,

permitindo que esses valores sejam modificados pelo usuário em tempo real, no decorrer da simulação da rede já treinada. Durante o treinamento, no início de cada episódio, essas informações de controle precisam ser sorteadas para permitir que o personagem se adapte aos diferentes valores a serem usados, após a rede treinada. Embora a ideia possa ser generalizada para diferentes informações, os experimentos mostrados neste trabalho exploram apenas a velocidade desejada.

Enquanto ajustes nas entradas da rede permitem uma forma de controle útil em tempo real, ajustes nas saídas da rede também podem ser úteis, por exemplo para adicionar restrições ao espaço de busca das ações, a fim de facilitar o trabalho do DRL. Nesse sentido, este trabalho explora o controle de simetria do movimento resultante através de restrições responsáveis por espelhar os movimentos dos membros, obtendo resultados com aparência mais natural.

Um vídeo com os resultados de todos os experimentos realizados neste trabalho pode ser encontrado no link disponível no final da Subseção 5.1.

1.1 Questões de Pesquisa

A avaliação da nossa proposta será orientada pelas seguintes questões de pesquisa:

QP1 Qual a dificuldade em se definir a função de recompensa na geração de movimentos de personagens fisicamente simulados usando DRL?

QP2 Qual o nível de controle que o animador pode esperar sobre o movimento resultante através da manipulação da função de recompensa?

QP3 Quais são os limites do aprendizado por reforço na produção de movimento para personagens fisicamente simulados, manipulando apenas a função de recompensa?

QP4 Como o controle do movimento pode ser explorado em tempo real, durante a execução da rede já treinada?

QP5 Como o controle da simetria pode contribuir na obtenção de melhores movimentos?

1.2 Contribuições

Este trabalho aborda o problema de controle de personagens articulados fisicamente simulados utilizando DRL e apresenta estas 5 contribuições principais:

1. Mostrar que nível de controle sobre o movimento resultante o animador pode esperar ao manipular a função de recompensa;
2. Propor uma estrutura de controle com DRL associada a uma função de recompensa que

possa ser adaptada ao movimento desejado e à morfologia do personagem controlado;
3. Incluir a velocidade no processo de treinamento para que, após treinada a rede neural, o personagem consiga adaptar seu movimento a diferentes velocidades em tempo real;
4. Permitir o controle da simetria do movimento gerado, através de ajustes na saída da rede neural responsáveis por restringir as ações dos membros do personagem; e
5. Possibilitar ao animador observar o surgimento espontâneo de movimentos, gerados por DRL, sem depender do uso de movimentos de referência.

1.3 Publicação de artigo

Durante o desenvolvimento deste trabalho ocorreu a publicação do artigo *Physics-Based Motion Control Through DRL's Reward Functions*, no *Symposium on Virtual and Augmented Reality (SVR)'21*. O referido artigo apresenta, de forma mais resumida, as principais contribuições deste trabalho. Nesta dissertação, as ideias apresentadas no artigo são exploradas com mais detalhes e experimentos relacionados ao controle de simetria também são acrescentados.

1.4 Estrutura da dissertação

O restante deste trabalho está dividido nos seguintes capítulos. No Capítulo 2 é apresentado o referencial teórico, englobando alguns conceitos sobre animação de personagens fisicamente simulados e aprendizado de máquina. No Capítulo 3, discutem-se os trabalhos relacionados mais relevantes na área de animação e aprendizado por reforço profundo que contribuíram na construção desta pesquisa, juntamente com o posicionamento deste trabalho em relação a eles. O Capítulo 4 aborda a estrutura de controle utilizada neste trabalho, como também o método proposto para a obtenção das animações. No Capítulo 5 é mostrada uma série de experimentos para demonstrar as capacidades de controle da abordagem utilizada. Por fim, no Capítulo 6, são apresentadas as principais conclusões alcançadas nesta pesquisa e são sugeridos alguns trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os conceitos e as técnicas utilizados no trabalho. Na Seção 2.1, são descritos os processos que envolvem a animação de personagens fisicamente simulados e o uso de controladores para solucionar o problema de controle de personagens articulados. Na Seção 2.2, são apresentados os conceitos de aprendizado de máquina, focando em aprendizado por reforço, abordando suas características e técnicas aplicadas na resolução de problemas desse tipo. Na Seção 2.3, são apresentados uma breve discussão do ambiente utilizado *Unity* e *ML-Agents*.

2.1 Animação de Personagens Articulados Fisicamente Simulados

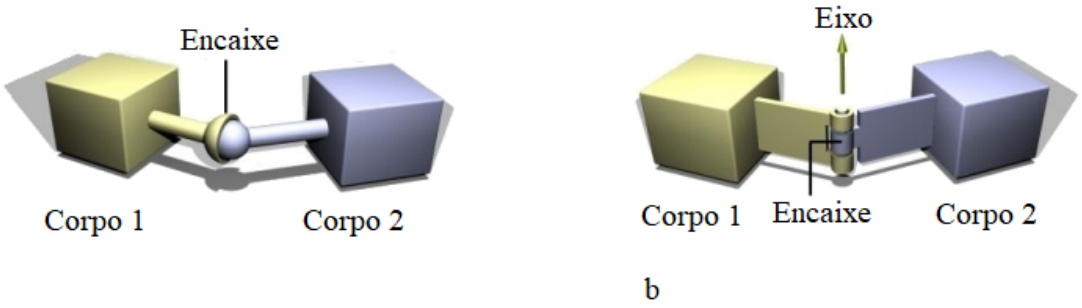
Nesta seção, são discutidos os conceitos sobre personagens articulados, representados como estruturas articuladas de corpos rígidos. Em seguida, são tratados alguns métodos disponíveis para animação de personagens articulados. Dentro do contexto dos métodos dinâmicos, é apresentada a diferença entre sistemas passivos e sistemas atuados. Finalmente, explica-se a motivação do uso de controladores para o problema de controle de personagens fisicamente simulados.

2.1.1 Personagens Articulados

Um personagem articulado é formado por um conjunto de segmentos de corpos rígidos conectados por juntas. Esses personagens são geralmente simplificações dos modelos que estão sendo simulados para aumentar o desempenho da simulação e torná-los mais fáceis de controlar, sendo que as partes dos corpos são modeladas usando formas geométricas simples como cubos, esferas e cilindros (GEIJTENBEEK, 2013).

As juntas servem para simular as restrições de movimento naturais dos corpos que estão conectados. Os movimentos permitidos por uma junta definem seus graus de liberdade (*Degrees of Freedom* (DOF) - *Degrees of Freedom*) (SILVA, 2014). Para modelar a articulação do joelho ou cotovelo, pode-se usar uma junta dobradiça, com um grau de liberdade, enquanto as articulações do quadril ou do ombro exigem o uso de uma junta esférica, com três graus de liberdade. Na Figura 2, são mostrados esses dois tipos de juntas. Normalmente, são especificados limites angulares inferiores e superiores para cada DOF a fim de imitar os limites naturais da articulação.

Figura 2 – (a) Junta esférica (*ball and socket*). (b) Junta dobradiça (*hinge*).



a b

Fonte: Nunes *et al.* (2012).

2.1.2 Métodos de Animação de Personagens Articulados

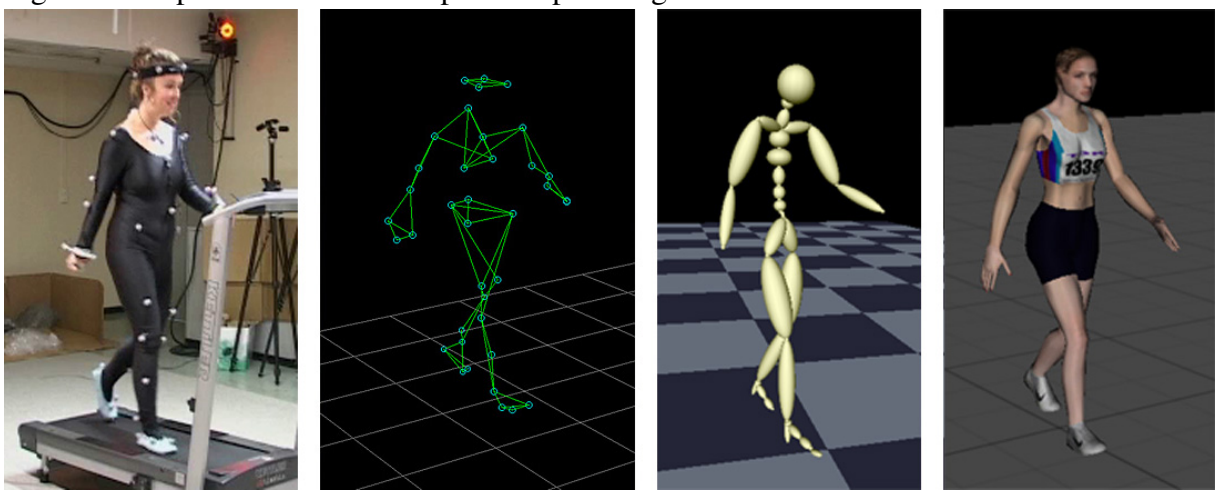
Segundo Nunes (2012), um dos critérios para classificar os métodos de animação de personagens é analisar se o método leva ou não em consideração a influência das forças atuantes e das massas dos corpos para gerar os movimentos. Os métodos cinemáticos não trabalham com as causas do movimento, sendo que as poses são definidas pela manipulação direta dos ângulos e velocidades das juntas, independentemente da física, sem a aplicação de forças e torques. Já os métodos dinâmicos levam em conta os aspectos físicos dos personagens e do ambiente simulado, sendo que o movimento é gerado pela aplicação de forças e torques diretamente nos personagens.

2.1.2.1 Métodos Cinemáticos

Os métodos cinemáticos são os mais comumente usados na animação de personagens. A fidelidade e a qualidade dos movimentos desses mecanismos são determinadas pelas habilidades do animador, que precisa garantir que o movimento seja apropriado. Os estados do personagem são definidos diretamente através dos ângulos e das velocidades angulares das juntas ao longo do tempo, não se preocupando com as forças e os torques que agem sobre ou dentro do personagem, nem com seus efeitos no movimento. Dessa forma, o animador tem controle total sobre a animação gerada.

Uma maneira de obter esses valores de ângulos e velocidades é através da captura de movimentos, sendo um caso especial da abordagem cinemática em que os dados do movimento do personagem animado são obtidos de uma fonte do mundo real, como uma pessoa caminhando (LASZLO, 1997). Para capturar os movimentos, normalmente são usados marcadores anexados em várias partes do corpo e câmeras apropriadas para registrar os movimentos dos marcadores. Na Figura 3, é possível visualizar uma atriz tendo seus movimentos capturados e transferidos para um personagem virtual.

Figura 3 – Captura de movimento para um personagem virtual.



Fonte: Langfield (2013).

Obter os movimentos capturados do mundo real possibilita que a animação seja mais realista e com movimentos mais naturais. Embora a captura de movimentos tenha essa vantagem, para qualquer modificação do movimento originalmente capturado, não há garantias de que ele continue realista. Por exemplo, ao fazer o personagem se mover em cenários diferentes dos usados na captura, a adaptação do movimento ao novo cenário virtual pode comprometer a qualidade dos movimentos resultantes. Também se faz necessário ter hardwares especializados para obter esses dados. Outro ponto é que os movimentos capturados são limitados ao mundo real e a movimentos que possam ser capturados com atores sem colocá-los em risco. Apesar de ter alguns trabalhos que capturam dados de movimentos de animais, como cachorros (PENG *et al.*, 2020; LUO *et al.*, 2020), trabalhar com outros tipos de animais parece ser bem limitante. E se forem usados animais fictícios, só se pode contar com a habilidade do animador para gerar os movimentos.

2.1.2.2 Métodos Dinâmicos

Nos métodos dinâmicos, a animação é gerada a partir da aplicação das leis da física que regem a mecânica clássica onde as forças e os torques são usados para obtenção do movimento desejado. Dessa forma, em vez de as posições e as velocidades dos graus de liberdade serem definidas diretamente, as poses dos personagens são obtidas automaticamente pelas equações de movimento do sistema. A principal vantagem do uso da dinâmica nas animações é que os movimentos gerados são fisicamente realistas. Assim, eles serão compatíveis com as situações a que o personagem é submetido, como caminhar por terrenos irregulares, reagir a colisões e carregar pesos (NOGUEIRA, 2007).

O uso de métodos dinâmicos envolve a utilização de uma simulação física com a integração das equações de movimento sobre o tempo, que exige um esforço computacional maior que o método cinemático. Além disso, a animação resultante depende de uma grande quantidade de variáveis a serem controladas, exigindo do animador um esforço maior já que o controle através de forças e torques não é intuitivo. Devido a isso, a simulação física é utilizada geralmente na simulação de fenômenos passivos, como movimentos de fluídos, roupas e objetos caindo, que são mais fáceis de serem modelados.

Mas, recentemente, os avanços em aprendizado por reforço profundo têm apresentado resultados visualmente interessantes usando métodos dinâmicos para: realização de movimentos acrobáticos (PENG *et al.*, 2018); controle de movimentos em tempo real (BERGAMIN *et al.*, 2019); jogos de basquete (LIU; HODGINS, 2018); movimentos emergentes (HEESS *et al.*, 2017) e diversos outros tipos de movimentos. Diversos trabalhos, como os de Peng *et al.* (2018), Chentanez *et al.* (2018), Aberman *et al.* (2019), combinam métodos cinemáticos e dinâmicos, em que os movimentos são obtidos a partir da captura de movimentos e as restrições físicas são acrescentadas para tornar a animação fisicamente realista.

Os sistemas dinâmicos podem ser divididos em sistemas passivos e sistemas atuados. Sistemas passivos não geram os torques e as forças por eles mesmos, dependendo totalmente de forças e torques externos para o movimento resultante. Um exemplo de movimento passivo seria um objeto caindo de um prédio onde apenas a força gravitacional agiria nele, resultando numa animação convincente apenas com isso. Por outro lado, os sistemas atuados têm a capacidade de gerar forças e torques que agem internamente em suas estruturas, sendo capazes de produzir movimentos por si mesmos. Eles possuem atuadores que são mecanismos geradores de movimentos a partir de estímulos recebidos. Esses atuadores funcionam como se fossem uma simplificação dos músculos dos animais, aplicando torques nas juntas dos personagens (NUNES, 2012).

2.1.3 Controladores

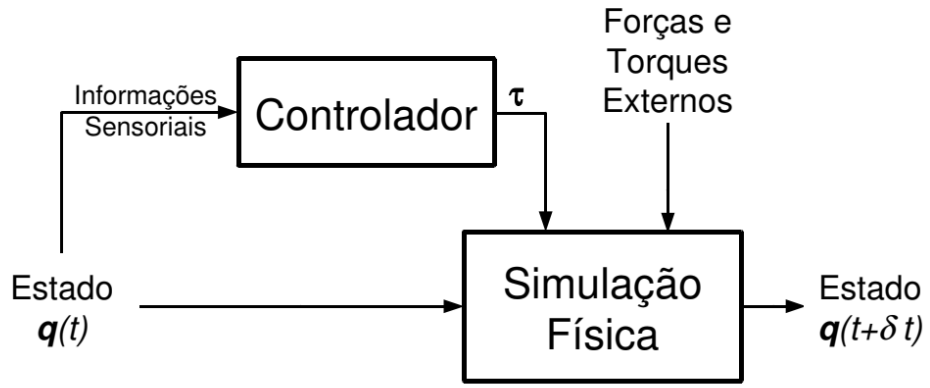
Uma dificuldade que existe na animação usando dinâmica é o problema de controle. Considerando um personagem articulado, um ambiente e um movimento desejado especificado pelo animador, é necessário definir quais seriam as forças e os torques de controle necessários para atingir o movimento desejado fisicamente realista. Por exemplo, um personagem que precise ficar em pé, em equilíbrio, para que isso aconteça, se faz necessária a aplicação contínua nos

atuadores de torques que garantam que o personagem não caia no chão.

O problema do equilíbrio surge pelo fato do personagem ser sub-atuado, ou seja, não ter atuadores para todos os seus graus de liberdade. Assim, os graus de liberdade globais, não atuados, são influenciados indiretamente pelos atuadores internos e pelas forças de reação geradas pela interação com o chão ou com o ambiente como um todo. Outro fator para o problema de controle é que a ação de um atuador em uma junta influencia as outras juntas da estrutura pelo fato de as equações de movimento do sistema serem acopladas.

Por mais que o problema de controle seja um desafio, a vantagem de obter uma interação automática com o ambiente justifica o investimento em melhores métodos. Definir os torques diretamente não é viável para um animador, pelo fato de a escolha dos valores não ser intuitiva. Contudo, há métodos que buscam representar esse controle através de camadas que mapeiam entradas mais fáceis de serem definidas pelo animador nos torques, a serem aplicados nos atuadores. A Figura 4 mostra o funcionamento do controlador. Durante toda a simulação física o controlador vai enviar os torques τ necessários para que o movimento desejado seja realizado (NUNES, 2012).

Figura 4 – Controlador agindo na simulação física.



Fonte: Nunes (2012).

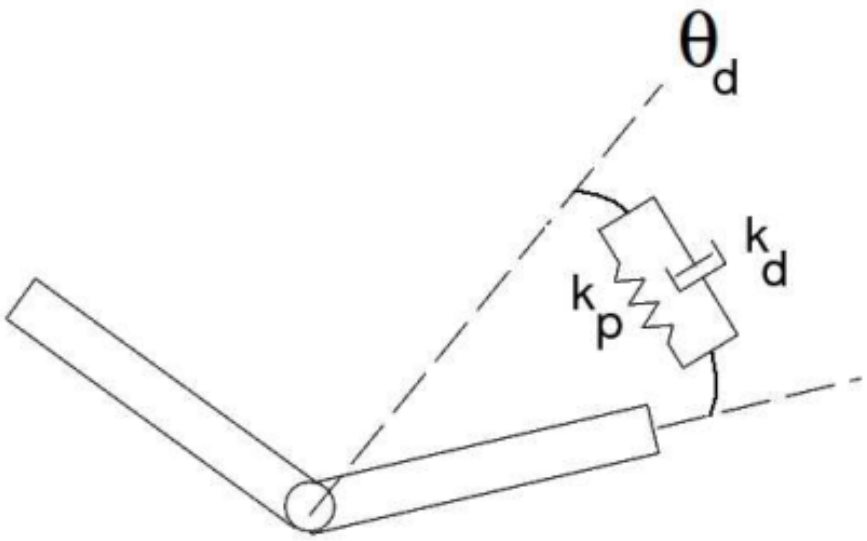
Muitas aplicações utilizam controladores Proporcional-Derivativo (PD) (Proporcional-Derivativo) como uma camada de controle de mais baixo nível de abstração para calcular os torques necessários a serem aplicados nas juntas a partir de ângulos desejados. Um controlador PD funciona simulando molas angulares amortecidas e os torques para essas juntas são calculados com base na seguinte equação:

$$\tau = k_p(\theta_d - \theta) + k_d(\dot{\theta}_d - \dot{\theta}), \tag{2.1}$$

onde τ é o torque gerado na junta, k_p é a constante da mola, k_d é a constante do amortecedor, θ representa a orientação atual, θ_d, a orientação desejada, $\dot{\theta}$, a velocidade angular atual da junta, $\dot{\theta}_d$ velocidade angular desejada da junta.

Na Figura 5, temos um exemplo de uma junta do tipo dobradiça, onde a mola angular funciona fazendo a junta assumir seu ângulo de relaxamento. Esse tipo de controlador tem sido usado há muito tempo na animação de personagens. O trabalho de Panne *et al.* (1994) usa controladores PD para gerar os torques necessários no seu *Pose Control Graphs*, que é uma máquina de estado que tem uma pose particular associada a cada estado do personagem ao longo do tempo. Zordan e Hodgins (2002) usam controladores PD para seguir dados de captura de movimento. Em trabalhos mais recentes, Yang *et al.* (2020) e Peng *et al.* (2020) também usam controladores PD com DRL para tarefas de locomoção.

Figura 5 – Visão de um controlador PD.



Fonte: Nunes (2012).

2.2 Aprendizado de Máquina

O aprendizado de máquina é uma subárea da inteligência artificial em que as aplicações aprendem a realizar tarefas sem usar instruções explícitas. Em vez disso, padrões nos dados são descobertos automaticamente e os resultados desejados são inferidos usando modelos matemáticos e estatísticos. Um problema de aprendizado de máquina geralmente consiste na

aplicação de um algoritmo que constrói um modelo matemático para um conjunto de dados (*dataset*), minimizando uma função de perda (MA, 2019).

2.2.1 Categorias

O aprendizado de máquina é dividido em três categorias principais: aprendizado supervisionado; aprendizado não supervisionado e aprendizado por reforço (RASCHKA; MIRJALILI, 2017). No aprendizado supervisionado, o conjunto de dados é organizado com entradas e suas saídas correspondentes. Os algoritmos tentam construir um modelo matemático que relacione as entradas às saídas correspondentes do conjunto de dados. Os problemas de aprendizado supervisionado podem ser de dois tipos: classificação, quando o objetivo é encontrar a classe para uma nova entrada, e regressão, em que o objetivo é relacionar a entrada a uma saída de valor contínuo.

No aprendizado não supervisionado, o conjunto de dados possui as entradas, mas não possui as saídas correspondentes, usando os algoritmos para que identifiquem os padrões dentro do conjunto de dados de treinamento e categorizem os objetos de entrada com base nos padrões que o próprio sistema identifica. Os algoritmos analisam a estrutura subjacente dos conjuntos de dados, extraindo informações úteis ou recursos deles. A tarefa mais comum em aprendizado não supervisionado é a clusterização, onde o conjunto de dados é dividido em grupos com características semelhantes.

No aprendizado por reforço, também não há um conjunto de dados com pares de entradas e saídas. Em vez disso, existe um agente que faz observações, realiza ações no ambiente e recebe uma recompensa em troca (GÉRON, 2017). Neste trabalho, foi utilizado aprendizado por reforço para tratar o problema de locomoção em personagens fisicamente simulados.

2.2.2 Redes Neurais

Uma rede neural é um modelo computacional composto por vários elementos de processamento interconectados, chamados neurônios. Esta estrutura é composta por camadas: uma camada de entrada, as camadas intermediárias, chamadas camadas ocultas, e a última, a camada de saída. Cada neurônio recebe um conjunto de estímulos (valores de entrada) e produz um valor correspondente. Cada valor de entrada está associado a um peso, correspondente à ativação desse estímulo de entrada. Assim, por meio de uma função, a rede neural calcula uma saída específica com base em seus pesos internos. O objetivo principal de uma rede neural é

encontrar a relação entre um conjunto de variáveis de entrada e as saídas. Depois disso, ela é capaz de prever valores de saída para uma determinada entrada, de acordo com a relação aprendida.

A rede neural precisa passar por um processo de treinamento onde várias entradas e suas saídas desejadas são mostradas. Para cada instância de treinamento, a rede neural altera seus pesos internos, num esforço para obter o conjunto de saída desejado para um determinado conjunto de entrada. Esse processo continua até que um critério de parada seja atendido. Após essa fase, essa rede memoriza seu aprendizado salvando os pesos encontrados, sendo capaz de predizer a saída correspondente a uma nova entrada.

2.2.3 Aprendizado por Reforço

Aprendizado por reforço tem chamado muita atenção atualmente com seus resultados em várias aplicações: videogames, como jogos de *Atari* (MNIH *et al.*, 2013) e *Starcraft* (VINYALS *et al.*, 2019); jogos de tabuleiro, como Xadrez (LAI, 2015) e GO (SILVER *et al.*, 2017); animação de personagens articulados (PENG *et al.*, 2018; YU *et al.*, 2018); e robótica (YANG *et al.*, 2020; LIANG *et al.*, 2018).

Com o aprendizado por reforço, um agente aprende a tomar decisões diante de uma determinada situação por tentativa e erro (SUTTON; BARTO, 2018). Durante o processo de aprendizado, o agente recebe uma recompensa ou uma punição para avaliar se cada ação escolhida foi boa ou não. O objetivo principal é que, através da exploração do ambiente, o agente encontre uma política ótima que faz um mapeamento ideal de um estado para uma ação possível naquele estado que tenta maximizar a recompensa acumulada (NOBLEGA, 2019). Avaliar o quanto um programa deve explorar novos estados e ações (*exploration*) ou estados e ações já conhecidos por explorações anteriores (*exploitation*) é um desafio em aprendizado por reforço. Uma maneira para descrever problemas de aprendizado por reforço é utilizar o formalismo matemático de Processos de Decisão de Markov.

Processos de Decisão de Markov (*Markov Decision Process - Markov Decision Process* (MDP)) é um modelo matemático que descreve problemas de tomada de decisão sequencial que podem sofrer influência de fatores aleatórios. Eles possuem a propriedade de Markov que diz que o estado atual depende apenas de um número finito de estados anteriores, e o mais simples é o MDP de primeira ordem, em que o estado atual depende apenas do estado anterior e não do seu histórico (SUTTON; BARTO, 2018). Segundo Arulkumaran *et al.* (2017),

MDP é definido como uma 5-tupla $\langle S, A, T, R, \gamma \rangle$, onde:

- S é o conjunto dos possíveis estados do ambiente;
- A é o conjunto das ações que podem ser executadas em um estado;
- T é a função de transição $T : S \times A \times S \Rightarrow [0,1]$, que representa a probabilidade de alcançar o estado $s' \in S$ se a ação $a \in A$ for executada estando em $s \in S$, $T(s,a,s')$;
- R é a função de recompensa $R : S \times A \Rightarrow \mathbb{R}$, que mede a qualidade de cada estado alcançado pela execução da respectiva ação $a \in A$ no estado $s \in S$, $R(s,a)$;
- $\gamma \in [0,1]$ é o fator de desconto, que representa a preferência do agente pelas recompensas atuais sobre as recompensas futuras.

Para cada tempo t no MDP, o agente encontra-se em um estado s_t do ambiente. Ele executa uma ação a_t, dentre as ações disponíveis para aquele estado. A função de transição $T(s_t, a_t, s'_{t+1})$ define a probabilidade de atingir o estado s' no tempo $t+1$, para essa ação executada em s. Depois da mudança de estado o agente recebe uma recompensa $R(s,a)$ e o fator de desconto indica quanto as recompensas futuras são importantes em comparação com as atuais.

Para solucionar um MDP, é necessário encontrar uma política (π) que representa um mapeamento de uma ação para cada estado do ambiente. As recompensas obtidas em cada estado são usadas para calcular a utilidade esperada (U_h) do histórico das ações executadas, dessa forma podendo medir a qualidade da política:

$$U_h = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + ... + \gamma^n R(s_n). \tag{2.2}$$

A utilidade esperada de um estado s usando uma política π pode ser definida como:

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)\right]. \tag{2.3}$$

Para escolher a melhor ação a ser executada em um estado, busca-se a utilidade máxima esperada para aquele estado, ou seja, o agente deve escolher a ação que maximize sua recompensa esperada. Segundo Russell e Norvig (2002), a utilidade de um estado é a recompensa imediata correspondente a esse estado mais a utilidade descontada esperada do próximo estado, assumindo que o agente escolha a ação ótima. Esse cálculo é feito utilizando a equação de Bellman:

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U(s'). \tag{2.4}$$

Com o valor das utilidades dos estados se constrói uma política ótima (π^*), obtendo as ações que maximizem a utilidade esperada do estado subsequente:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} T(s,a,s')U(s'). \tag{2.5}$$

Para construir as políticas ótimas, duas abordagens podem ser usadas: iteração de valor e iteração de política. Na iteração de valor, calcula-se a utilidade de cada estado usando a equação de Bellman (2.4) e, de posse dos valores de cada estado, a melhor ação é escolhida (2.5). Na iteração de política, usa-se a equação de Bellman (2.4) sem o operador max, usando como referência a ação obtida pela política atual. A partir de uma política inicial aleatória, a cada iteração, a política vai sendo aperfeiçoada de maneira gulosa, buscando modificar as ações recomendadas para cada estado (FRANÇA, 2019).

2.3 Ferramentas utilizadas

Nesta seção, são discutidos os conceitos fundamentais das ferramentas Unity e ml-agents, que são utilizadas para criar e treinar agentes de aprendizado por reforço em ambientes de simulação. A Unity é uma plataforma de desenvolvimento para criar jogos e aplicações interativas, enquanto que o ml-agents é um *plugin open-source* para a Unity que permite aos desenvolvedores aplicar algoritmos de aprendizado por reforço para simulações dentro da Unity.

2.3.1 Unity

Neste trabalho os ambientes para os experimentos foi utilizada a Unity 3D, uma plataforma de desenvolvimento que inclui um motor de jogo feito pela Unity Technologies. Sendo uma das mais populares do mundo, ela domina cerca de 50% do mercado (UNITY, 2020b). Sendo uma plataforma de desenvolvimento de jogos e aplicações interativas que permite a criação de ambientes virtuais para simulação e treinamento de agentes de aprendizado por reforço. Ele oferece uma variedade de ferramentas de desenvolvimento, incluindo modelagem 3D e 2D, física, iluminação e som, que permitem a criação de ambientes altamente realistas e imersivos. Além disso, tem uma ampla comunidade de desenvolvedores e uma vasta variedade de recursos, incluindo modelos, tutoriais e exemplos, o que facilita o desenvolvimento de ambientes personalizados com suporte às linguagens C# e JavaScript.

A Unity suporta diversos *plug-ins* para acrescentar diferentes componentes, tais como: realidade virtual, realidade aumentada, inteligência artificial, além de possuir uma loja

onde diversos *assets* podem ser adquiridos. Existe a versão Personal, que é gratuita, o que torna a Unity mais acessível a desenvolvedores de jogos e pesquisadores.

O ambiente da Unity possui um motor físico integrado (PhysX) que suporta simulação em tempo real de várias entidades físicas, tais como: corpos rígidos, juntas, corpos macios, tecidos, fluidos, molas, detecção de colisão e algoritmos de resposta, com foco na aplicação para a indústria de jogos (MACIEL *et al.*, 2009).

Segundo Juliani *et al.* (2020) a Unity apresenta diversas características que a tornam uma boa escolha para simulações de ambientes de aprendizado por reforço, tais como:

- Alta fidelidade gráfica;
- Mecanismos de física integrados;
- Linguagem de scripts rica para controle das animações;
- Criação de cenários com vários agentes.
- Simulação rápida e distribuída;
- Controle flexível dos componentes da simulação.

2.3.2 ML-Agents

A Unity recentemente apresentou o kit de ferramentas ML-Agents, um projeto de código aberto, que permite que pesquisadores e desenvolvedores convertam jogos e simulações em ambientes de aprendizado, onde agentes inteligentes podem ser treinados usando algoritmos de aprendizado por reforço profundo (UNITY, 2020a). Ele oferece implementação com base no Tensorflow dos algoritmos *Soft Actor-Critic* (SAC) e Proximal Policy Optimization (*Proximal Policy Optimization* (PPO)).

Segundo Juliani *et al.* (2020) e a documentação (UNITY, 2020a), o ML-Agents é composto de cinco elementos de alto nível, que podem ser visualizados na Figura 6:

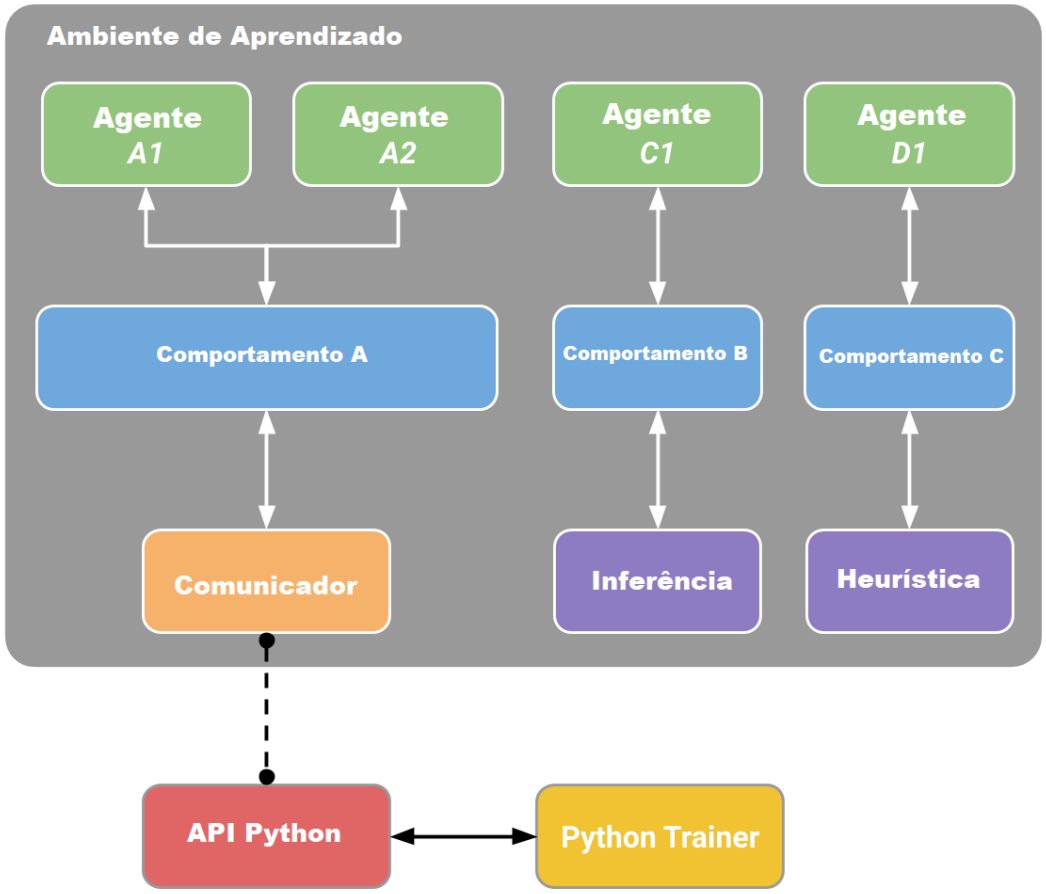
- Ambiente de aprendizado: composto por uma cena na Unity, contém os agentes e todos os elementos necessários para eles fazerem suas observações, agirem e aprenderem. Possui dois componentes básicos:
  - Agente: um *gameobject* dentro da cena da Unity com capacidade de fazer observações do ambiente, realizar ações e receber recompensas. Cada agente está vinculado a um comportamento.
  - Comportamento: responsável por receber as observações e atribuir as recompensas ao agente do ambiente. Pode ser de três tipos: Aprendizagem, não possui uma política

vinculado e está pronto para ser treinado; Heurística, que contém um script que diz ao agente o que fazer; e Inferência, que usa um modelo já treinado.

- API Python: é a interface em Python para interagir e manipular um ambiente de aprendizado, e não faz parte do Unity. Possibilita utilizar outros algoritmos de aprendizado por reforço, além dos implementados pelo ML-Agents.
- Comunicador: faz parte do ambiente de aprendizado e permite que os componentes da Unity se comuniquem com a API Python.
- Python Trainer: contém os algoritmos de aprendizado de máquina utilizados para treinar os agentes.
- Gym Wrappers: são *wrappers* para interagir com o ambiente de aprendizado, e possibilitam usar os algoritmos de aprendizado por reforço implementados pela openai[1].

Os experimentos apresentados neste trabalho foram realizados usando a versão 3 do ML-Agents[2].

Figura 6 – Estrutura do ML-Agents.



Fonte: (UNITY, 2020a)

[1] https://github.com/openai/gym
[2] https://github.com/Unity-Technologies/ml-agents/tree/release_3

3 TRABALHOS RELACIONADOS

Neste capítulo é apresentada uma síntese dos trabalhos relacionados que servem de base para esta pesquisa. São abordados trabalhos que tratam de animação de personagens fisicamente simulados, principalmente os que envolvem o uso de aprendizado por reforço profundo. Um resumo é inicialmente apresentado, seguido de uma comparação entre os trabalhos discutidos e o trabalho proposto.

A animação de personagens fisicamente simulados tem sido um tópico de interesse significativo na computação gráfica há muito tempo. O trabalho elaborado por Geijtenbeek e Pronost (2012) apresenta uma visão geral sobre o tema, mostrando conceitos e os principais trabalhos na área até 2012, servindo como uma boa introdução ao tema e aos desafios da área.

Animações mais naturais e suaves para personagens fisicamente simulados podem ser obtidas planejando cuidadosamente o projeto de movimentos mais específicos. Os métodos que seguem esse direcionamento acabam perdendo a capacidade de generalização para novos personagens e situações. É o caso do uso de movimentos capturados (ZORDAN; HODGINS, 2002) ou informações intrinsecamente relacionadas à morfologia específica de um personagem (COROS *et al.*, 2011) para guiar a geração do movimento desejado.

Embora uma das características mais atraentes do uso de DRL seja a expectativa de que os personagens possam aprender comportamentos por conta própria, emergindo com o mínimo de informação necessária para definir o resultado desejado, tal objetivo precisa ser equilibrado com a necessidade de não perder tanto controle sobre a animação final. Nesse sentido, combinar DRL com abordagens mais específicas provavelmente tem sido a razão do recente crescimento do uso de DRL no contexto de controle de personagens articulados.

Avanços surpreendentes têm sido obtidos usando movimentos capturados para guiar o processo de aprendizagem e tornar os resultados mais naturais e bem parecidos com comportamentos humanos reais (PENG *et al.*, 2018; LIU; HODGINS, 2018; PARK *et al.*, 2019; CHENTANEZ *et al.*, 2018).

Em Peng *et al.* (2018), personagens de diferentes morfologias (humano, robô, dinossauro e dragão) aprendem diversos tipos de movimentos de locomoção e acrobacias utilizando uma função de recompensa que premia por seguir poses de clipes curtos de movimentos capturados e por atingir objetivos secundários. A política aprendida é capaz de interagir com o ambiente e se recuperar de perturbações, além de explorar um método para integrar diferentes tipos de movimento em um único personagem.

Já Liu e Hodgins (2018) aplicam dados de captura de movimento de jogadores de basquete para criar animações de dribles. Para lidar com a dinâmica da bola, são treinados separados o controle da locomoção e o controle de braço.

Park *et al.* (2019) criam animações a partir de uma base de dados de movimentos capturados, não organizados e pouco rotulados, onde os personagens podem interagir entre si e com o ambiente, utilizando um gerador de movimentos e usando redes neurais recorrentes para o controlador. O gerador de movimentos guia a simulação usando uma sequência de quadros de movimentos futuros para serem rastreados usando DRL.

Chentanez *et al.* (2018) utilizam captura de movimentos para gerar os ângulos desejados para os controladores PD e utilizam DRL para calcular os ganhos e os torques corretivos do personagem. Também é mostrado como generalizar a técnica para uma grande base de dados de movimentos.

Por outro lado, alguns pesquisadores ainda tentam seguir a linha de abordagens minimalistas para o problema de aprendizagem generalizado de locomoções (HEESS *et al.*, 2017; YU *et al.*, 2018). Heess *et al.* (2017) treinam agentes para andar e correr em terrenos irregulares e dinâmicos com dificuldades crescentes utilizando uma função de recompensa simples. Os movimentos de correr, pular, agachar e virar surgem naturalmente, na medida em que essa habilidade é exigida pelo ambiente, sem orientação explícita na função de recompensa. Yu *et al.* (2018), por exemplo, usam DRL essencialmente focando nas hipóteses de que locomoções naturais tendem a ser simétricas e a gastar pouca energia. Além disso, termos específicos de recompensa são definidos para controlar a velocidade da locomoção. O controle da velocidade combinado com a premiação pelo baixo uso de energia faz com que a locomoção mais adequada para a respectiva velocidade surja naturalmente. Entretanto, um novo treinamento é feito para cada velocidade desejada diferente. Em contraste, este trabalho propõe que a velocidade desejada possa ser modificada pelo usuário em tempo real, durante a simulação da rede já treinada.

Essa crescente disseminação do uso de DRL na animação de personagens pode também ser notada pela publicação de trabalhos (BERGAMIN *et al.*, 2019) que consistem basicamente em realizar o processo de treinamento de DRL para tornar um personagem simulado capaz de imitar as animações geradas a partir de métodos cinemáticos anteriores (CLAVET, 2016). Ou seja, DRL permite que técnicas cinemáticas, que geralmente possuem um maior grau de controle sobre a animação, possam ser convertidas para abordagens fisicamente simuladas, para permitir a interação automática com o ambiente.

Dentre outras direções de pesquisa em DRL, podem ser citados trabalhos que investem no planejamento de transições entre fragmentos de controle (LIU *et al.*, 2016; LIU; HODGINS, 2017); que exploram a variação de terrenos (PENG *et al.*, 2016; HEESS *et al.*, 2017); que utilizam o processamento em GPU (LIANG *et al.*, 2018; CHENTANEZ *et al.*, 2018) para acelerar o processo de treinamento; ou que investem em uma estratégia, conhecida como *Curriculum Learning* (*Curriculum Learning* (CL)), que torna o ambiente ou o contexto de treinamento gradativamente mais difícil para facilitar o progresso da aprendizagem (HEESS *et al.*, 2017; YU *et al.*, 2018).

Considerando a área de animação de personagens simulados em geral, uma das direções de pesquisa consiste em investir na redução e simplificação da representação do controle dos graus de liberdade (*Degrees of Freedom* - DOF) internos do personagem, correspondentes aos ângulos (orientações) das suas articulações. Assumindo que um espaço de busca adequadamente reduzido se comporta melhor sem elementos de controle não significativos e que podem teoricamente ser ignorados, métodos de otimização podem se beneficiar de uma melhor representação do controle. Dessa forma, a análise de componentes principais (*Principal Component Analysis* - *Principal Component Analysis* (PCA)) (SAFONOVA *et al.*, 2004; YE; LIU, 2008) e a análise modal (KRY *et al.*, 2009; NUNES *et al.*, 2012) são possíveis abordagens usadas para reduzir a representação do controle.

No contexto de DRL, Ranganath *et al.* (2019), por exemplo, usam uma base reduzida de movimentos para definir uma representação simplificada, mas ainda significativa, para a saída da rede, facilitando o processo de aprendizagem. Os movimentos dessa base correspondem a oscilações coordenadas dos DOF, chamadas de coativações. Essas coativações são extraídas automaticamente a partir de uma base de dados capturados, contendo movimentos semelhantes aos movimentos a serem reproduzidos usando DRL.

Este trabalho utiliza o kit de ferramentas ML-Agents que possibilita a utilização do estado da arte em algoritmos de aprendizado por reforço profundo em simulações e jogos desenvolvidos utilizando o motor de jogos Unity 3D. Uma boa introdução ao ambiente é fornecida por Juliani *et al.* (2020), em que se discute sobre os ambientes de simulação para aprendizado de máquina e apresenta o ML-Agents com a Unity como uma alternativa de plataforma rica em complexidade visual, flexível, interativa e facilmente configurável.

Dentro do contexto de animação de personagens articulados fisicamente simulados, o próprio artigo de Juliani *et al.* (2020) apresenta alguns ambientes de exemplo, *walker*, *crawler*,

reacher e *worm*, que podem servir como *benchmarks* e pontos de partidas para pesquisas nessa área. Booth e Booth (2019) criaram um conjunto de *benchmarks* para tarefas de controle contínuo envolvendo locomoção de personagens, utilizando o ambiente ML-Agents, que implementa ambientes inspirados nos trabalhos de *Deep Mind Control Suite* (TASSA *et al.*, 2018), Heess *et al.* (2017) e Peng *et al.* (2018). No trabalho de Booth e Ivanov (2020), eles usaram o ML-Agents para criar um controle realístico de personagens utilizando dados de captura de movimento e interação com entrada de usuários.

Ademais, outros trabalhos também utilizam o ML-Agents dentro do contexto de animação: Krupnik *et al.* (2020) treinam agentes para interagir cooperando ou competindo; Bapst *et al.* (2019) apresentam o controle de agentes em um conjunto de tarefas de construção simuladas; e Lagula e Karlsson (2020) comparam o uso de imitação de movimentos com uma versão sem usar imitação.

3.1 Comparação

O Quadro 1 a seguir faz uma comparação entre os principais trabalhos que combinam animação de personagens articulados fisicamente simulados e aprendizado por reforço profundo e que influenciaram a realização deste trabalho.

Para a avaliação dos trabalhos relacionados com este trabalho foram analisados os seguintes pontos:

1. Movimentos emergentes - Trabalhos que permitem e exploram o surgimento espontâneo de movimentos no treinamento, sem depender de movimentos de referência;
2. Funções de recompensa generalizáveis - Trabalhos que possibilitam que as mesmas funções de recompensa possam ser utilizadas em outros personagens com morfologias diferentes;
3. Controle em tempo real - Trabalhos que analisam a influência de alterações no ambiente ou modificações nas variáveis de entrada após o treinamento, possibilitando um controle do personagem em tempo real, durante a execução da simulação física com a rede já treinada.

Quadro 1 – Comparação entre os trabalhos relacionados.

Trabalho	**Movimentos emergentes**	**Funções de recompensa generalizáveis**	**Controle em tempo real**
Juliani *et al.* (2020)	Sim	Não	Não
Peng *et al.* (2018)	Não	Sim	Não
Heess *et al.* (2017)	Sim	Sim	Não
Bergamin *et al.* (2019)	Não	Não	Sim
Yu *et al.* (2018)	Sim	Sim	Não
Park *et al.* (2019)	Não	Não	Sim
Chentanez *et al.* (2018)	Não	Não	Não
Ranganath *et al.* (2019)	Não	Sim	Não
Booth e Booth (2019)	Sim	Não	Não
Booth e Ivanov (2020)	Não	Não	Sim
Este trabalho	**Sim**	**Sim**	**Sim**

Fonte: Próprio autor (2022).

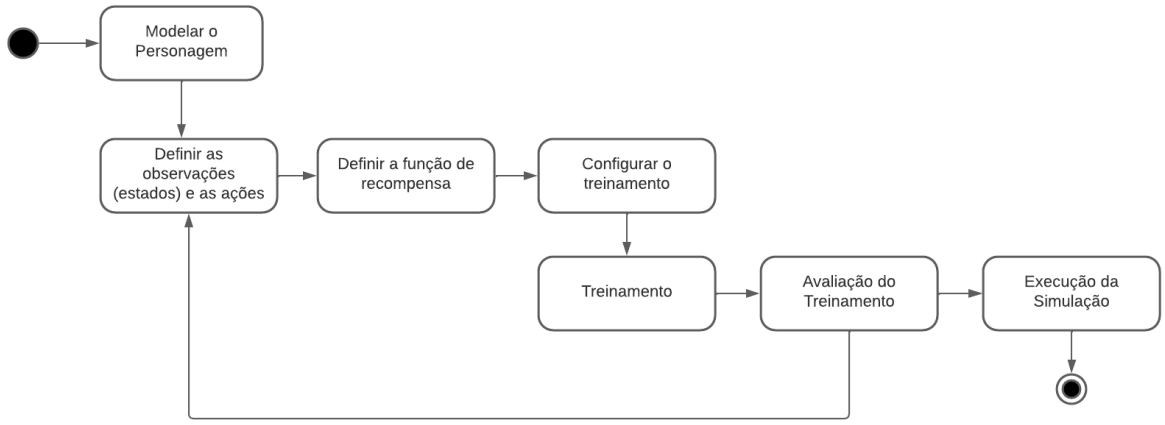
4 ESTRUTURA DE CONTROLE

Neste capítulo são apresentadas as etapas necessárias para realizar os experimentos utilizados neste trabalho, uma visão geral da estrutura de controle proposta, as especificações dos personagens usados e mais detalhes sobre os componentes correspondentes às principais contribuições deste trabalho: a definição das recompensas e os ajustes realizados na entrada e na saída da rede. Esses ajustes são responsáveis por permitir tanto o controle em tempo real sobre a velocidade do personagem quanto o controle de simetria do movimento gerado pelo personagem.

4.1 Etapas para a Realização dos Experimentos

O diagrama de atividades a seguir (Figura 7) apresenta os passos para a criação de um ambiente de treinamento de aprendizado por reforço profundo para personagens articulados fisicamente simulados utilizando a Unity e o ML-Agents.

Figura 7 – Fluxo de atividades para o treinamento do agente usando a Unity e o ML-Agents



Fonte: elaborado pelo autor (2022).

4.1.1 Modelar o personagem

Os personagens são modelados utilizando corpos rígidos conectados por juntas. São utilizadas formas geométricas simples (cápsulas, cubos e esferas) para facilitar o tratamento de colisões. As juntas utilizadas nos exemplos apresentados neste trabalho são de dois tipos: juntas dobradiças, que possuem um grau de liberdade, e juntas esféricas, que possuem três graus de liberdade. Cada junta apresenta limites angulares inferior e superior para simular a amplitude do movimento real. Os controladores PD já estão disponíveis nas juntas, não sendo necessária sua

implementação. São adicionados às partes que formam o personagem os componentes físicos que permitem que os corpos rígidos sejam influenciados pela simulação. Para que os personagens sejam capazes de serem utilizados como agentes de aprendizado por reforço, se faz necessário adicionar um *script* que herda da classe *Agent*.

4.1.2 Definir as observações (estados) e as ações

As observações (estados) são as informações necessárias para que o agente consiga realizar suas tarefas. Essas informações correspondem às entradas da rede neural. São utilizados dados de posição, orientação, velocidade, velocidade angular e toque (valor discreto informando se o corpo está tocando o chão) para cada parte do personagem. Outras informações podem ser utilizadas caso se deseje controlá-las em tempo real, tais como velocidade e direção desejadas do personagem. As ações são o resultado do uso da política, que mapeia um estado em uma ação adequada. No início do treinamento, essas ações são aleatórias e na medida em que o aprendizado vai ocorrendo, elas vão ficando melhores. As ações geradas são os ângulos desejados de cada grau de liberdade que são utilizados pelos controladores PD para gerar os torques. Outras informações de saída podem ser usadas, tais como o limite de força máxima que pode ser aplicada a cada junta pelo controlador PD.

4.1.3 Definir a função de recompensa

Em tarefas de aprendizado por reforço, a recompensa avalia o quão bom ou ruim foi uma ação executada, dessa forma, guiando o processo de aprendizado. A criação da função de recompensa envolve relacionar uma expressão matemática com um resultado que se deseja alcançar. Após a escolha do movimento que o personagem deve realizar, são analisados quais comportamentos são esperados para esse movimento, para os quais deve ser dada uma recompensa positiva. Por exemplo, para tarefas de locomoção, normalmente, utiliza-se como recompensa o deslocamento ou a velocidade. Tipos de movimento mais complexos envolvem a combinação de vários termos para se produzir o movimento adequado, como o utilizado no Experimento 5.2.5, em que foram combinados termos necessários para pular, para se deslocar horizontalmente e para manter os pés juntos. A definição de uma função de recompensa adequada, muitas vezes, envolve um processo de tentativa e erro.

4.1.4 Configurar o treinamento

A definição do algoritmo que será utilizado para o treinamento e a configuração dos hiper-parâmetros do algoritmo e da rede neural são feitas em um arquivo separado, que é acessado no início do treinamento. As configurações dos parâmetros da simulação física são feitas na própria Unity, no componente Physics, sendo possível definir, por exemplo, o tamanho do passo de tempo usado na simulação física ou a aceleração da gravidade. Para possibilitar um treinamento mais rápido, vários agentes são treinados em paralelo. As escolhas das configurações e dos hiper-parâmetros dependem dos agentes que são treinados, da avaliação da dificuldade do treinamento e do algoritmo utilizado (PPO ou SAC).

4.1.5 Treinamento

Para realizar o treinamento, um processo é iniciado fora do ambiente da *Unity* através do *Python trainers*, um pacote que contém todos os algoritmos DRLs utilizados pelo ML-Agents para o treinamento dos agentes. Os algoritmos de aprendizado por reforço são implementados em Python, tanto utilizando a biblioteca *Tensorflow*, na versão do ML-Agents que foi utilizada neste trabalho, quanto utilizando a biblioteca *Pytorch*, nas versões mais atuais da ferramenta. O pacote expõe um único utilitário de linha de comando, chamado mlagents-learn, que oferece suporte a todos os métodos e opções de treinamento. As várias opções são utilizadas para definir alguns parâmetros do treinamento e a localização do arquivo de configurações onde estão as definições dos hiper-parâmetros a serem usados durante o treinamento. As sessões de treinamento podem ser feitas tanto utilizando um arquivo executável do ambiente quanto diretamente no ambiente da *Unity*. O processo de treinamento para geração de animações de personagens fisicamente simulados é demorado, podendo levar cerca de 10 horas, como neste trabalho, ou até dias, como em experimentos realizados em Peng *et al.* (2018).

4.1.6 Avaliação do treinamento

Além dos aspectos visuais, que podem ser observados acompanhando a evolução do treinamento através da *Unity* ou do executável da simulação, algumas informações mais objetivas também podem ser usadas para a avaliação do treinamento. A cada determinada quantidade de passos do treinamento, são geradas estatísticas que permitem acompanhar a evolução do aprendizado. Exemplos de informações úteis utilizadas são a recompensa acumulada, que deve

crescer na medida em que o treinamento progride, e a entropia, que tende a diminuir durante o processo. Essas estatísticas são acompanhadas no utilitário do *Tensorflow* chamado *Tensorboard*. A Figura 8 ilustra a visualização de dois treinamentos distintos, fornecida pelo *Tensorboard*, de quatro parâmetros de treinamento: a recompensa acumulada, o tamanho dos episódios já executados, a entropia e a taxa de aprendizado. O eixo x representa o números de passos na simulação em todos os gráficos.

A recompensa acumulada é uma medida importante do desempenho de um agente de aprendizado por reforço sendo obtida pelos agentes ao longo de um episódio. Durante um treinamento bem-sucedido, é esperado que a recompensa acumulada média do episódio aumente, pois isso indica que o agente está aprendendo a tomar decisões mais eficazes e obtendo recompensas maiores. Acompanhar o desempenho do agente através da recompensa acumulada média pode ajudar a identificar quando o agente atingiu um platô ou se ainda há espaço para melhoria. Além disso, serve para comparar diferentes configurações de modelo e identificar qual é a mais eficaz. Neste gráfico o eixo y representa os valores médios das recompensas obtidas pelos agentes ao longo dos episódios.

Considerando o tamanho dos episódios, na medida em que o treinamento evolui, é esperado que os personagens tomem decisões melhores e sejam capazes de manter episódios durando mais. Isso indica que o agente está aprendendo e se tornando mais eficiente em tomar decisões. Portanto, quanto maior o tempo do episódio, mais o agente obtém resultados melhores. Neste trabalho foi estabelecido um tamanho máximo de 5 mil passos para cada episódio (JULIANI *et al.*, 2020), com cada passo correspondendo à observação coletada e ação executada. No entanto, foi aplicado um término antecipado (PENG *et al.*, 2018), caso os personagens tocassem outras partes do corpo no chão, além dos pés. Isso permite que o treinamento seja interrompido quando o agente atinge uma situação que seria difícil se recuperar. Neste gráfico o eixo y representa os valores médios das durações dos episódios em passos pelos agentes.

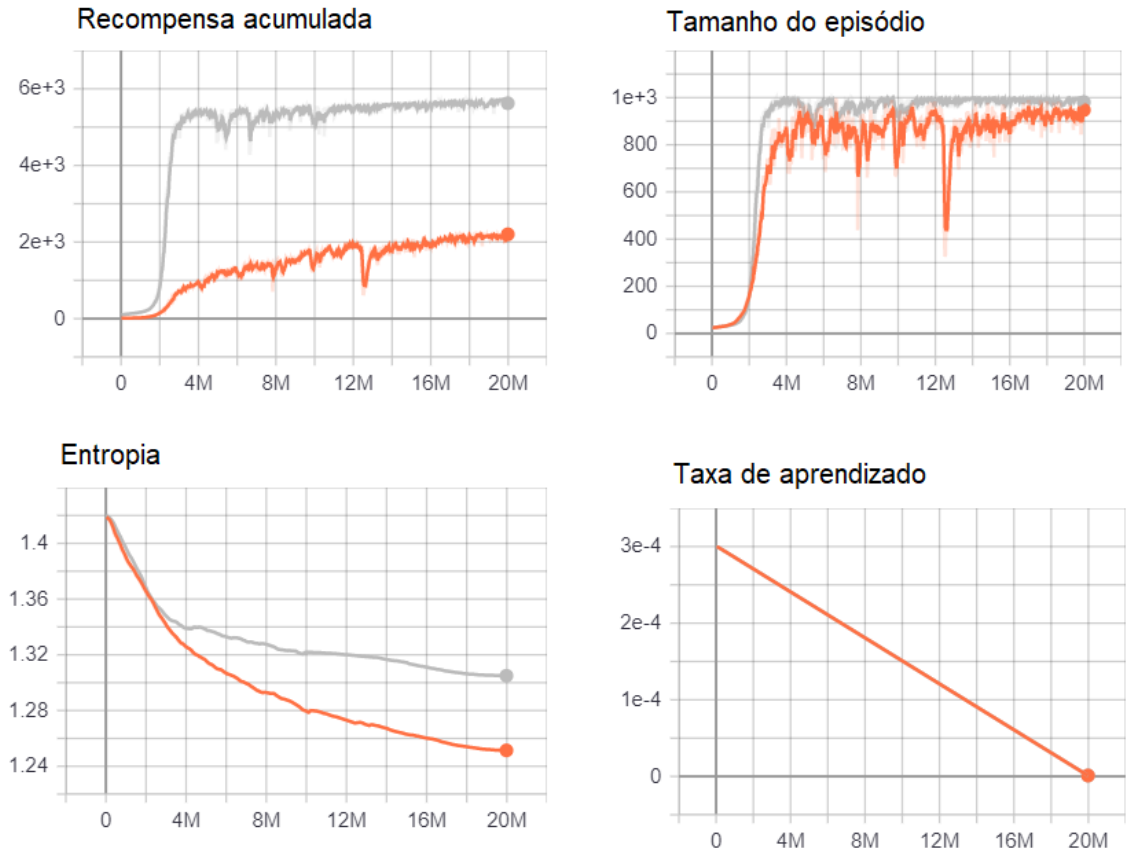
A entropia é uma medida da incerteza ou imprevisibilidade no aprendizado por reforço, usada para medir a aleatoriedade das decisões tomadas pelo modelo. Quanto maior a entropia, maior a incerteza e menor a confiança do modelo em suas decisões. Durante um processo de treinamento bem-sucedido, é esperado que a entropia do modelo diminua lentamente. Isso indica que o modelo está aprendendo a tomar decisões mais informadas e confiantes. Quando a entropia se mantém constante, pode indicar que o modelo está tendo dificuldades para aprender

com as novas informações, e pode ser necessário rever a arquitetura do modelo ou ajustar os parâmetros. Neste gráfico o eixo y representa os valores do hiperparâmetro beta (valor inicial 0,005) ao longo dos episódios.

A taxa de aprendizado é uma medida da velocidade com que o algoritmo de aprendizado por reforço atualiza sua política. Ela determina o passo necessário para encontrar a política que maximiza a recompensa esperada. É esperado que a taxa de aprendizado diminua com o tempo para garantir que o algoritmo não passe por cima da política ideal, mas também para garantir que o algoritmo não fique preso em um sub-ótimo. Neste gráfico o eixo y representa os valores da taxa de aprendizado (valor inicial 0,0003) ao longo dos episódios.

Esses gráficos vão sendo atualizados durante o processo de treinamento e podem ser bastante úteis para entender a evolução do aprendizado e realizar correções adequadas para otimizar o modelo. Como se trata de um processo demorado, caso essas métricas não estejam evoluindo como esperado, pode-se encerrar o treinamento antecipadamente e já revisar os passos anteriores.

Figura 8 – Gráficos do *Tensorboard* correspondentes a dois treinamentos distintos. Um deles representado na cor cinza e um outro na cor laranja.



Fonte: elaborado pelo autor (2022).

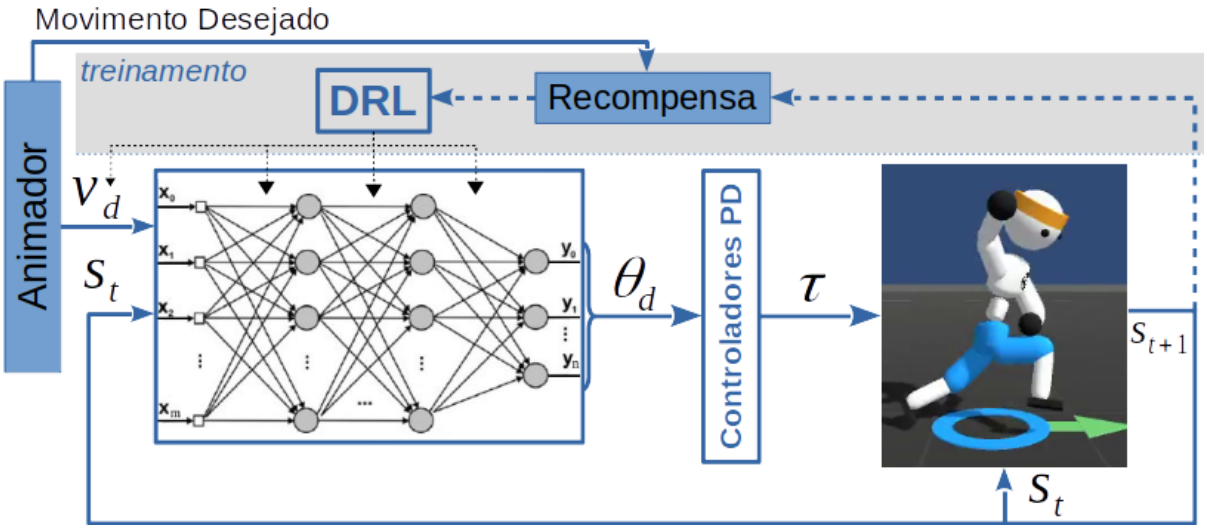
4.1.7 Execução da simulação

Após o término do treinamento, é gerado um arquivo contendo a política aprendida. Esse arquivo gerado pode ser executado tanto no ambiente de simulação na *Unity* quanto no executável da cena, sendo anexado aos agentes para que possam executar em modo inferência. Nesse modo, o agente utiliza a política aprendida para executar suas ações dentro da simulação. Mesmo com os dados do treinamento sendo acompanhados no gráfico do *Tensorboard* e evoluindo de acordo com o esperado, o resultado da animação gerada pode não ser a desejada, sendo necessário recomeçar os passos. Uma aplicação útil para as redes que são treinadas com sucesso é usá-las na animação de *Non-Player Character* (NPC) em jogos ou em ambientes de simulação em geral.

4.2 Visão Geral da Estrutura de Controle

A Figura 9 apresenta a visão geral do funcionamento do controlador, que é representado por uma rede neural treinada usando DRL e que conduz a simulação física do personagem para gerar a animação resultante.

Figura 9 – Visão geral da estrutura de controle e do treinamento da rede usando DRL.



Fonte: elaborado pelo autor (2022).

A simulação física consiste em, a partir de um estado do sistema s_t, definido em um determinado instante t, integrar as equações diferenciais de movimento da estrutura modelada para obter um novo estado s_{t+1} correspondente ao próximo instante de tempo $t + \delta t$ considerado. Entretanto, apenas a simulação física não é suficiente para gerar animação natural para sistemas

ativos. Se nenhuma atuação interna for exercida por parte do personagem, ele se comporta como um sistema passivo, que obedece as regras físicas definidas mas não se movimenta naturalmente como um humano real. No caso, o personagem agiria como uma boneca de pano (*ragdoll*) e apenas cairia sob o efeito da gravidade. O problema de controle consiste em definir os torques τ a serem aplicados em cada instante da simulação física pelos atuadores localizados nas articulações do personagem. Esses torques aplicados são automaticamente adicionados às equações de movimento e influenciam no cálculo de cada novo estado s_{t+1}.

No contexto deste trabalho, uma rede neural é responsável por gerar os ângulos desejados para os Controladores PD, que geram os torques necessários para manter o movimento do personagem e funcionam como uma camada de controle de mais baixo nível, responsável por calcular os torques capazes de atingir as orientações desejadas fornecidas pela rede.

Para funcionar de maneira adequada, a rede precisa ser treinada e uma função de recompensa definida pelo animador avalia a qualidade das ações executadas de acordo com um tipo de movimento desejado escolhido. Os movimentos não são especificados em detalhes. Movimentos adequados, de acordo com as características do movimento definido através da recompensa, emergem naturalmente usando DRL. O método de aprendizado da rede basicamente explora o espaço de estados e ações do sistema, através da execução exaustiva de várias tentativas de simulação, chamados de episódios, e avalia para cada estado se uma determinada ação contribui para aumentar o valor da função de recompensa. Essas informações locais de recompensa em cada estado vão sendo computadas e combinadas com as recompensas dos estados vizinhos de modo que os caminhos compostos por sequências de estados e ações ([estado] $\rightarrow$ [ação tomada] $\rightarrow$ [estado alcançado devido à respectiva ação tomada] $\rightarrow$ [nova ação] $\rightarrow \ldots$) que possuem o maior acúmulo de recompensas possam ser identificados.

Após o processo de aprendizagem, durante a simulação em tempo real, a rede treinada é responsável por decidir qual melhor ação local θ_d (saída da rede) tomar a partir do estado atual do sistema s_t (entrada da rede), embora cada ação já tenha sido analisada num contexto global, dentro de vários possíveis caminhos explorados (episódios) durante o treinamento.

4.3 Personagens

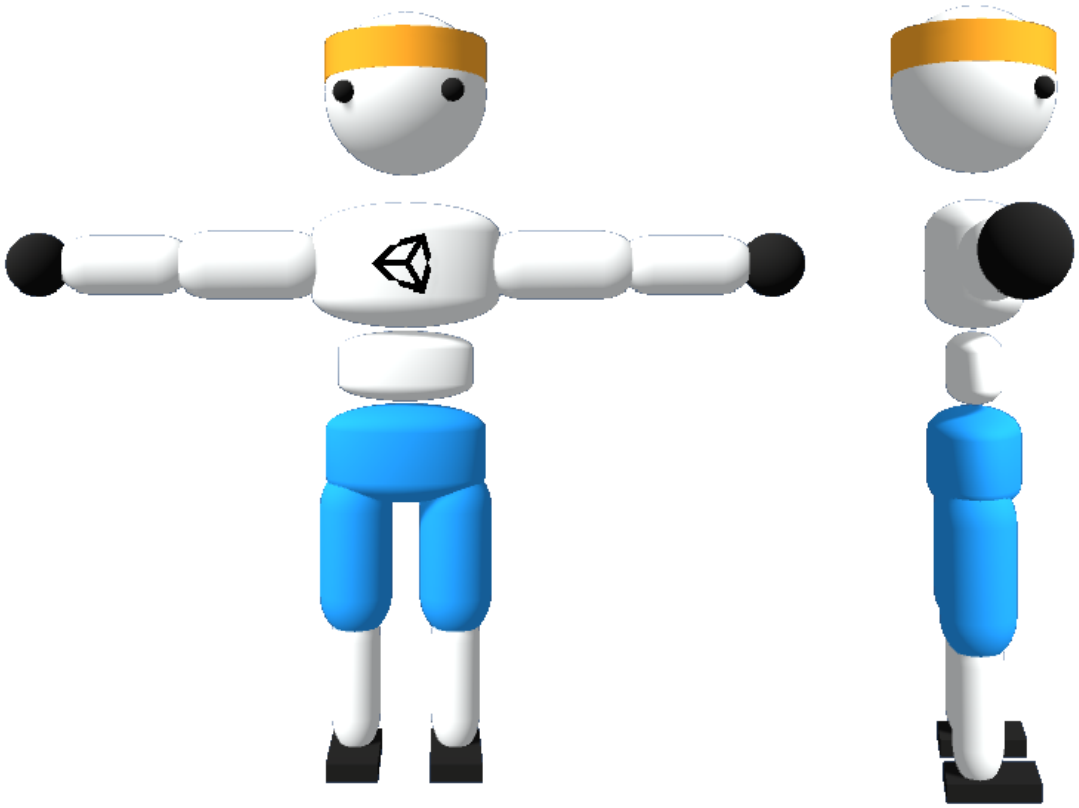
Dois personagens com diferentes morfologias são usados nos experimentos. O primeiro personagem, chamado Humano 3D (Figura 10), é um personagem tridimensional composto de 16 corpos rígidos com 26 graus de liberdade. O quadril corresponde ao corpo

raiz e os outros corpos são conectados através de 13 juntas do tipo *Configurable Joint* da Unity. Os pulsos são fixos. Os 16 corpos rígidos são constituídos pelo quadril, coluna, peito, cabeça, coxas, canelas, pés, braços, antebraços e mãos. Esse modelo corresponde ao mesmo utilizado no ambiente exemplo disponibilizado junto com o ML-Agents, chamado “Walker”, presente em (JULIANI *et al.*, 2020).

No Humano 3D, é utilizada uma rede neural com a entrada de 236 valores referentes a posição, orientação, velocidade linear, velocidade angular e contato com o chão (informação discreta) de cada corpo rígido. A direção apontando do personagem para a localização de um alvo a ser alcançado é também usado como entrada. Para o experimento da velocidade em tempo real, a velocidade a ser alcançada é também usada como entrada.

Para os experimentos usando o Humano 3D, é usada uma rede neural com 3 camadas ocultas, contendo 512 nós cada uma. A saída da rede corresponde a 39 valores referentes aos ângulos desejados dos 26 graus de liberdades das articulações e aos limites máximos de força que os controladores PD podem aplicar em cada uma das 13 articulações.

Figura 10 – Humano 3D.



Fonte: elaborado pelo autor (2022).

O segundo personagem, chamado Dálmata 2D (Figura 11), é bidimensional e consti-

Figura 11 – Dálmata 2D.

Fonte: elaborado pelo autor (2022).

tuído de 14 corpos rígidos e 13 DOFs. O tronco do meio corresponde ao corpo raiz e os outros corpos são conectados através de 13 articulações do tipo *Hinge Joint* da Unity, que são juntas do tipo dobradiça e que permitem rotação em torno de apenas um eixo. O personagem é constituído das três partes do tronco, pescoço, cabeça, duas pernas (coxas, canelas e pés) e as três partes do rabo.

Como entrada da rede, tem-se 98 valores correspondendo a posição, velocidade, velocidade angular e contato com o chão (informação discreta) de cada parte do personagem. Também foi definida uma rede neural com três camadas ocultas, que contém 512 nós cada uma. A saída da rede são os valores correspondentes aos ângulos desejados dos 13 graus de liberdade das articulações do personagem. Para este personagem, os limites máximos de força não fazem parte da saída da rede.

4.4 Recompensas

A principal contribuição deste trabalho corresponde a acessar o nível de controle, a partir da definição de uma recompensa, que o animador pode ter sobre o movimento resultante. Como a definição de uma recompensa no contexto do DRL envolve traduzir características específicas do movimento desejado em uma expressão matemática, essa tarefa geralmente não é trivial. A ideia portanto é mostrar que é possível gerar deliberadamente diferentes tipos de movimento, com características nitidamente perceptíveis, a partir da definição de um conjunto base de recompensas apropriadas, que possa ser identificado como uma ferramenta de controle

útil sobre a animação final. A Figura 1 ilustra bem esse objetivo. Para isso, as recompensas definidas precisam ser simples e de fácil adaptação. Por exemplo, uma recompensa pode ser considerada útil se ela pode ser adaptada a um movimento desejado equivalente, mas aplicado a um outro personagem, com morfologia possivelmente diferente. Em outras palavras, este trabalho estuda quanto de controle sobre o tipo do movimento resultante a simples definição da função de recompensa pode proporcionar ao animador.

As funções de recompensa usadas nos experimentos deste trabalho são definidas como somas ponderadas de termos, que resultam em um único valor para que o critério de melhoria usado no treinamento do DRL seja bem objetivo. Os significados transmitidos através da definição dos termos sendo combinados e dos pesos escolhidos é que são responsáveis por traduzir a subjetividade do desejo do animador. O valor único resultante da função de recompensa é portanto usado para treinar uma política responsável por guiar o personagem. Valores de recompensa positivos são definidos para estimular comportamentos desejados e valores negativos para penalizar comportamentos indesejáveis. Os pesos dos termos foram baseados naqueles propostos por Juliani *et al.* (2020).

Nas subseções a seguir, são mostradas as funções de recompensa usadas para controlar uma série de movimentos: corrida, corrida saltando com uma perna só, pulo, corrida saltando com os dois pés juntos e controle de velocidade.

4.4.1 Controle da Corrida

Nesta subseção, são apresentadas as funções de recompensa para controlar os movimentos de corrida dos personagens Humano 3D e Dálmata 2D. A função de recompensa usada para controlar o movimento de corrida do personagem Humano 3D é definida como

$$r = w_v T_{vel} + w_o T_{ori} + w_h T_h. \tag{4.1}$$

O primeiro termo, que usa o produto escalar, $T_{vel} = \vec{v}_r \cdot \vec{d}_t$, premia o alinhamento da direção do vetor velocidade do corpo raiz, $\vec{v}_r$, com a direção ao alvo, $\vec{d}_t$, um objeto localizado distante do personagem no eixo x. O vetor unitário $\vec{d}_t$ é representado pelo vetor verde mostrado na Figura 13. O segundo termo, $T_{ori} = \vec{f}_h \cdot \vec{d}_t$, recompensa o produto escalar do vetor unitário *forward* da cabeça $\vec{f}_h$, com $\vec{d}_t$, para encorajar que a orientação do personagem também esteja alinhada com a direção ao alvo. O terceiro termo, $T_h = (h_h - h_{lf}) + (h_h - h_{rf})$, premia a altura da cabeça,

h_h, em relação às alturas dos pés esquerdo e direito, h_{lf} e h_{rf}, para encorajar o personagem a correr com a postura ereta. Os pesos w_v, w_o *e* w_h são $0,02$, $0,01$ e $0,01$, respectivamente.

A função de recompensa usada para controlar o movimento de corrida do Dálmata 2D é definida como

$$r = w_{v_x} T_{vel_x} + w_h T_{head} + w_{v_y} T_{vel_y}. \tag{4.2}$$

O primeiro termo, $T_{vel_x} = v_{rx}$, premia a velocidade horizontal do corpo raiz (tronco do meio), para encorajar a locomoção para frente. O segundo termo, $T_{head} = ||\vec{v}_h - \vec{v}_r||$, penaliza o módulo da velocidade relativa da cabeça, obtido pela diferença entre a velocidade da cabeça, $\vec{v}_h$, e a velocidade do corpo raiz, $\vec{v}_r$, para evitar o movimento excessivo da cabeça em relação ao corpo raiz. O terceiro termo, $T_{vel_y} = v_{ry}$, penaliza a velocidade vertical do corpo raiz, com o intuito de estabilizar melhor a corrida. Os pesos, w_{v_x}, w_h e w_{v_y}, são 0,03, -0,01 e -0,01, respectivamente. Os termos da função de recompensa foram adaptados ao novo contexto bidimensional e à morfologia do novo personagem, o que naturalmente resulta em um estilo de locomoção diferente.

4.4.2 Controle da Corrida em uma Perna Só

Nesta subseção, são apresentadas as funções de recompensa para controlar os movimentos de corrida em uma perna só dos personagens Humano 3D e Dálmata 2D. A função de recompensa usada para controlar o movimento de corrida em uma perna só do personagem Humano 3D é definida como

$$r = w_v T_{vel} + w_o T_{ori} + w_h T_h + T_{lf}, \tag{4.3}$$

onde

$$T_{lf} = \begin{cases} -1, & \text{se o pé esquerdo tocar o chão,} \\ 0, & \text{caso contrário.} \end{cases} \tag{4.4}$$

penaliza o personagem quando seu pé esquerdo toca o chão.

A função de recompensa usada para controlar o movimento de corrida numa perna só do personagem Dálmata 2D é definida como

$$r = w_{v_x} T_{vel_x} + w_h T_{head} + w_{v_y} T_{vel_y} + T_{fp} + w_{ht} T_{height}, \tag{4.5}$$

onde

$$T_{fp} = \begin{cases} -1, & \text{se a pata da frente toca o chão,} \\ 0, & \text{caso contrário.} \end{cases} \tag{4.6}$$

Foi adicionado um termo condicional semelhante, T_{fp}, que penaliza o personagem apenas quando sua pata dianteira toca o chão. Também foi usado um termo extra, T_{height} para recompensar a altura da pata dianteira, com $w_{ht} = 0,03$.

4.4.3 Controle do Pulo

Nesta subseção, são apresentadas as funções de recompensa para controlar os movimentos de pulo do personagem Humano 3D e do personagem Dálmata 2D. A função de recompensa usada para controlar o movimento de pulo do personagem Humano 3D é definida como

$$r = w_h T_h + w_j T_j, \tag{4.7}$$

onde

$$T_j = |v_{p_y}|. \tag{4.8}$$

O novo termo, $|v_{p_y}|$, recompensa o valor absoluto da velocidade vertical do corpo raiz (pélvis) com peso de $w_j = 0,02$.

A função de recompensa usada para controlar o movimento de pulo do personagem Dálmata 2D é definida como

$$r = T_{fp} + w_j T_j + w_{ht} T_{height} + w_{hd} T_{head}, \tag{4.9}$$

onde

$$T_j = |v_{ry}|. \tag{4.10}$$

O novo termo, $|v_{ry}|$, recompensa o valor absoluto da velocidade vertical do corpo raiz com peso de $w_j = 0,02$.

4.4.4 Controle da Corrida com os Pés Juntos

Nesta subseção, é apresentada a função de recompensa para controlar o movimento de corrida com os pés juntos do personagem Humano 3D, definida como

$$r = w_v T_{vel} + w_o T_{ori} + w_h T_h + w_{fd} T_{fd}, \tag{4.11}$$

onde

$$T_{fd} = |p_{lf_x} - p_{rf_x}| + |p_{lf_y} - p_{rf_y}| \tag{4.12}$$

penaliza a distância Manhattan dos pés do personagem para manter os pés juntos. Os termos p_{lf_x} e p_{rf_x} são, respectivamente, a coordenada x da posição do pé esquerdo e direito, p_{lf_y} e p_{rf_y} são, respectivamente, a coordenada y da posição do pé esquerdo e direito. Apenas a coordenada lateral da distância dos pés não é penalizada. Como um termo de punição deve ter um peso negativo, foi usado $w_{fd} = -0,01$.

4.4.5 Controle de Velocidade

Nesta subseção, são apresentadas as funções de recompensa para controlar as velocidades dos personagens Humano 3D e Dálmata 2D. A função de recompensa usada para controlar a velocidade do personagem Humano 3D é definida como

$$r = w_v T'_{vel} + w_o T_{ori} + w_h T_h + w_s T_{speed}, \tag{4.13}$$

onde $T'_{vel} = \vec{v}_r{}' \cdot \vec{d}_t$ e

$$\vec{v}_r{}' = \begin{cases} s_{des} \frac{\vec{v}_r}{||\vec{v}_r||}, & \text{se} ||\vec{v}_r|| > s_{des}; \\ \vec{v}_r, & \text{caso contrário.} \end{cases} \tag{4.14}$$

Assim, os esforços adicionais do personagem para superar a velocidade desejada não são mais recompensados, e a velocidade do personagem acaba ficando perto da s_{des}. O novo termo

$$T_{speed} = (s_{des} - ||\vec{v}_r||)^2 \tag{4.15}$$

penaliza o quadrado da diferença entre a velocidade desejada escolhida e o tamanho do vetor velocidade do corpo raiz, o que melhora a precisão do controle de velocidade. Foi usado um peso negativo $w_s = -0,01$.

A função de recompensa usada para controlar a velocidade do personagem Dálmata 2D é definida como

$$r = w_s T_{speed} + w_h T_{head} + w_{v_y} T_{vel_y}, \tag{4.16}$$

onde

$$T_{speed} = (s_{des} - v_{rx})^2 \tag{4.17}$$

é usado para penalizar o quadrado da diferença entre a velocidade desejada escolhida e a velocidade horizontal do corpo raiz. Foi usado um peso negativo $w_s = -0,01$.

4.5 Entrada e Saída da Rede

Além do controle fornecido através da definição das recompensas, este trabalho também tem como objetivo explorar outras duas formas de controle: o controle em tempo real sobre a velocidade do personagem e o controle de simetria do movimento gerado pelo personagem. Essas formas de controle são proporcionadas através de ajustes realizados, respectivamente, na entrada e na saída da rede.

4.5.1 Controle em Tempo Real

Na direção de fornecer ao animador um maior controle sobre a animação resultante, este trabalho propõe que características que se deseja controlar nos movimentos, como a velocidade da locomoção, possam fazer parte da entrada da rede neural, permitindo que elas sejam modificadas pelo usuário em tempo real, durante a simulação física controlada pela rede já treinada.

Na Figura 9, a velocidade desejada, V_d, ganha destaque como uma das entradas da rede neural, justamente por ser a informação que é adicionada como uma nova entrada para tornar possível o seu controle em tempo real. Assim como as outras entradas, que correspondem ao estado do personagem, determinam a escolha da melhor ação a ser tomada pela rede, incluir a velocidade desejada como uma das entradas da rede significa que ela também passa a influenciar na ação tomada pela rede em cada instante da simulação. A diferença é que, enquanto as outras entradas são capturadas automaticamente da própria simulação física, a velocidade desejada é fornecida diretamente pelo animador, que é permitido alterar essa nova entrada a seu critério, como uma forma de controle sobre a velocidade que ele quer que o personagem alcance.

Ou seja, enquanto, para uma determinada velocidade desejada escolhida, a rede tem um conjunto específico de ações para os vários possíveis estados do personagem, uma vez que a velocidade desejada muda, um novo conjunto de ações diferentes passam a ser acessadas a partir dos mesmos estados, correspondendo a uma mudança de comportamento da rede, associada à nova velocidade desejada. De fato, a uma baixa velocidade, o personagem deve apresentar um movimento bem diferente do movimento que ele teria a uma alta velocidade, ao ponto de, por exemplo, alternar entre um movimento de caminhada e um movimento de corrida.

Mas, para que a rede esteja preparada para essa adaptação em tempo real, durante o treinamento essa entrada nova também precisa sofrer mudanças parecidas com as que possam ser escolhidas pelo animador após o treinamento. Como a velocidade desejada não é capturada automaticamente, é preciso definir um procedimento de atualização dessa informação durante o treinamento. Portanto, no início de cada episódio, uma velocidade diferente é sorteada e mantida fixa durante todo esse episódio, simulando o controle do usuário, para permitir que o personagem se adapte aos diferentes valores a serem usados após a rede treinada. Uma outra alternativa é variar gradativamente a velocidade desejada dentro de um mesmo episódio.

Além de parte das entradas, a velocidade desejada também influencia o processo de aprendizado como parte da própria função de recompensa. No caso da velocidade desejada, o controle já tinha sido pensado na forma de um termo da função de recompensa. Para usá-la como característica a ser controlada em tempo real, o mesmo termo explorado na Subseção 4.4.5, responsável por penalizar a diferença entre a velocidade atual e a velocidade desejada, foi aproveitado.

4.5.2 Controle de Simetria

Os movimentos de humanos, e de animais em geral, muitas vezes possuem natureza simétrica, o que indica que explorar essa simetria durante o treinamento pode facilitar o aprendizado de movimentos mais realistas. Ou seja, ao explorar movimentos simétricos, pode-se aumentar a velocidade do aprendizado, obtendo soluções mais eficientes e esteticamente mais agradáveis (ABDOLHOSSEINI *et al.*, 2019).

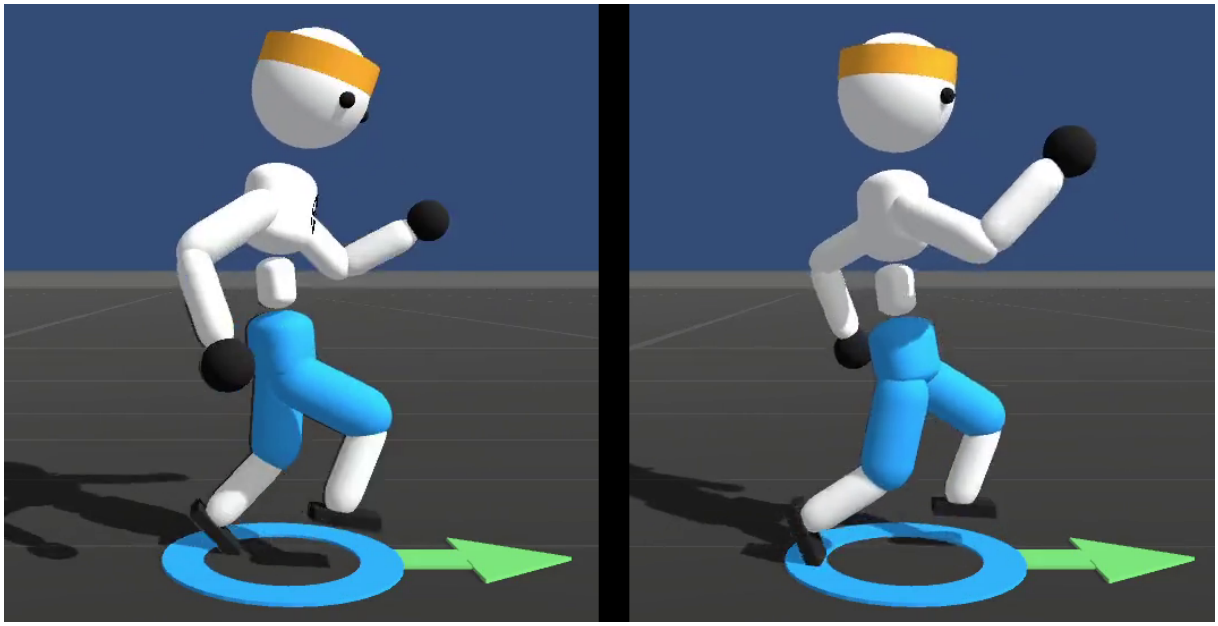
Alguns trabalhos investem em definir representações reduzidas de controle para facilitar o trabalho de otimização. Por exemplo, Nunes *et al.* (2012) usam os modos de vibração naturais do personagem para animá-los e Ranganath *et al.* (2019) usam coativações obtidas a partir de uma base de dados de movimentos capturados. Isso ajuda a diminuir a complexidade

da rede, que passa a lidar com uma saída menor do que se fossem usados todos os graus de liberdade do personagem.

Neste trabalho, utilizou-se uma representação reduzida para os movimentos dos membros do Humano 3D, alterando a saída da rede neural para que os movimentos dos braços e os das pernas fossem espelhados. A abordagem escolhida foi aplicar a simetria na arquitetura da rede neural, onde o vetor de ações do personagem foi reduzido, restringindo a saída da rede neural de tal maneira que as ações de um lado do personagem possam ser aproveitadas e automaticamente aplicadas ao seu outro lado. As partes do tronco e da cabeça não são reduzidas porque a simetria não se aplica. Essa abordagem requer conhecimento sobre as estruturas de simetria do movimento e da morfologia do personagem para que essas restrições possam ser aplicadas.

Na Figura 12 pode-se ver a simetria aplicada ao movimento de corrida do personagem Humano 3D, em que os movimentos da perna esquerda são aproveitados automaticamente para definir os movimentos da perna direita, mas de maneira espelhada. O mesmo acontece com os braços. Além disso, os movimentos dos braços também estão atrelados aos movimentos das pernas. Para aplicar a simetria ao movimento, a saída da rede neural é reduzida de 39 para 24 valores.

Figura 12 – Simetria aplicada a corrida do personagem Humano 3D.



Fonte: elaborado pelo autor (2022).

5 RESULTADOS

Este capítulo apresenta os resultados obtidos: explorando as funções de recompensa propostas na Seção 4.4, para gerar diferentes tipos de locomoção; explorando os ajustes na entrada da rede propostos na Seção 4.5.1, para permitir o controle em tempo real da velocidade; e explorando os ajustes na saída da rede propostos na Seção 4.5.2, para aplicar simetria ao movimento. Uma breve discussão do ambiente utilizado e do contexto da realização dos experimentos é mostrada inicialmente.

5.1 Ambiente de execução dos experimentos

Os experimentos apresentados neste trabalho foram todos realizados usando a versão 2019.3.3f1 Personal da Unity 3D como ambiente de simulação e o ml-agents na versão 3 como plataforma para executar o algoritmo de aprendizado por reforço. Embora a versão 3 do ml-agents ofereça suporte a GPU, a documentação (UNITY, 2020a) recomenda o treinamento usando apenas CPU, pois até essa versão, apresentava um desempenho melhor.

Os experimentos foram realizados tanto em um notebook próprio quanto remotamente usando o ambiente do Google Colab[1]. O Google Colab corresponde ao ambiente de aprendizado de máquina do Google, hospedado em máquinas virtuais Linux de configuração variável, dependendo da demanda. O notebook usado possui as seguintes especificações:

- Sistema Operacional: Windows 10 Home 64-bit
- CPU: Intel Core i7-8565U 1.80 GHz, com 4 núcleos e 8MB de cache
- GPU: GeForce MX130 2GB GDDR5
- Memória: 8 GB DDR

A duração dos treinamentos para o Humano 3D foi de cerca de 10 horas no notebook e cerca de 16 horas no Google Colab; os treinamentos do Dálmata 2D levaram aproximadamente 8 horas no notebook e 12 horas no Google Colab. Para os dois personagens, o algoritmo *Proximal Policy Optimization* (PPO) foi utilizado com 20 milhões de passos, cada passo correspondendo ao par observação coletada e ação executada. Os hiperparâmentros utilizados na criação das redes neurais e configuração do algoritmo usado nos experimentos podem ser encontrados no Anexo A, junto com uma breve explicação sobre a função e a influência deles no treinamento da rede neural.

[1] https://colab.research.google.com/

Os detalhes dos vários experimentos realizados neste trabalho são apresentados na sequência e podem ser visualizados no vídeo complementar [2].

5.2 Humano 3D

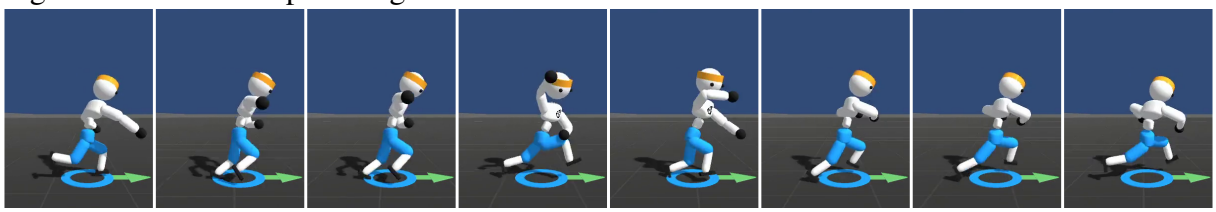
Nesta seção, é apresentada uma série de experimentos usando o personagem Humano 3D. Em cada experimento, é discutido como a função de recompensa foi usada para controlar o movimento desejado e são analisados os resultados obtidos.

5.2.1 Corrida bípede

O primeiro experimento é baseado em um dos experimentos disponíveis em Juliani *et al.* (2020), chamado "Walker", que serviu de ponto de partida para os testes realizados.

A função de recompensa da Equação 4.1 é usada. Se qualquer uma das partes do corpo do personagem, exceto os pés, tocar o chão, um novo episódio de treinamento começa. A Figura 13 mostra o controlador resultante, após o treinamento, atuando sobre o Humano 3D em tempo real. O personagem atinge a velocidade máxima de cerca de 20 m/s, e uma velocidade média de 17 m/s.

Figura 13 – Corrida bípede original.



Fonte: elaborado pelo autor (2022).

Observa-se que o aprendizado por reforço pode apresentar algumas inconsistências no contexto de problemas de controle contínuo de alta dimensão, como controlar personagens com base na física. Após o treinamento, desde que as condições iniciais da simulação sejam mantidas e não haja interações do usuário, a rede neural treinada se comporta de forma determinística. No entanto, devido à natureza estocástica do processo de treinamento, mesmo um treinamento replicado usando exatamente os mesmos parâmetros pode alcançar resultados muito diferentes. Escolher as funções de recompensa adequadas para controlar o tipo de movimento desejado se tornaria impraticável com essas inconsistências. Como em Peng *et al.* (2018), verifica-se também

[2] Video Complementar: https://sites.google.com/view/dissertaoantssousa

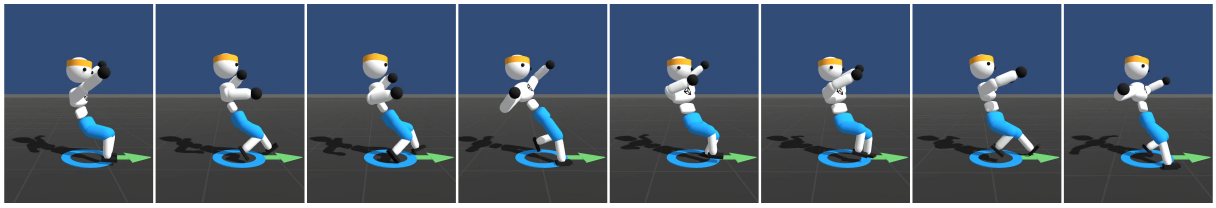
que as configurações iniciais do personagem em cada novo episódio têm influência crítica sobre o resultado do treinamento. No experimento original “Walker”, uma rotação aleatória do eixo y é sorteada usando o intervalo completo (0º a 360º), no início de cada episódio. Essa liberdade para a rotação do eixo y muitas vezes resultou numa corrida de costas após o treinamento para o primeiro experimento. Limitar a rotação aleatória do eixo y ao intervalo de -45º a 45º apresentou uma consistência satisfatória nos resultados.

Os próximos experimentos mostram como foi fornecido ao animador um melhor controle sobre o tipo de movimento desejado, fazendo pequenas alterações na função de recompensa de forma a torná-la um mecanismo de controle mais geral.

5.2.2 Corrida bípede de costas

Com o intuito de testar outros tipos de locomoção, foi planejada uma função de recompensa que resultasse numa corrida de costas, porém, agora propositalmente, conforme ilustrado na Figura 14. Um pequeno ajuste na Equação 4.1, substituindo o peso w_v por -0,02, que originalmente tinha o valor 0,02, já foi suficiente para se atingir o objetivo. Dessa forma, é recompensado o alinhamento da direção do vetor velocidade do corpo raiz, $\vec{v}_r$, com o vetor no sentido oposto ao $\vec{d}_t$ (vetor verde), incentivando o personagem a correr de costas.

Figura 14 – Corrida bípede de costas.



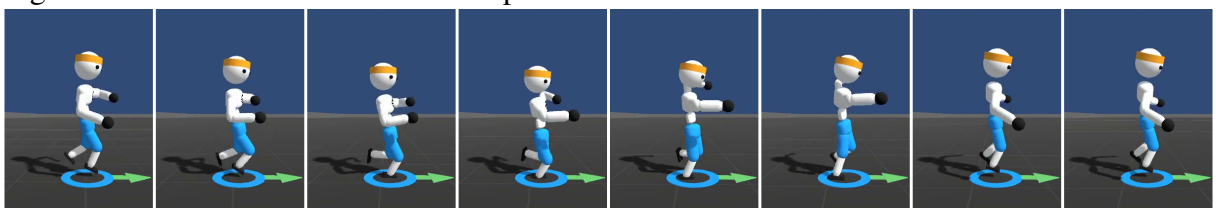
Fonte: elaborado pelo autor (2022).

5.2.3 Corrida Saltitando em Uma Perna Só

Para alcançar outros estilos de locomoção, foi testada a capacidade do personagem de se mover para frente usando apenas uma perna. Novamente, apenas um pequeno ajuste na função de recompensa foi suficiente para atingir o novo objetivo. Usando a função de recompensa da Equação 4.3, que faz com que o personagem salte usando apenas a perna direita, mantendo a perna esquerda levantada, a animação resultante é muito próxima de uma locomoção humana saltitante. Observa-se que os movimentos dos braços, que auxiliam no equilíbrio do personagem,

surgem naturalmente do resultado do treinamento (Figura 15). A influência nessa parte superior não foi tão perceptível no Experimento 5.2.1 (corrida bípede), por exemplo.

Figura 15 – Corrida saltitando em uma perna só.



Fonte: elaborado pelo autor (2022).

5.2.4 Pulo

Além de realizar locomoções, também foi testada a capacidade do personagem de realizar outros tipos de movimento, como por exemplo, pular em um mesmo local. Como o deslocamento horizontal não é compatível com o movimento desejado, e como não há necessidade de restringir a direção em que o personagem olha durante o pulo, foi mantido T_h na função de recompensa, utilizando o mesmo peso $w_h = 0.01$, e acrescentado um termo extra, T_j, para encorajar o personagem a pular. Três expressões diferentes foram utilizadas para representar esse novo termo, mas mantendo o peso associado $w_j = 0.02$.

Primeiro, foi usado $T_j = h_p$ para recompensar a altura do corpo raiz (pélvis) em relação ao chão, porém o personagem ficou parado. Esse comportamento pode ser explicado pelo fato de que a pose inicial do personagem já permite que o personagem mantenha facilmente uma boa altura para a pélvis, simplesmente mantendo o equilíbrio quando seus pés estão tocando o chão. Em situações como essa, o aprendizado por reforço tende a manter o estado que possibilita a maior recompensa encontrada, pois, a exploração de outros caminhos poderia causar uma diminuição da recompensa. Então, agachar para conseguir impulso para depois obter uma melhor elevação com o pulo, para compensar a perda de altura durante o agachamento, não se apresentou um caminho satisfatório.

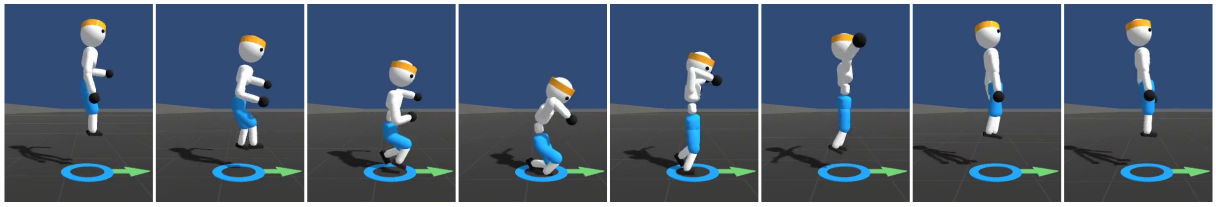
Segundo, foi usado $T_j = v_{p_y}$ para recompensar a velocidade vertical da pélvis, mas, novamente, sem sucesso. O resultado é semelhante ao primeiro, pois a velocidade que o personagem ganha no movimento ascendente do pulo é perdida no movimento descendente. Então, é melhor ficar parado.

Terceiro, foi utilizado a função de recompensa da Equação 4.7, que foi suficiente para atingir o objetivo desejado (Figura 16). A nova função de recompensa é capaz de recom-

pensar qualquer deslocamento vertical, incentivando o movimento de agachamento necessário. Novamente, um movimento de braços diferente surge naturalmente. O personagem visivelmente joga seus braços para baixo para impulsionar sua pélvis ainda mais para cima.

As três tentativas diferentes de representar T_j mostram que pequenos ajustes podem de fato fazer a diferença no sucesso do experimento.

Figura 16 – Pulo.

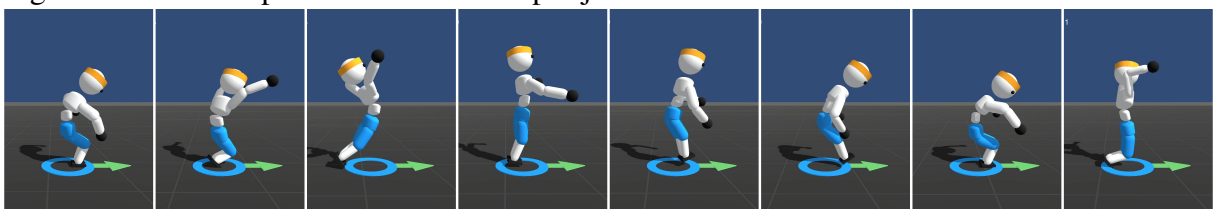


Fonte: elaborado pelo autor (2022).

5.2.5 Corrida Pulando com os Dois Pés Juntos

Aproveitando o sucesso do novo termo do pulo, T_j, foi testado o movimento de saltar no estilo canguru, avançando com as duas pernas juntas. Usando a função de recompensa da Equação 4.11, a Figura 17 mostra que a nova função de recompensa é capaz de fazer o personagem saltar com sucesso, apresentando uma forma muito interessante de locomoção, semelhante ao movimento de competidores numa corrida de saco. Novamente, o movimento dos braços, que ajuda o personagem tanto a avançar quanto a manter o equilíbrio, surge naturalmente.

Figura 17 – Corrida pulando com os dois pés juntos.



Fonte: elaborado pelo autor (2022).

5.2.6 Equilíbrio

Motivado pelas duas tentativas sem sucesso do experimento do pulo, em que o personagem ficou em pé parado, foi realizado um experimento cujo único objetivo era manter o equilíbrio do personagem, agora de maneira proposital.

Primeiro, foi usado simplesmente o termo original da altura, T_h, da Equação 4.1,

com o mesmo peso $w_h = 0.01$. O personagem conseguiu manter o equilíbrio, mas apresentou oscilações intensas e indesejadas nos braços. Depois, combinou-se T_h com o termo adicional da Equação 4.15, usando $s_{des} = 0$, a fim de penalizar o quadrado do comprimento do vetor velocidade da pélvis. A nova recompensa $w_h T_h + w_b T_{vel}$ diminui essas oscilações indesejadas, ainda que não atue explicitamente nos braços. Novamente, o movimento dos braços surge naturalmente.

5.2.7 Corrida Bípede com Controle de Velocidade

Foram realizadas quatro sessões de treinamento independentes para cada uma das seguintes velocidades desejadas: 1, 3, 5 e 10 m/s, usando a Equação 4.13. A nova recompensa apresenta uma boa precisão para baixas velocidades desejadas, mas para velocidades mais altas, como 12 m/s, a função básica da corrida fica comprometida e o personagem nem consegue correr. Isso provavelmente acontece porque, para tentar corrigir o grande erro inicial do termo T_{speed}, já que o personagem inicia cada episódio em repouso, o personagem projeta sua pélvis para frente e isso acaba tendo uma prioridade maior do que focar no planejamento para melhorar gradativamente as passadas para a corrida. Essa melhora gradual é fornecida pelo termo T_{vel}, por exemplo. Definir uma velocidade inicial aleatória para o personagem, no início de cada episódio, pode ajudar no treinamento.

5.2.8 Corrida Bípede com Controle de Velocidade em Tempo Real

Um dos resultados mais interessantes e promissores deste trabalho é permitir o controle da velocidade em tempo real, durante a simulação com a rede já treinada. Como proposto na Seção 4.5.1 e conforme ilustrado no esquema de controle da Figura 9, para que essa escolha deliberada e interativa da velocidade desejada tenha influência sobre a simulação em tempo real, a velocidade desejada deve fazer parte da entrada da rede e o personagem precisa se deparar com diferentes valores para ela durante o treinamento. Uma velocidade desejada aleatória é, portanto, sorteada no início de cada episódio, considerando uma faixa de valores de 0 m/s a 20 m/s. A função de recompensa é a mesma utilizada no Experimento 5.2.7 (corrida bípede com controle de velocidade).

Dentro da faixa de velocidade de 1 m/s a 12 m/s, o controle de velocidade desejada em tempo real mostrou acurácia satisfatória, mas menor do que ao definir uma única velocidade desejada ao longo do treinamento. Acima de 12 m/s, como no experimento anterior, o personagem

apresenta locomoção instável.

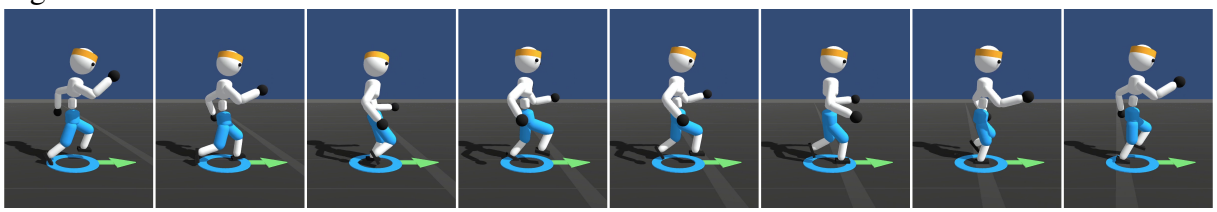
Uma das vantagens que chama a atenção nesse experimento é a influência no estilo de locomoção do personagem, que pode ser percebida em tempo real na medida em que a velocidade desejada é modificada pelo animador. Embora mudanças essenciais de locomoção ainda não tenham ocorrido, como transições entre caminhar e correr, que poderiam surgir naturalmente de acordo com a mudança da velocidade desejada, acredita-se que esse tipo de controle possa ser alcançado incorporando termos de recompensa que melhor explorem o gasto energético do personagem (YU *et al.*, 2018). Melhorias nesse sentido serão investigadas no futuro.

5.2.9 Corrida Bípede com Controle de Simetria

Como proposto na Seção 4.5.2, neste experimento, a saída da rede neural é reduzida para forçar um movimento simétrico dos braços e das pernas do personagem Humano 3D. A função de recompensa utilizada foi a mostrada na Equação 4.1, a mesma usada no experimento da Seção 5.2.1. Além de obter as poses de um braço de forma espelhada à do outro e as poses de uma perna de forma espelhada à da outra, a sincronia entre braços e pernas foi também forçada ao utilizar a mesma saída da rede, que controla o quanto as pernas vão para frente ou para trás, para controlar também o quanto os braços vão para frente ou para trás. No caso, a mesma saída da rede que indica que a perna esquerda deve rotacionar para frente, por exemplo, automaticamente também já é aproveitada para definir que o braço direito também deve rotacionar para frente. Consequentemente, essa mesma saída também já define que tanto a perna direita quanto o braço esquerdo devem rotacionar para trás, como pode ser percebido no segundo quadro da Figura 18.

O resultado após o treinamento, mostrado na mesma Figura 18, se assemelha aos movimentos naturais de corrida de um ser humano, com alternância entre as pernas e os braços. Esses resultados foram gerados sem usar movimentos de referência. Sendo assim, o animador possui uma alternativa útil para criar controladores de locomoção para personagens, simplesmente explorando esses aspectos de simetria dos movimentos, mesmo que não haja dados de movimento capturado disponíveis. Embora o personagem demonstre uma locomoção mais natural em comparação com alguns trabalhos existentes em DRL (HEESS *et al.*, 2017; SOUSA *et al.*, 2022), a qualidade do movimento ainda não está no mesmo nível de trabalhos anteriores em animação por computador que exploram dados de captura de movimentos do mundo real (PENG *et al.*, 2018; BERGAMIN *et al.*, 2019; RANGANATH *et al.*, 2019).

Figura 18 – Corrida com simetria.



Fonte: elaborado pelo autor (2022).

5.3 Dálmata 2D

Com o objetivo de testar a generalização do método, um personagem com morfologia diferente também foi usado. Nesta subseção são apresentados os estudos de casos para o Dálmata 2D, descrevendo as funções de recompensa escolhidas e os resultados obtidos.

5.3.1 Corrida

O primeiro experimento visa incentivar a locomoção para frente da melhor forma que o personagem for capaz. Os termos da função de recompensa foram adaptados considerando o novo contexto bidimensional e a nova morfologia do personagem, que naturalmente resulta em um estilo de locomoção diferente do personagem bípede. A função de recompensa da Equação 4.2 é usada.

O personagem atinge picos de velocidade de 20 m/s, e velocidade média de 16 m/s. Um estilo de locomoção completamente novo emerge de acordo com as características geométricas e físicas do novo personagem. Pode-se ver todo o corpo do Dálmata 2D trabalhando em conjunto para alcançar o movimento de galope.

5.3.2 Corrida de costas

Semelhante ao caso do Humano 3D, apenas um pequeno ajuste no peso w_{v_x} do termo T_{velx} da função de recompensa 4.2 foi suficiente para atingir a corrida de costas (Figura 19). O valor anterior de 0,03 foi substituído por -0,03, passando a premiar a velocidade negativa do tronco no eixo x. Modificações de recompensas semelhantes produziram mudanças de comportamento semelhantes para ambos os personagens, embora em estilos de locomoção diferentes.

Figura 19 – Corrida de costas (Dálmata 2D).

Fonte: elaborado pelo autor (2022).

5.3.3 Corrida com controle de velocidade

Semelhante ao Humano 3D, o controle da velocidade corresponde a penalizar o quadrado da diferença entre a velocidade desejada escolhida e a velocidade horizontal do corpo raiz. Esse controle é considerado na função de recompensa 4.16, usada neste experimento.

Foram realizados três treinamentos independentes, cada um com uma velocidade desejada (5, 10, -10 m/s). O personagem conseguiu atingir a velocidade desejada de 10 m/s com uma boa precisão (Figura 20). Para 5 m/s, o personagem obteve uma boa precisão, embora não tenha apresentado um comportamento visualmente atraente usando o corpo inteiro. Um possível motivo pode ser a limitação da morfologia, já que essa versão 2D não possui as quatro patas necessárias para andar. Seria interessante experimentar um novo personagem quadrúpede no futuro.

Figura 20 – Dálmata galopando com uma velocidade desejada de 10 m/s.

Fonte: elaborado pelo autor (2022).

5.3.4 Corrida com controle de velocidade em tempo real

O controle da velocidade em tempo real é implementado de maneira bem similar ao usado com o Humano 3D. Semelhante ao experimento da corrida bípede com controle de velocidade em tempo real (5.2.8), a escolha de velocidades variando dentro do intervalo entre 1 m/s e 12 m/s mostrou um resultado satisfatório. Comportamentos semelhantes de aceleração e frenagem, conforme a velocidade desejada é alterada, são apresentados para os dois personagens. Isso indica que generalizar o controle de velocidade em tempo real proposto para personagens arbitrários é uma opção promissora.

5.3.5 Corrida saltitando em uma perna só

Para o movimento de corrida em uma perna só para o Dálmata 2D, também foram mantidos os três termos da Equação 4.2, e considerado um termo condicional que penaliza

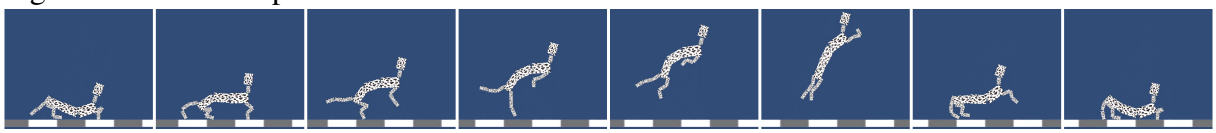
quando a pata dianteira toca o chão e um termo adicional para incentivar que a pata dianteira ganhe altura. A função de recompensa da Equação 4.5 é usada.

A nova recompensa faz o Dálmata 2D pular usando sua perna traseira, sua cauda e sua cabeça, mantendo a perna dianteira levantada. Provavelmente devido às suas características morfológicas, este exemplo foi mais difícil do que com o Humano 3D e não resultou na locomoção esperada. Porém, um comportamento interessante usando sua cauda e sua cabeça para ajudar no equilíbrio surgiu espontaneamente.

5.3.6 Pulo

O experimento do pulo também foi reproduzido para o Dálmata 2D. Além do termo do salto para recompensar o valor absoluto da velocidade vertical do corpo raiz, que funcionou bem para o Humano 3D, também foram usados alguns termos dos experimentos 2D anteriores. Os termos usados que fizeram sentido para pular foram recompensar a altura da pata dianteira, penalizar se ela tocar o chão e evitar o movimento excessivo da cabeça em relação ao torso. A função de recompensa 4.9 foi usada neste experimento. O resultado obtido é muito parecido com um cachorro real saltando para alcançar alguma coisa com a pata dianteira (Figura 21).

Figura 21 – Dálmata pulando.



Fonte: elaborado pelo autor (2022).

Antes dessa tentativa bem-sucedida, um comportamento inesperado e muito interessante também foi obtido. Numa tentativa de minimizar o tempo do personagem no chão, associando que mais tempo no ar resultaria em saltos maiores, em vez de recompensar o módulo da velocidade vertical, foi usado um termo que penalizava quando as patas do personagem tocavam o chão. Como resultado, a Figura 22 mostra o surgimento de um pulo de cabeça para baixo, usando a cabeça e o rabo.

Na medida em que é dada mais liberdade ao personagem, removendo a restrição que recompensa se o personagem estiver de cabeça para cima, por exemplo, o processo de aprendizagem pode fornecer novos comportamentos surpreendentes, comprovando que a emergência natural do movimento, que é a essência do DRL, ainda está sendo explorada em nossa abordagem, permitindo que surjam tais soluções inesperadas, mas interessantes e curiosas.

Figura 22 – Dálmata pulando de cabeça para baixo.

Fonte: elaborado pelo autor (2022).

5.3.7 Resumo dos resultados obtidos

De acordo com os resultados obtidos de **QP1** e **QP3**, a definição da função de recompensa é um desafio em ambientes fisicamente simulados para personagens articulados devido à complexidade das interações físicas e das interações entre os personagens e o ambiente. A função de recompensa deve ser criada de forma a incentivar o agente a tomar ações que resultem em movimentos naturais e realistas, mas também deve ser capaz de lidar com a incerteza e as variações que ocorrem como resultado das interações físicas. Apesar dos vários experimentos realizados com diferentes tipos de movimentos para os personagens (Experimentos 5.2.1, 5.3.1, 5.2.6) muitas vezes as funções de recompensas foram definidas através de tentativa e erro. A principal dificuldade é definir a função de recompensa de forma precisa e completa, algumas vezes resultando em movimentos imprecisos ou não desejados, como visto nos Experimentos 5.2.4 e 5.3.6.

Ao analisar a **QP2**, o animador pode esperar que o controle varie de acordo com a complexidade e precisão da função de recompensa definida. Quanto mais simples for a função menos controle se obtém sobre o movimento desejado (Experimentos 5.2.1, 5.3.1), enquanto uma função de recompensa mais complexa permite um controle mais preciso sobre as ações do personagem (Experimento 5.2.5). Vale ressaltar que a eficácia da função de recompensa também pode ser afetada pelos dados de entrada e pela precisão do modelo. Portanto, um animador que tenha um bom entendimento da função de recompensa e da simulação pode atingir o nível desejado de controle.

Tratando a **QP4**, o controle do movimento pode ser explorado em tempo real, durante a execução da rede já treinada, através da manipulação dos parâmetros de entrada da rede. Neste trabalho, o controle foi exercido através da velocidade desejada, onde, durante o treinamento, o personagem tinha diferentes velocidades como entrada para a rede neural e sua função de recompensa se baseava em se aproximar de cada uma dessas velocidades. Depois do processo de treinamento, durante a execução da simulação, é possível trocar os valores de entrada da velocidade e, com isso, o personagem altera sua velocidade em tempo real de acordo com a entrada fornecida, isso pode ser observado nos Experimentos 5.2.8 e 5.3.4.

Observando a **QP5**, a incorporação de controle de simetria no aprendizado de movimento de personagens fisicamente simulados se mostrou eficaz na obtenção de resultados melhores, como foi evidenciado no Experimento 5.2.9. A utilização de simetria como uma forma de regularização pode ajudar a produzir movimentos mais próximos aos movimentos humanos reais, como alternância dos pés e braços. Além disso, o controle de simetria pode ser usado para restringir a variedade de movimentos gerados, o que pode levar a movimentos mais naturais e esteticamente agradáveis e coerentes com os movimentos reais.

5.4 Adaptabilidade dos Controladores

Como uma das contribuições deste trabalho é avaliar o quanto a função de recompensa, por si só, é capaz de influenciar o processo de aprendizado do personagem no contexto de DRL, evitou-se que o ambiente tivesse uma influência direta nos experimentos. Assim, um ambiente básico, com piso plano uniforme, foi escolhido para ser usado nos testes. No entanto, para mostrar como os controladores gerados podem lidar com interações com o ambiente, a fim de avaliar a generalização da abordagem proposta para novas situações, nesta seção, os mesmos controladores obtidos para a Corrida Bípede (5.2.1) e para o Pulo (5.2.4) são testados em duas situações bem desafiadoras. As duas situações tratam sobre como lidar com instabilidades, causadas tanto por arremessos de bolas no personagem quanto por alterações no terreno, deixando-o cada vez mais irregular. A interface do usuário permite escolher uma posição inicial para as bolas e lançá-las em uma parte específica do personagem, em tempo real. É importante destacar que não foram realizados novos treinamentos. Estes novos experimentos usam exatamente os mesmos controladores, já treinados sem o lançamento de bolas e sem o uso de terrenos irregulares.

Como pode ser visto no vídeo complementar (Seção 5.1), após o personagem receber os impactos das bolas, ele ainda é capaz de recuperar seu movimento aprendido. Na verdade, essa habilidade está intimamente relacionada a uma característica essencial do DRL, de não se restringir às condições avaliadas durante a fase de treinamento. Mesmo para situações não vistas antes durante o treinamento, o DRL tende a generalizar sua saída e a fornecer uma ação potencialmente adequada com base no espaço de busca coberto. A abordagem de simulação baseada em física, combinada com DRL, também é essencial para tornar essa recuperação possível.

Ao contrário, por exemplo, da abordagem conhecida como space-time constraints

(WAMPLER; POPOVIC, 2009; WAMPLER *et al.*, 2014), a trajetória dos personagens é obtida de forma indireta, a partir das ações determinadas pela rede neural. Como a abordagem space-time constraints inclui diretamente a própria trajetória como parte dos parâmetros de otimização a serem encontrados, qualquer modificação na trajetória resultante, como as causadas pelos impactos, exigiria um novo processo de otimização prévio. Assim, a interação em tempo real mostrada com as bolas não é permitida. Além disso, a abordagem space-time constraints usa as equações de movimento do personagem e do ambiente como restrições do problema de otimização. Portanto, ela tem que lidar com um espaço de busca contendo trajetórias que não obedecem às equações de movimento e com o complicado problema de especificar restrições de contato com o chão como parte da otimização. Neste trabalho, o processo de aprendizagem ocorre em um espaço de busca que contém apenas trajetórias fisicamente válidas. Além disso, os contatos surgem espontaneamente da simulação física e, portanto, não precisam ser explicitamente abordados no processo de aprendizado. Em situações mais complexas, a técnica usando DRL ainda pode exigir um novo treinamento, mas não se compara com a exigência de uma nova otimização já implícita ao usar a abordagem space-time constraints.

Uma limitação atual do DRL é que os resultados apresentam alguns movimentos esquisitos, no personagem como um todo ou em alguns de seus membros. Embora movimentos de referência (PENG *et al.*, 2018; LEE; POPOVIC, 2010; WAMPLER *et al.*, 2014) possam atuar como um mecanismo de regularização que melhora a qualidade do resultado, eles impõem um esforço extra de lidar com um banco de dados de movimento. Além disso, ao usar movimentos de referência, é preciso encontrar um bom equilíbrio entre ainda se beneficiar do surgimento espontâneo de novos comportamentos durante o treinamento e se beneficiar da capacidade de definir restrições explícitas de pose.

6 CONCLUSÕES E TRABALHOS FUTUROS

O controle de personagens articulados fisicamente simulados é um desafio devido à relação não intuitiva entre os parâmetros do controlador e o movimento gerado. Utilizando aprendizado por reforço profundo para esse controle, é possível definir uma função de recompensa que represente matematicamente o movimento desejado. Neste trabalho, foi apresentado como usar as funções de recompensa no DRL para obter diversos tipos de movimento para personagens articulados fisicamente simulados. Através de uma série de experimentos, foi apresentado que o DRL é uma abordagem promissora para resolver o problema de controle de tais personagens. A combinação de DRL com dados de captura de movimento ou técnicas de imitação de movimento adiciona mais realismo às animações, mas esses dados nem sempre estão disponíveis. Além disso, o uso de movimentos de referência pode sacrificar o surgimento do comportamento natural, que é a essência do DRL. Dentro de um escopo em que nenhum movimento de referência foi usado, foi apresentado ao animador uma ferramenta que proporciona um maior grau de controle sobre o tipo do movimento resultante obtido, através de pequenas alterações na função de recompensa. Esse controle fornecido se mostrou capaz de generalizar para novos personagens e novas situações, ainda permitindo que comportamentos surjam de maneira espontânea.

Definir os termos de recompensa e seus respectivos pesos é o maior desafio para a obtenção dos movimentos desejados, muitas vezes exigindo várias tentativas de definição com a correspondente interpretação dos movimentos gerados. A utilização de dois personagens com morfologias diferentes permitiu visualizar que poucas adaptações da função de recompensa de um personagem para outro é suficiente para atingir comportamentos semelhantes. Isso sugere uma generalização promissora para personagens arbitrários. Além dos ajustes da função de recompensa na fase de pré-treinamento, também foi demonstrado o controle em tempo real da velocidade dos personagens depois da rede treinada. Usando essa versão em tempo real do controle também foi possível perceber atualizações no estilo de movimento na medida em que a velocidade era alterada, apresentando transições satisfatórias entre os movimentos nas diferentes velocidades. Além disso, foi usada uma representação simplificada para a saída da rede neural para que fossem gerados movimentos espelhados (simétricos) para o Humano 3D, possibilitando uma corrida mais realista, com alternância adequada de braços e pernas.

Como trabalhos futuros, pretende-se generalizar o ambiente para personagens articulados arbitrários (diferentes morfologias e criaturas imaginárias), criando uma interface de modelagem de personagens com um painel para permitir ao usuário definir facilmente a função

de recompensa. Pretende-se explorar *Curriculum Learning*, onde as tarefas de aprendizagem aumentam gradativamente a dificuldade na medida em que o personagem evolui em seu treinamento, conforme utilizado por (YU *et al.*, 2018). Isso pode aumentar a velocidade do treinamento e o aprendizado de movimentos mais complexos.

REFERÊNCIAS

ABDOLHOSSEINI, F.; LING, H. Y.; XIE, Z.; PENG, X. B.; PANNE, M. van de. On learning symmetric locomotion. In: **Proc. ACM SIGGRAPH Motion, Interaction, and Games (MIG 2019)**. New York, NY, USA: Association for Computing Machinery, 2019. p. 1–10. Disponível em: https://doi.org/10.1145/3359566.3360070. Acesso em: 01 maio 2022.

ABERMAN, K.; WU, R.; LISCHINSKI, D.; CHEN, B.; COHEN-OR, D. Learning character-agnostic motion for motion retargeting in 2d. **ACM Trans. Graph.**, Association for Computing Machinery, New York, NY, USA, v. 38, n. 4, jul 2019. ISSN 0730-0301. Disponível em: https://doi.org/10.1145/3306346.3322999. Acesso em: 10 jul. 2020.

ARULKUMARAN, K.; DEISENROTH, M. P.; BRUNDAGE, M.; BHARATH, A. A. A brief survey of deep reinforcement learning. **CoRR**, abs/1708.05866, 2017. Disponível em: http://arxiv.org/abs/1708.05866. Acesso em: 15 jan. 2020.

BAPST, V.; SANCHEZ-GONZALEZ, A.; DOERSCH, C.; STACHENFELD, K. L.; KOHLI, P.; BATTAGLIA, P. W.; HAMRICK, J. B. Structured agents for physical construction. **CoRR**, abs/1904.03177, 2019. Disponível em: http://arxiv.org/abs/1904.03177. Acesso em: 02 fev. 2020.

BERGAMIN, K.; CLAVET, S.; HOLDEN, D.; FORBES, J. R. Drecon: Data-driven responsive control of physics-based characters. **ACM Trans. Graph.**, Association for Computing Machinery, New York, NY, USA, v. 38, n. 6, nov 2019. ISSN 0730-0301. Disponível em: https://doi.org/10.1145/3355089.3356536. Acesso em: 20 jun. 2020.

BOOTH, J.; BOOTH, J. Marathon environments: Multi-agent continuous control benchmarks in a modern video game engine. **CoRR**, abs/1902.09097, 2019. Disponível em: http://arxiv.org/abs/1902.09097. Acesso em: 05 jun. 2020.

BOOTH, J.; IVANOV, V. Realistic physics based character controller. **CoRR**, abs/2006.07508, 2020. Disponível em: https://arxiv.org/abs/2006.07508. Acesso em: 03 jun. 2020.

CHENTANEZ, N.; MüLLER, M.; MACKLIN, M.; MAKOVIYCHUK, V.; JESCHKE, S. Physics-based motion capture imitation with deep reinforcement learning. In: **Proceedings of the 11th ACM SIGGRAPH Conference on Motion, Interaction and Games**. New York, NY, USA: Association for Computing Machinery, 2018. (MIG '18). ISBN 9781450360159. Disponível em: https://doi.org/10.1145/3274247.3274506. Acesso em: 03 abr. 2020.

CLAVET, S. Motion matching and the road to next-gen animation. In: **Proc. of Game Developers Conference (GDC)**. San Francisco, USA: [*S. n.*], 2016. v. 2016. Disponível em: https://www.gdcvault.com/play/1023280/Motion-Matching-and-The-Road. Acesso em: 20 abr 2020.

COROS, S.; KARPATHY, A.; JONES, B.; REVERET, L.; PANNE, M. van de. Locomotion skills for simulated quadrupeds. **ACM Trans. Graph.**, Association for Computing Machinery, New York, NY, USA, v. 30, n. 4, jul 2011. ISSN 0730-0301. Disponível em: https://doi.org/10.1145/2010324.1964954. Acesso em: 20 ago. 2020.

FRANÇA, Í. G. F. M. **Aprendizagem por reforço e por imitação com redes neurais aplicada ao jogo bomberman**. Dissertação (Mestrado) – Universidade Federal Fluminense - UFF, 2019. Disponível em: http://www.icaromotta.com/wp-content/uploads/2019/07/Dissertacao-de-Mestrado_versao-final.pdf. Acesso em: 02 fev. 2020.

GEIJTENBEEK, T. **Animating virtual characters using physics-based simulation**. Tese (Doutorado) – Utrecht University, 2013. Disponível em: https://dspace.library.uu.nl/handle/1874/289259. Acesso em: 07 out. 2020.

GEIJTENBEEK, T.; PRONOST, N. Interactive character animation using simulated physics: A state-of-the-art review. **Comput. Graph. Forum**, The Eurographs Association amp; John Wiley amp; Sons, Ltd., Chichester, GBR, v. 31, n. 8, p. 2492–2515, dec 2012. ISSN 0167-7055. Disponível em: https://doi.org/10.1111/j.1467-8659.2012.03189.x. Acesso em: 23 fev. 2020.

GÉRON, A. **Hands-on machine learning with Scikit-Learn and TensorFlow**: concepts, tools, and techniques to build intelligent systems. California, USA: "O'Reilly Media, Inc.", 2017.

HEESS, N.; TB, D.; SRIRAM, S.; LEMMON, J.; MEREL, J.; WAYNE, G.; TASSA, Y.; EREZ, T.; WANG, Z.; ESLAMI, S. M. A.; RIEDMILLER, M. A.; SILVER, D. Emergence of locomotion behaviours in rich environments. **CoRR**, abs/1707.02286, 2017. Disponível em: http://arxiv.org/abs/1707.02286. Acesso em: 23 jan. 2020.

JULIANI, A.; BERGES, V.; VCKAY, E.; GAO, Y.; HENRY, H.; MATTAR, M.; LANGE, D. Unity: A general platform for intelligent agents. **CoRR**, abs/1809.02627, 2020. Disponível em: http://arxiv.org/abs/1809.02627. Acesso em: 03 jan. 2020.

KRUPNIK, O.; MORDATCH, I.; TAMAR, A. Multi-agent reinforcement learning with multi-step generative models. **CoRR**, PMLR, v. 100, p. 776–790, 30 Oct–01 Nov 2020. Disponível em: http://proceedings.mlr.press/v100/krupnik20a.html. Acesso em: 20 jan. 2020.

KRY, P. G.; REVÉRET, L.; FAURE, F.; CANI, M.-P. Modal locomotion: Animating virtual characters with natural vibrations. In: WILEY ONLINE LIBRARY. **Computer Graphics Forum**. [*S. l.*], 2009. v. 28, n. 2, p. 289–298.

LAGULA, D.; KARLSSON, F. **Comparison of Two Different Methods of Generating Physics-Based Character Animation using Reinforcement Learning**. Monografia (Graduação) – Blekinge Institute of Technology, 2020. Disponível em: https://www.diva-portal.org/smash/get/diva2:1454725/FULLTEXT02. Acesso em: 15 fev. 2021.

LAI, M. Giraffe: Using deep reinforcement learning to play chess. **arXiv preprint arXiv:1509.01549**, 2015. Disponível em: https://arxiv.org/abs/1509.01549. Acesso em: 22 jun. 2020.

LANGFIELD, I. R. H. **Mocap**. 2013. Disponível em: http://designandmakeresearch.blogspot.com. Acesso em: 07 out. 2020.

LASZLO, J. **Controlling bipedal locomotion for computer animation**. Dissertação (Mestrado) – University of Toronto, 1997. Disponível em: http://www.dgp.toronto.edu/~jflaszlo/masc.thesis/thesis.html. Acesso em: 20 jun. 2020.

LEE, S. J.; POPOVIC, Z. Learning behavior styles with inverse reinforcement learning. In: **ACM SIGGRAPH 2010 Papers**. New York, NY, USA: Association for Computing Machinery, 2010. (SIGGRAPH '10). ISBN 9781450302104. Disponível em: https://doi.org/10.1145/1833349.1778859. Acesso em: 22 jun. 2020.

LIANG, J.; MAKOVIYCHUK, V.; HANDA, A.; CHENTANEZ, N.; MACKLIN, M.; FOX, D. Gpu-accelerated robotic simulation for distributed reinforcement learning. **arXiv preprint**

arXiv:1810.05762, 2018. Disponível em: http://arxiv.org/abs/1810.05762. Acesso em: 10 jul. 2020.

LIU, L.; HODGINS, J. Learning to schedule control fragments for physics-based characters using deep q-learning. **ACM Transactions on Graphics (TOG)**, ACM New York, NY, USA, v. 36, n. 3, p. 1–14, 2017. ISSN 0730-0301. Disponível em: https://doi.org/10.1145/3072959.3083723. Acesso em: 22 jan. 2020.

LIU, L.; HODGINS, J. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. **ACM Transactions on Graphics (TOG)**, ACM New York, NY, USA, v. 37, n. 4, p. 1–14, 2018. ISSN 0730-0301. Disponível em: https://doi.org/10.1145/3197517.3201315. Acesso em: 22 jul. 2020.

LIU, L.; PANNE, M. V. D.; YIN, K. Guided learning of control graphs for physics-based characters. **ACM Transactions on Graphics (TOG)**, ACM New York, NY, USA, v. 35, n. 3, p. 1–14, 2016. Disponível em: https://www.cs.ubc.ca/~van/papers/2016-TOG-controlGraphs/index.html. Acesso em: 20 jul. 2020.

LUO, Y.-S.; SOESENO, J. H.; CHEN, T. P.-C.; CHEN, W.-C. Carl: Controllable agent with reinforcement learning for quadruped locomotion. **arXiv preprint arXiv:2005.03288**, 2020. Disponível em: https://arxiv.org/abs/2005.03288. Acesso em: 20 jul. 2021.

MA, X. **Extending a Game Engine with Machine Learning and Artificial Intelligence**. 53 p. Dissertação (Mestrado) – Aalto University. School of Science, 2019. Disponível em: http://urn.fi/URN:NBN:fi:aalto-201906234003. Acesso em: 20 jul. 2020.

MACIEL, A.; HALIC, T.; LU, Z.; NEDEL, L. P.; DE, S. Using the PhysX engine for physics-based virtual surgery with force feedback. **The International Journal of Medical Robotics and Computer Assisted Surgery**, Wiley Online Library, v. 5, n. 3, p. 341–353, 2009. Disponível em: https://doi.org/10.1002/rcs.266. Acesso em: 20 ago. 2020.

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; GRAVES, A.; ANTONOGLOU, I.; WIERSTRA, D.; RIEDMILLER, M. Playing atari with deep reinforcement learning. **arXiv preprint arXiv:1312.5602**, 2013. Disponível em: http://arxiv.org/abs/1312.5602. Acesso em: 23 abr. 2020.

NOBLEGA, A. **Adaptive neural reinforcement learning-based agent for game balancing**. Dissertação (Mestrado) – Universidade Federal Fluminense - UFF, 2019. Disponível em: http://dx.doi.org/10.22409/PGC.2019.m.06416015730. Acesso em: 22 jul. 2020.

NOGUEIRA, Y. L. B. **Um modelo de sistema nervoso para o problema do controle de animação por dinâmica direta**. Dissertação (Mestrado) – Universidade Federal do Ceará - UFC, 2007. Disponível em: https://repositorio.ufc.br/handle/riufc/18656. Acesso em: 20 jul. 2020.

NUNES, R. F. **Usando vibrações naturais na descrição e no controle de locomoções fisicamente simuladas de personagens articuladas arbitrários**. Tese (Doutorado) – Universidade Federal do Ceará - UFC, 2012. Disponível em: http://www.repositorio.ufc.br/handle/riufc/18684. Acesso em: 10 jan. 2020.

NUNES, R. F.; CAVALCANTE-NETO, J. B.; VIDAL, C. A.; KRY, P. G.; ZORDAN, V. B. Using natural vibrations to guide control for locomotion. In: **Proceedings of**

the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. New York, NY, USA: Association for Computing Machinery, 2012. p. 87–94. Disponível em: https://doi.org/10.1145/2159616.2159631. Acesso em: 20 out. 2019.

PANNE, M. Van de; KIM, R.; FIUME, E. Virtual wind-up toys for animation. In: CITESEER. **Graphics Interface**. Alberta, CA, 1994. p. 208–208. Disponível em: https://doi.org/10.20380/GI1994.25. Acesso em: 20 jan. 2020.

PARK, S.; RYU, H.; LEE, S.; LEE, S.; LEE, J. Learning predict-and-simulate policies from unorganized human motion data. **ACM Transactions on Graphics (TOG)**, ACM New York, NY, USA, v. 38, n. 6, p. 1–11, 2019. Disponível em: https://doi.org/10.1145/3355089.3356501. Acesso em: 20 jul. 2020.

PENG, X. B.; ABBEEL, P.; LEVINE, S.; PANNE, M. van de. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. **ACM Transactions on Graphics (TOG)**, ACM New York, NY, USA, v. 37, n. 4, p. 1–14, 2018. Disponível em: https://xbpeng.github.io/projects/DeepMimic/index.html. Acesso em: 20 jun. 2020.

PENG, X. B.; BERSETH, G.; PANNE, M. Van de. Terrain-adaptive locomotion skills using deep reinforcement learning. **ACM Transactions on Graphics (TOG)**, ACM New York, NY, USA, v. 35, n. 4, p. 1–12, 2016. Disponível em: https://doi.org/10.1145/2897824.2925881. Acesso em: 20 jul. 2020.

PENG, X. B.; BERSETH, G.; YIN, K.; PANNE, M. V. D. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. **ACM Transactions on Graphics (TOG)**, ACM New York, NY, USA, v. 36, n. 4, p. 1–13, 2017. Disponível em: https://xbpeng.github.io/projects/DeepLoco/index.html. Acesso em: 219 jul. 2020.

PENG, X. B.; COUMANS, E.; ZHANG, T.; LEE, T.-W.; TAN, J.; LEVINE, S. Learning agile robotic locomotion skills by imitating animals. **arXiv preprint arXiv:2004.00784**, 2020. Disponível em: https://arxiv.org/abs/2004.00784. Acesso em: 01 maio 2022.

PENG, X. B.; KANAZAWA, A.; MALIK, J.; ABBEEL, P.; LEVINE, S. Sfv: Reinforcement learning of physical skills from videos. **ACM Transactions on Graphics (TOG)**, ACM New York, NY, USA, v. 37, n. 6, p. 1–14, 2018. Disponível em: https://xbpeng.github.io/projects/SFV/index.html. Acesso em: 22 jul. 2020.

RANGANATH, A.; XU, P.; KARAMOUZAS, I.; ZORDAN, V. Low dimensional motor skill learning using coactivation. In: **Motion, Interaction and Games**. New York, NY, USA: Association for Computing Machinery, 2019. p. 1–10. Disponível em: https://doi.org/10.1145/3359566.3360071. Acesso em: 21 jun. 2020.

RASCHKA, S.; MIRJALILI, V. **Machine Learning mit Python und Scikit-Learn und TensorFlow**: Das umfassende praxis-handbuch für data science, predictive analytics und deep learning. Berlim, DE: MITP-Verlags GmbH & Co. KG, 2017.

RUSSELL, S.; NORVIG, P. **Artificial intelligence**: a modern approach. Nova Jersey, EUA: Prentice Hall, 2002.

SAFONOVA, A.; HODGINS, J. K.; POLLARD, N. S. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. **ACM Transactions on Graphics (ToG)**, ACM New York, NY, USA, v. 23, n. 3, p. 514–521, 2004. Disponível em: https://doi.org/10.1145/1015706.1015754. Acesso em: 19 jul. 2020.

SILVA, D. B. d. **Um modelo de contato simplificado para tratamento de equilíbrio**. Dissertação (Mestrado) – Universidade Federal do Ceará - UFC, 2014. Disponível em: http://www.mdcc.ufc.br/teses/doc_download/250-202-danilo-borges-da-silva. Acesso em: 03 set. 2019.

SILVER, D.; SCHRITTWIESER, J.; SIMONYAN, K.; ANTONOGLOU, I.; HUANG, A.; GUEZ, A.; HUBERT, T.; BAKER, L.; LAI, M.; BOLTON, A. *et al.* Mastering the game of go without human knowledge. **nature**, Nature Publishing Group, v. 550, n. 7676, p. 354–359, 2017. Disponível em: https://www.deepmind.com/publications/mastering-the-game-of-go-without-human-knowledge. Acesso em: 05 jul. 2020.

SOUSA, A. S. de; NUNES, R. F.; VIDAL, C. A.; CAVALCANTE-NETO, J. B.; SILVA, D. B. da. Physics-based motion control through drl's reward functions. In: **Symposium on Virtual and Augmented Reality**. New York, NY, USA: Association for Computing Machinery, 2022. (SVR'21), p. 127–136. ISBN 9781450395526. Disponível em: https://doi.org/10.1145/3488162.3488217. Acesso em: 12 nov. 2022.

SUTTON, R. S.; BARTO, A. G. **Reinforcement learning**: An introduction. Second. Cambridge, EUA: MIT press, 2018.

TASSA, Y.; DORON, Y.; MULDAL, A.; EREZ, T.; LI, Y.; CASAS, D. d. L.; BUDDEN, D.; ABDOLMALEKI, A.; MEREL, J.; LEFRANCQ, A. *et al.* Deepmind control suite. **arXiv preprint arXiv:1801.00690**, 2018. Disponível em: https://www.deepmind.com/open-source/deepmind-control-suite. Acesso em: 12 jul. 2020.

UNITY. **Unity ML-Agents Toolkit Documentation**. 2020. Disponível em: https://github.com/Unity-Technologies/ml-agents/blob/release_3_docs/docs/Readme.md. Acesso em: 05 abr. 2022.

UNITY. **Welcome to Unity**. 2020. Disponível em: https://unity.com/our-company. Acesso em: 10 maio 2022.

VINYALS, O.; BABUSCHKIN, I.; CZARNECKI, W. M.; MATHIEU, M.; DUDZIK, A.; CHUNG, J.; CHOI, D. H.; POWELL, R.; EWALDS, T.; GEORGIEV, P. *et al.* Grandmaster level in starcraft ii using multi-agent reinforcement learning. **Nature**, Nature Publishing Group, v. 575, n. 7782, p. 350–354, 2019. Disponível em: https://www.seas.upenn.edu/~cis520/papers/RL_for_starcraft.pdf. Acesso em: 12 jul. 2020.

WAMPLER, K.; POPOVIC, Z. Optimal gait and form for animal locomotion. **ACM Trans. Graph.**, Association for Computing Machinery, New York, NY, USA, v. 28, n. 3, jul. 2009. ISSN 0730-0301. Disponível em: https://doi.org/10.1145/1531326.1531366. Acesso em: 03 jul. 2020.

WAMPLER, K.; POPOVIC, Z.; POPOVIC, J. Generalizing locomotion style to new animals with inverse optimal regression. **ACM Trans. Graph.**, Association for Computing Machinery, New York, NY, USA, v. 33, n. 4, jul. 2014. ISSN 0730-0301. Disponível em: https://doi.org/10.1145/2601097.2601192. Acesso em: 03 jul. 2020.

YANG, C.; YUAN, K.; HENG, S.; KOMURA, T.; LI, Z. Learning natural locomotion behaviors for humanoid robots using human bias. **IEEE Robotics and Automation Letters**, IEEE, v. 5, n. 2, p. 2610–2617, 2020. Disponível em: https://doi.org/10.1109/LRA.2020.2972879. Acesso em: 10 nov. 2021.

YE, Y.; LIU, C. K. Animating responsive characters with dynamic constraints in near-unactuated coordinates. In: **ACM SIGGRAPH Asia 2008 papers**. New York, NY, USA: Association for Computing Machinery, 2008. p. 1–5. Disponível em: https://doi.org/10.1145/1409060.1409065. Acesso em: 19 jul. 2020.

YU, W.; TURK, G.; LIU, C. K. Learning symmetric and low-energy locomotion. **ACM Transactions on Graphics (TOG)**, ACM New York, NY, USA, New York, NY, USA, v. 37, n. 4, p. 1–12, 2018. Disponível em: https://doi.org/10.1145/3197517.3201397. Acesso em: 10 jun. 2020.

ZORDAN, V. B.; HODGINS, J. K. Motion capture-driven simulations that hit and react. In: **Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation**. New York, NY, USA: Association for Computing Machinery, 2002. p. 89–96. Disponível em: https://doi.org/10.1145/545261.545276. Acesso em: 03 jun. 2020.

ANEXO A – HIPERPARÂMETROS USADOS NOS TREINAMENTOS

Segue abaixo um dos arquivos de configuração utilizados no treinamento do Humano 3D, esse arquivo de Juliani *et al.* (2020) e usado no ambiente do personagem "Walker". Este arquivo serviu de base para os experimentos utilizados neste trabalho, e ele contém informações importantes sobre os hiperparâmetros da rede neural e do algoritmo de treinamento utilizado. Dados como o número de camadas da rede, o número de neurônios em cada camada, o tipo de ativação utilizado e a taxa de aprendizado, são alguns exemplos.

Além disso, esse arquivo também contém informações sobre o ambiente de simulação, como o tamanho do episódio, a duração da simulação e as condições de término. Ele também pode conter configurações adicionais, como o uso de técnicas de regularização ou ajustes de parâmetros para o algoritmo de treinamento.

Esse arquivo específico contém informações sobre o treinamento do agente chamado "WalkerStatic". Os principais parâmetros especificados nele são:

- trainer_type - define qual algoritmo de treinamento será utilizado, neste trabalho foi usado o PPO;
- learning_rate - define a velocidade de aprendizado do algoritmo, especificando quanto o peso deve ser ajustado em resposta a cada atualização de gradiente;
- beta - define o valor a ser utilizado na fórmula da entropia para regularização da política do agente;
- epsilon - define a taxa de atualizações da política para que sejam suficientemente conservadoras, evitando grandes variações da política em cada passo de atualização;
- hidden_units - especifica o número de neurônios em cada camada oculta da rede neural utilizada no treinamento;
- num_layers - define o número de camadas na rede neural utilizada no algoritmo de treinamento;
- max_steps - define o número máximo de passos que os agentes executam durante o treinamento.

```yaml
# WalkerStatic.yaml
# Hiperparâmetros usados nos treinamentos do Walker 3D
behaviors:
    WalkerStatic:
        trainer_type: ppo
        hyperparameters:
            batch_size: 2048
            buffer_size: 20480
            learning_rate: 0.0003
            beta: 0.005
            epsilon: 0.2
            lambd: 0.95
            num_epoch: 3
            learning_rate_schedule: linear
        network_settings:
            normalize: true
            hidden_units: 512
            num_layers: 3
            vis_encode_type: simple
        reward_signals:
            extrinsic:
                gamma: 0.995
                strength: 1.0
        keep_checkpoints: 5
        max_steps: 20000000
        time_horizon: 1000
        summary_freq: 30000
        threaded: true
```