SISTEMAS MULTIAGENTES E O TRÁFEGO URBANO

MULT-AGENTE SYSTEMS AND URBAN TRAFFIC

Michael Washington dos Santos[1], Murilo Lopes Bianchi[2], Marcelo Tadeu Boer[3]

[1]Fundação Educacional de Fernandópolis, maicopwgpde@gmail.com
[2]Fundação Educacional de Fernandópolis, murilolopes.b@gmail.com
[3]Fundação Educacional de Fernandópolis, marcelotadeuboer@fef.edu.br

Área Ciências da Computação
Subárea: Inteligência Artificial

RESUMO

Este trabalho investiga o impacto da comunicação entre veículos autônomos cooperativos no contexto de sistemas de transporte inteligentes, utilizando aprendizado por reforço profundo em um ambiente simulado. Foi desenvolvido um cenário em Unity, com dois agentes veiculares (líder e seguidor) trafegando em uma rodovia de três faixas com obstáculos dinâmicos. Ambos os agentes foram treinados utilizando o framework Unity ML-Agents e o algoritmo *Proximal Policy Optimization* (PPO), em episódios múltiplos, registrando métricas como sucesso na conclusão do trajeto, número de colisões, duração do episódio e recompensa acumulada. Os resultados indicam que o agente seguidor com comunicação apresenta trajetórias mais estáveis, maior taxa de sucesso e menor incidência de colisões em comparação ao cenário sem comunicação, no qual o comportamento é mais reativo e sujeito a manobras bruscas. Apesar das limitações relacionadas ao tempo de treinamento, capacidade computacional e simplificações do ambiente, os experimentos evidenciam o potencial da comunicação entre agentes para aprimorar a segurança e a eficiência em tráfego veicular autônomo cooperativo.

Palavras-chave: Veículos Autônomos. Sistemas Multiagentes. Unity ML-Agents. Algoritimo Proximal Policy Optimization

ABSTRACT

This work investigates the impact of communication between cooperative autonomous vehicles in the context of intelligent transportation systems, using deep reinforcement learning in a simulated environment. A scenario was developed in Unity with two vehicular agents (leader and follower) traveling on a three-lane highway with dynamic obstacles. Both agents were trained using the Unity ML-Agents framework and the Proximal Policy Optimization (PPO) algorithm over multiple episodes, recording metrics such as task-completion success rate, number of collisions, episode duration, and accumulated reward.
The results indicate that the follower agent with communication exhibits more stable trajectories, a higher success rate, and fewer collisions compared to the scenario without communication, in which the behavior is more reactive and prone to abrupt maneuvers. Despite limitations related to training time, computational capacity, and environmental simplifications, the experiments highlight the potential of inter-agent communication to enhance safety and efficiency in cooperative autonomous vehicle traffic.

Keywords: *Autonomous Vehicles. Multi-agent Systems. Unity ML-Agents. Algorithm Proximal Policy Optimization.*

1. INTRODUÇÃO

A crescente urbanização e o aumento do número de veículos nas vias públicas têm gerado desafios significativos no gerenciamento do tráfego, especialmente em grandes centros urbanos. Acidentes, congestionamentos e a necessidade de respostas rápidas a imprevistos são apenas alguns dos problemas enfrentados diariamente. Nesse contexto, a tecnologia tem desempenhado um papel fundamental na busca por soluções mais eficientes e inteligentes.

Este trabalho propõe o desenvolvimento de uma simulação de tráfego baseada em um sistema multiagente de inteligência artificial, na qual os agentes — representando veículos autônomos — são capazes de se comunicar entre si para compartilhar informações sobre o ambiente, como a presença de obstáculos na via. A comunicação entre os agentes permite que decisões coordenadas sejam tomadas, otimizando o fluxo de tráfego e aumentando a segurança.

O foco da simulação é um cenário de estrada com três faixas, em que um veículo que detecta um obstáculo à frente envia um alerta aos veículos que estão atrás, possibilitando que estes reajam com antecedência e desviem do obstáculo de maneira coordenada. Dessa forma, busca-se investigar como a troca de informações entre veículos influenciam o comportamento global do tráfego em um ambiente controlado.

A modelagem do comportamento dos agentes é realizada por meio de aprendizado por reforço, com uso do pacote Unity ML-Agents, sem a necessidade de reproduzir com fidelidade todos os aspectos físicos de um veículo real.

O foco recai sobre a análise comparativa entre dois cenários: um em que o veículo seguidor se baseia apenas em suas percepções locais e outro em que recebe informações adicionais provenientes da comunicação com o veículo líder.

Este trabalho está organizado da seguinte forma: No Capítulo 2, apresenta-se o referencial teórico necessário para o entendimento do estudo, abordando inteligência artificial, aprendizado de máquina, aprendizado por reforço, sistemas multiagentes e comunicação entre agentes em cenários de tráfego. No Capítulo 3, descrevem-se a metodologia adotada, o ambiente de simulação desenvolvido, a arquitetura dos agentes e o processo de treinamento com o Unity ML-Agents. No Capítulo 4, são apresentados e analisados os resultados obtidos nos cenários com e sem comunicação entre veículos, com base nas métricas coletadas durante os episódios de simulação. Por fim, o Capítulo 5 traz as conclusões do trabalho, as limitações identificadas e sugestões de trabalhos futuros relacionados ao tema

2 REFERENCIAL TEÓRICO

2.1 Inteligência Artificial

A inteligência artificial (IA) é um campo da ciência que se concentra na criação de sistemas computacionais capazes de raciocinar, aprender e agir de maneira que, tradicionalmente, exigiria inteligência humana (RUSSELL; NORVIG, 2021). Trata-se de uma área interdisciplinar que envolve ciência da computação, estatística, neurociência, filosofia, psicologia e engenharia. A Figura 1 é utilizada para exemplificar conceitos de Sistemas com Inteligência Artificial.

Figura 1 – Exemplos de Sistemas com Inteligência Artificial

Fonte: Elaborada pelos autores.

No contexto aplicado, a IA moderna é amplamente baseada em métodos de aprendizado de máquina e aprendizado profundo, utilizados para classificação, previsão, reconhecimento de padrões, processamento de linguagem natural e sistemas de recomendação (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.2 Aprendizado de Máquina (Machine Learning)

O aprendizado de máquina é um subconjunto da inteligência artificial dedicado ao desenvolvimento de algoritmos capazes de aprender padrões e realizar generalizações a partir de dados, sem que regras explícitas sejam programadas (MITCHELL, 1997). À medida que o modelo é exposto a mais dados, ajusta seus parâmetros internos e tende a melhorar seu desempenho.

Modelos baseados em redes neurais profundas têm se destacado em tarefas como visão computacional e reconhecimento de fala, devido à sua capacidade de extrair representações hierárquicas de alta complexidade (LECUN; BENGIO; HINTON, 2015). A Figura 2 exemplifica-se os tipos de aprendizado de máquina: aprendizado supervisionado, o modelo aprende a partir de dados já com respostas corretas (rótulos). O objetivo é prever ou classificar novas informações; aprendizado não supervisionado, trabalha com dados sem respostas e tenta encontrar padrões ou grupos por conta própria; aprendizado por reforço, aprende tentando ações e recebendo recompensas ou punições. Assim, ele descobre a melhor forma de agir em cada situação.

Figura 2 – Tipos de Aprendizado de Máquina

Fonte: Elaborada pelos autores.

2.3 Inteligência Artificial Distribuída

A Inteligência Artificial Distribuída (IAD) estuda como múltiplos agentes autônomos podem cooperar e coordenar suas ações para resolver problemas de forma conjunta, mesmo quando cada agente possui percepção parcial do ambiente. Em sistemas distribuídos, o comportamento global emerge das decisões locais tomadas por cada agente, sendo esse paradigma particularmente útil em domínios como tráfego veicular, onde decisões precisam ser tomadas em tempo real, tornando inviável depender de um controlador centralizado (FERBER, 1999).

2.4 Sistemas Multiagentes

Sistemas multiagentes (SMA) constituem uma abordagem estruturada da IAD, na qual várias entidades autônomas interagem em um ambiente comum para alcançar objetivos individuais ou coletivos. Segundo Weiss (2013), SMA envolvem mecanismos de coordenação, negociação e cooperação entre agentes.

Para Wooldridge (2009), um agente é uma entidade capaz de perceber o ambiente, raciocinar e agir de maneira autônoma. A interação entre múltiplos agentes permite a resolução de problemas complexos que seriam inviáveis para um único agente isolado.

Aplicações incluem direção autônoma, robótica colaborativa, simulações sociais, negociação automatizada e coordenação em ambientes hostis.

2.5 Comunicação entre agentes

A comunicação é um componente fundamental de sistemas multi-agentes, permitindo que agentes compartilhem percepções, intenções e estados internos. Linguagens padronizadas como a FIPA ACL (*Foundation for Intelligent Physical Agents — Agent Communication Language*) definem estruturas e atos de fala utilizados em plataformas como JADE. Conforme discutido por Wooldridge, os sistemas multi-agente (MAS) oferecem o arcabouço teórico para modelar agentes autônomos que interagem e cooperam.

Segundo Wooldridge (2009), protocolos de comunicação e estruturas de interação são essenciais para garantir cooperação e coordenação em ambientes dinâmicos.

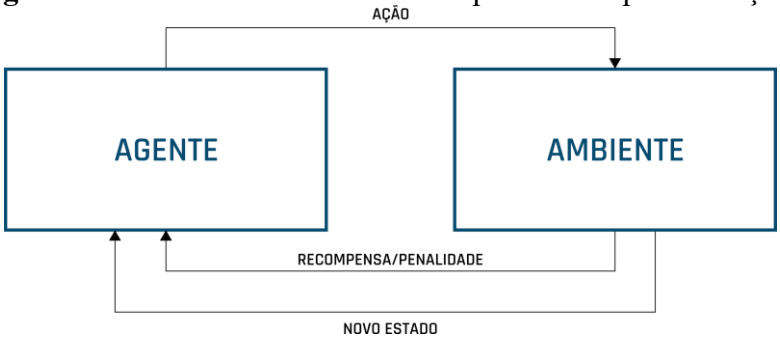
No contexto deste trabalho, a comunicação ocorre entre veículos-autônomos-agentes, onde um veículo líder transmite informações sobre obstáculos para veículos seguidores — uma abordagem alinhada com pesquisas recentes sobre cooperação em tráfego inteligente. Por exemplo, estudos no contexto de veículos conectados/autônomos têm explorado o compartilhamento de percepções e decisões entre veículos para coordenar manobras, evitar colisões e melhorar o fluxo de tráfego.

2.6 Aprendizado por reforço

O aprendizado por reforço (*Reinforcement Learning* – RL) é um paradigma em que um agente aprende por tentativa e erro ao interagir com o ambiente e receber recompensas que indicam a qualidade de suas ações (SUTTON; BARTO, 2018). Problemas de RL são frequentemente modelados como Processos de Decisão de Markov (MDP) (PUTERMAN, 1994).

O dilema exploração–exploração é central na literatura, exigindo que o agente equilibre a busca por novas ações com o uso daquelas já conhecidas como eficazes (SUTTON; BARTO, 2018). Em ambientes com estados contínuos e alta dimensionalidade, torna-se necessário o uso de técnicas de aprendizado por reforço profundo. A Figura 3 exemplifica um diagrama que sintetiza o ciclo de decisões, observações e recompensas que caracteriza esse processo.

Figura 3 – Estrutura do Processo de Aprendizado por Reforço



Fonte: Elaborada pelos autores.

2.7 Aprendizado por reforço profundo e Unity ML-Agents

O aprendizado por reforço profundo (*Deep Reinforcement Learning* – DRL) combina RL com redes neurais profundas para lidar com ambientes complexos (FRANCOIS-LAVET et al., 2018). Ao invés de tabelas de valores, redes neurais são usadas para aproximar políticas e funções de valor (MNIH et al., 2015).

O Unity ML-Agents oferece uma interface que integra ambientes 3D com algoritmos modernos de RL, permitindo treinamentos com políticas contínuas ou discretas (JULIAN et al., 2020). Neste trabalho utiliza-se o algoritmo (*Proximal Policy Optimization* – PPO), popular por sua estabilidade e eficiência (SCHULMAN et al., 2017). YAML é um formato de arquivo de configuração baseado em texto, feito para ser fácil de ler por humanos. Ele usa identação por espaços para representar hierarquia (níveis de configuração). No ML-Agents, os arquivos .yaml são usados para definir, hiperparâmetros como taxa de aprendizado, tamanho de lote (*batch size*), gamma e arquitetura da rede — componentes essenciais para a convergência do treinamento.

2.8 Sistemas de transporte inteligentes e veículos autônomos cooperativos

Sistemas de Transporte Inteligentes (*Intelligent Transportation Systems* – ITS) aplicam tecnologias de comunicação, sensores e algoritmos avançados para monitorar e melhorar o fluxo de tráfego. Veículos autônomos utilizam sensores como câmeras, radares e LIDARs para perceber o ambiente e planejar trajetórias (THRUN; MONTEMERLO, 2006).

A comunicação veículo-veículo (V2V) é um elemento-chave para cooperação entre veículos, reduzindo colisões e melhorando a fluidez do tráfego. Simulações computacionais têm sido amplamente usadas para testar comportamentos cooperativos de agentes autônomos antes de implementações reais.

Este trabalho se apoia nessa perspectiva ao comparar cenários com e sem comunicação entre veículos, analisando o impacto da troca de informações na qualidade das decisões e no fluxo de tráfego.

3. METODOLOGIA

3.1 Tipo de pesquisa e abordagem

Este trabalho consiste no desenvolvimento de uma simulação computacional com o objetivo de demonstrar a aplicação de sistemas multiagentes em cenários de tráfego veicular, utilizando aprendizado por reforço.

A pesquisa caracteriza-se como aplicada, pois busca abordar um problema prático — a otimização do tráfego veicular e a redução de colisões — por meio do uso de técnicas de inteligência artificial; exploratória, uma vez que investiga a aplicação de inteligência artificial distribuída e comunicação entre agentes em um ambiente simulado, analisando os efeitos dessa comunicação sobre o comportamento dos veículos e experimental, pois envolve a definição de cenários controlados (com e sem comunicação entre agentes), o treinamento de agentes em cada cenário e a comparação sistemática dos resultados obtidos.

A abordagem adotada combina análise quantitativa, por meio da coleta de métricas numéricas a cada episódio de simulação, e interpretação qualitativa, ao se observar o comportamento emergente dos agentes em situações como aproximação de obstáculos, trocas de faixa e conclusão do percurso.

3.2 Ferramentas e ambiente de desenvolvimento

A simulação foi desenvolvida utilizando o motor de jogos Unity, com a linguagem de programação C#, por se tratar de uma plataforma adequada para a criação de ambientes tridimensionais interativos, com suporte a física, animação e integração com bibliotecas de inteligência artificial.

Para o treinamento dos agentes por aprendizado por reforço foi utilizado o pacote Unity ML-Agents, que faz a ponte entre o ambiente Unity e algoritmos de aprendizado por reforço profundo. O uso dessa ferramenta permite definir agentes diretamente na cena, configurar observações, ações e recompensas, além de treinar políticas utilizando algoritmos modernos, como o (PPO).

A escolha desse conjunto de ferramentas se justifica pela possibilidade de: modelar um ambiente dinâmico e visualmente representativo; ter controle detalhado sobre a lógica dos agentes por meio de scripts em C#;integrar facilmente a simulação com algoritmos de aprendizado por reforço profundo via ML-Agents; registrar métricas de desempenho de forma estruturada para posterior análise.

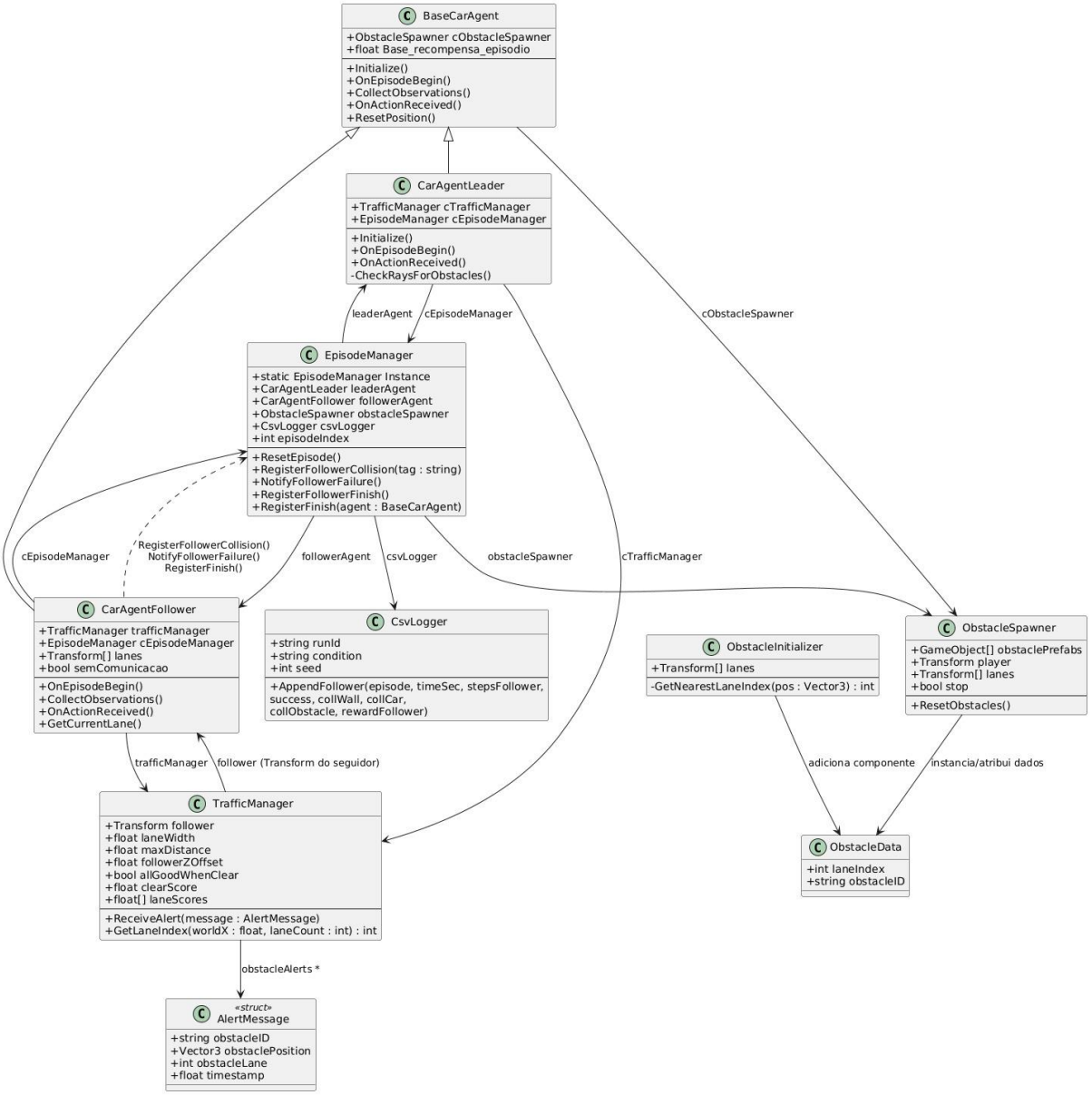
3.3 Arquitetura do sistema e agentes

O ambiente de simulação foi desenvolvido na plataforma Unity, utilizando o pacote ML-Agents para o treinamento dos agentes por aprendizado por reforço. A cena principal representa uma pista retilínea com três faixas de rolamento, ao longo da qual são posicionados obstáculos estáticos que devem ser evitados pelos veículos.

A arquitetura do sistema é composta por diferentes componentes de software, organizados em scripts C#, que desempenham funções específicas. A seguir são descritos os principais módulos exemplificados por meio da Figura 4.

3.3.1 Agentes veiculares

Figura 4 – Diagrama Agentes Veiculares



Fonte: Elaborada pelos autores.

A classe BaseCarAgent define o comportamento básico de movimento dos veículos, incluindo a dinâmica de avanço na pista, a aplicação de aceleração para frente e movimentos laterais entre faixas, bem como a lógica de término de episódio. Esse componente concentra a estrutura de recompensas base recebidas pelo agente ao progredir na pista e penalidades associadas a eventos indesejados, como colisões e extrapolação do tempo máximo de episódio. As classes específicas de agente (líder e seguidor) herdam dessa base e adicionam suas particularidades.

A classe CarAgentLeader representa o veículo líder. Esse agente utiliza sensores de percepção (por exemplo, sensores de raios configurados via ML-Agents) para detectar

obstáculos à frente. Ao identificar situações de risco, o líder gera mensagens de alerta (AlertMessage) que são enviadas ao componente de gerenciamento de tráfego. O agente líder recebe recompensas ao completar a pista sem colisões e penalidades quando ocorrem impactos com obstáculos ou paredes, seguindo a lógica de recompensas herdada da classe base, com ajustes específicos para seu papel de "batedor" na pista.

A classe CarAgentFollower representa o veículo seguidor e é o foco principal de análise neste trabalho. Esse agente possui dois modos de operação:

Modo sem comunicação: o seguidor toma decisões apenas com base em suas observações locais do ambiente, obtidas por sensores na própria entidade (como distâncias até obstáculos, posição e velocidade atual, faixa em que se encontra). Modo com comunicação: além das percepções locais, o agente recebe informações adicionais sobre o estado das faixas, fornecidas pelo componente TrafficManager, derivadas das mensagens de alerta enviadas pelo líder.

As observações do seguidor incluem, entre outros elementos, dados de posição e velocidade, identificação da faixa atual e escores numéricos associados à segurança de cada faixa. A função de recompensa do seguidor combina a recompensa base por avançar na pista com recompensas e penalidades específicas relacionadas à escolha de faixas, à presença de obstáculos e à ocorrência de colisões.

O componente TrafficManager é responsável por receber as mensagens de alerta geradas pelo veículo líder e manter uma representação dos obstáculos relevantes à frente do seguidor. A partir dessas informações, o componente calcula, para cada faixa, um escore de segurança (laneScore), que indica o quão livre aquela faixa se encontra em relação a obstáculos nas proximidades.Esses escores são utilizados como parte das observações do agente seguidor no cenário com comunicação, permitindo que o agente antecipe mudanças de faixa com base em informações que vão além do seu campo de percepção local imediato.

O componente ObstacleSpawner é responsável pela geração controlada de obstáculos ao longo da pista, respeitando parâmetros como faixa de rolamento, distância mínima e máxima entre obstáculos e limite de obstáculos ativos na cena. Obstáculos que ficam muito atrás do veículo, ou tornam-se irrelevantes para o episódio em andamento, podem ser destruídos para otimizar o desempenho da simulação.

O script ObstacleInitializer é utilizado para configurar obstáculos posicionados diretamente na cena, associando-os às faixas correspondentes logo no início da simulação. Uma estrutura de dados auxiliar (como uma classe ObstacleData) armazena informações sobre cada obstáculo, como identificador único e índice da faixa em que se encontra.

O componente EpisodeManager coordena o início e o término dos episódios de simulação. A cada novo episódio, são reinicializados o estado dos agentes e a disposição dos obstáculos, garantindo condições controladas para o treinamento e a coleta de dados. Esse gerenciador registra o tempo de início do episódio, o número de passos realizados pelo agente seguidor e os tipos de colisões que ocorrem. Ao final de um episódio, o componente calcula a duração, verifica se o seguidor completou com sucesso o percurso e coleta a recompensa acumulada. Esses dados são então repassados ao módulo de registro em arquivo.

O script CsvLogger é responsável pelo registro das métricas em arquivo no formato CSV. Para cada episódio, são salvos dados como: identificador de execução (run_id), condição do experimento (comunicação ou não), semente de aleatoriedade utilizada, número do episódio, duração em segundos, passos do agente seguidor, indicador de sucesso, contagem de colisões por tipo e recompensa final do seguidor. O logger permite configurar o caminho de saída dos arquivos e serve de base para a posterior análise estatística e visualização dos resultados em ferramentas externas.

3.4 Processo de treinamento dos agentes

O treinamento dos agentes foi conduzido utilizando o Unity ML-Agents, por meio de arquivos de configuração no formato YAML. Nesses arquivos são especificados tanto os parâmetros do algoritmo de aprendizado por reforço quanto as informações sobre os comportamentos associados a cada agente na cena.

O algoritmo escolhido para o treinamento foi o PPO, devido à sua estabilidade, à ampla utilização em cenários contínuos e ao suporte nativo no ML-Agents.

Entre os principais hiperparâmetros configurados destacam-se: taxa de aprendizado do otimizador; tamanho dos lotes de experiência; número de passos de coleta de dados por atualização; fator de desconto das recompensas; coeficiente de entropia[1], que controla o grau de exploração da política; número de épocas de atualização da rede a cada iteração de treino.

Os valores desses hiperparâmetros foram inicialmente definidos com base nas recomendações presentes na documentação do ML-Agents e, em seguida, ajustados empiricamente, observando-se a evolução das recompensas e a estabilidade do comportamento dos agentes ao longo do treinamento.

Para cada condição experimental (com comunicação e sem comunicação), foram realizadas sessões de treinamento independentes, utilizando diferentes sementes de aleatoriedade para o ambiente. Esse procedimento permite: avaliar a robustez das políticas aprendidas; reduzir a influência de particularidades de uma única execução; obter amostras mais representativas para a comparação entre cenários.

Durante o treinamento, as redes neurais que definem a política dos agentes foram atualizadas de forma iterativa, a partir das recompensas obtidas em múltiplos episódios de simulação. Ao final de cada sessão, foram selecionados os modelos com melhor desempenho segundo as métricas de interesse para serem utilizados na fase de avaliação. A (Figura 5) é utilizada para exemplificar o ambiente de simulação do treinamento, composto por uma pista em linha reta onde os agentes interagem com outros veículos. Esse cenário simples permite avaliar de forma clara os comportamentos aprendidos e o desempenho das políticas geradas durante o treinamento.

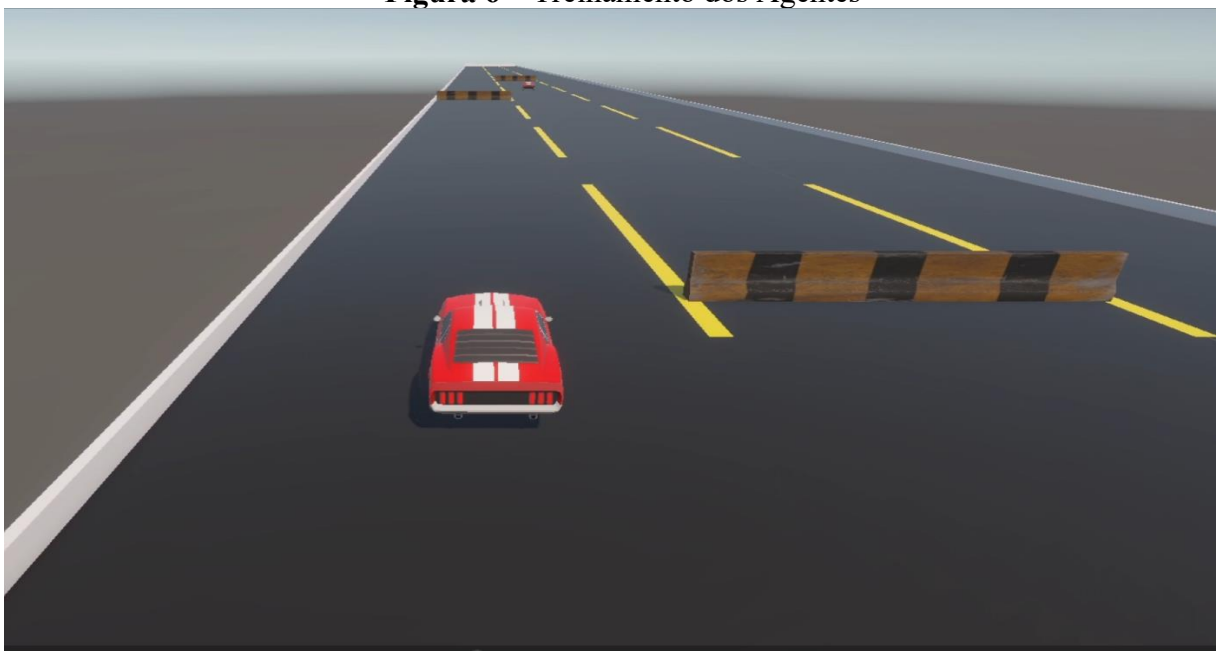
[1] Medida que, num sistema termodinâmico, determina o grau de desordem, pela ação de uma temperatura; representada por "S": gelo derretendo é um exemplo comum de aumento da entropia.

Figura 5 – Ambiente da Simulação

Fonte: Elaborada pelos autores.

Utiliza-se a (Figura 6) para exemplificar uma etapa do processo de treinamento dos agentes, mostrando o veículo interagindo com obstáculos posicionados ao longo da pista. Esses elementos foram incluídos para incentivar a aprendizagem de comportamentos de desvio e tomada de decisão em situações dinâmicas, contribuindo para o desenvolvimento de políticas mais robustas durante o treinamento.

Figura 6 – Treinamento dos Agentes



Fonte: Elaborada pelos autores.

3.5 Cenários experimentais: com e sem comunicação

Para avaliar o impacto da comunicação entre veículos no desempenho do sistema, foram definidos dois cenários experimentais principais. Em ambos os casos, a estrutura básica do ambiente — pista de três faixas, obstáculos estáticos e papel dos agentes líder e seguidor — é mantida, variando-se apenas a disponibilidade de informações fornecidas ao seguidor.

3.5.1 Cenário sem comunicação

No cenário sem comunicação, o veículo seguidor opera com o parâmetro de configuração semComunicacao ativado. Nesse modo: o agente utiliza apenas suas percepções locais para tomar decisões; as observações são compostas por informações obtidas pelos sensores acoplados ao próprio seguidor (como distâncias até obstáculos, posição e velocidade) e pela identificação da faixa atual; os escores de faixa calculados pelo TrafficManager não são incluídos no vetor de observações; a função de recompensa é ajustada para desconsiderar a influência direta de qualquer forma de comunicação, focando no avanço seguro e na evasão de colisões com base apenas na percepção local.

Esse cenário funciona como linha de base (baseline) para comparação com o cenário em que há comunicação explícita entre agentes.

3.5.2 Cenário com comunicação

No cenário com comunicação, o veículo líder, ao detectar obstáculos à frente, envia mensagens de alerta ao TrafficManager. Esse componente: atualiza uma representação interna dos obstáculos relevantes à frente do seguidor; calcula escores para cada faixa, indicando o quão livre ou perigosa aquela faixa se encontra em uma região de interesse à frente do veículo; disponibiliza esses escores como parte das observações recebidas pelo agente seguidor.

Dessa forma, o seguidor passa a ter acesso não apenas às suas percepções locais, mas também a informações agregadas provenientes da comunicação indireta com o líder, mediada pelo TrafficManager.

A função de recompensa nesse cenário é ajustada para: incentivar a escolha de faixas consideradas mais seguras segundo os escores recebidos; penalizar a permanência em faixas ocupadas ou perigosas; manter penalidades em caso de colisões e recompensas pelo avanço consistente na pista.

A comparação entre o cenário sem comunicação e o cenário com comunicação permite analisar em que medida a troca de informações entre veículos contribui para a redução de colisões e para a melhoria do desempenho geral do agente seguidor ao longo dos episódios de simulação.

3.6 Métricas de avaliação

Para avaliar o desempenho dos agentes nos diferentes cenários, foram coletadas, ao final de cada episódio, as seguintes métricas, registradas pelo componente CsvLogger:

Sucesso do episódio (success): variável binária que indica se o agente seguidor conseguiu completar o percurso até o final da pista sem colisões (1 para sucesso e 0 para falha).

Número de passos do seguidor (*steps_follower*): quantidade de decisões tomadas pelo agente seguidor durante o episódio, correspondente ao número de passos de simulação em que a política foi executada.

Duração do episódio (*time_sec*): tempo, em segundos, entre o início e o fim do episódio, calculado a partir do registro do EpisodeManager.

Colisões por tipo (c*ollisions_wall, collisions_car, collisions_obstacl*e): contagem de colisões do seguidor com paredes, outros veículos e obstáculos na pista, respectivamente.

Recompensa acumulada do seguidor *(reward_follower*): soma de todas as recompensas recebidas pelo agente seguidor ao longo do episódio, refletindo tanto o avanço na pista quanto penalidades por comportamentos indesejados.

Essas métricas são salvas em arquivos CSV para cada condição experimental (com e sem comunicação) e posteriormente analisadas por meio de ferramentas de visualização e estatística. A partir delas, é possível comparar quantitativamente o desempenho dos agentes, identificar diferenças na segurança (redução de colisões) e na eficiência (tempo e número de passos) e avaliar o impacto da comunicação entre agentes no tráfego simulado.

4 RESULTADOS E DISCUSSÃO

4.1 Configuração dos experimentos de avaliação

Após o término do processo de treinamento descrito no Capítulo 3, foram conduzidos experimentos de avaliação com o objetivo de comparar o desempenho do agente seguidor em dois cenários: cenário sem comunicação: o agente seguidor toma decisões com base apenas em suas percepções locais, sem acesso às informações de faixa calculadas pelo TrafficManager.

Cenário com comunicação: além das percepções locais, o agente seguidor recebe escores de segurança das faixas, derivados das mensagens de alerta geradas pelo agente líder e processadas pelo TrafficManager.

Em ambos os cenários, foram utilizados modelos treinados com o algoritmo PPO, conforme configurado nos arquivos YAML do Unity ML-Agents. As simulações foram realizadas sobre o mesmo ambiente de pista retilínea com três faixas e obstáculos estáticos distribuídos ao longo do trajeto.

Ao todo, foram analisados 1050 episódios, distribuídos entre os dois modos de operação do agente seguidor, com registro das métricas de interesse pelo componente CsvLogger, a saber: sucesso do episódio, número de passos do seguidor, duração em segundos, colisões por tipo e recompensa acumulada.

Os resultados são apresentados a seguir por meio de gráficos e discussão qualitativa, com foco na comparação entre os cenários com e sem comunicação.

4.2 Resultados quantitativos dos treinamentos

4.2.1 Visão geral das métricas por episódio

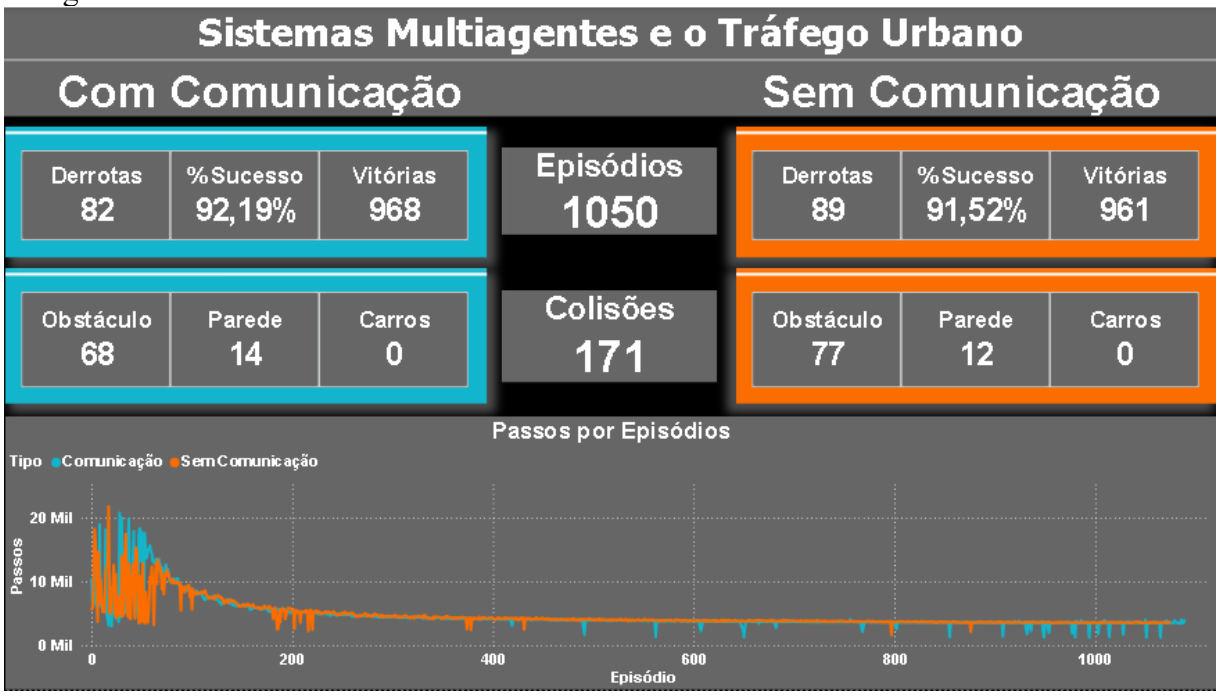
O gráfico geral de resultados exemplificado pela Figura 7, sintetiza o comportamento das principais métricas ao longo dos episódios avaliados, considerando tanto o modo com comunicação quanto o modo sem comunicação.

De maneira geral, observa-se que o treinamento do agente sem comunicação é mais instável, apresentando: maior frequência de colisões; taxa de sucesso inferior; variação mais acentuada no número de passos por episódio.

Em contraste, o agente com comunicação tende a apresentar um comportamento mais consistente, com episódios mais estáveis em termos de passos realizados e melhor desempenho médio nas métricas de sucesso e segurança. Essa diferença sugere que a disponibilidade de

informações adicionais sobre o estado das faixas contribui para uma política de navegação mais robusta.

Figura 7 – Comparação do desempenho do sistema multiagentes com e sem comunicação no tráfego urbano.

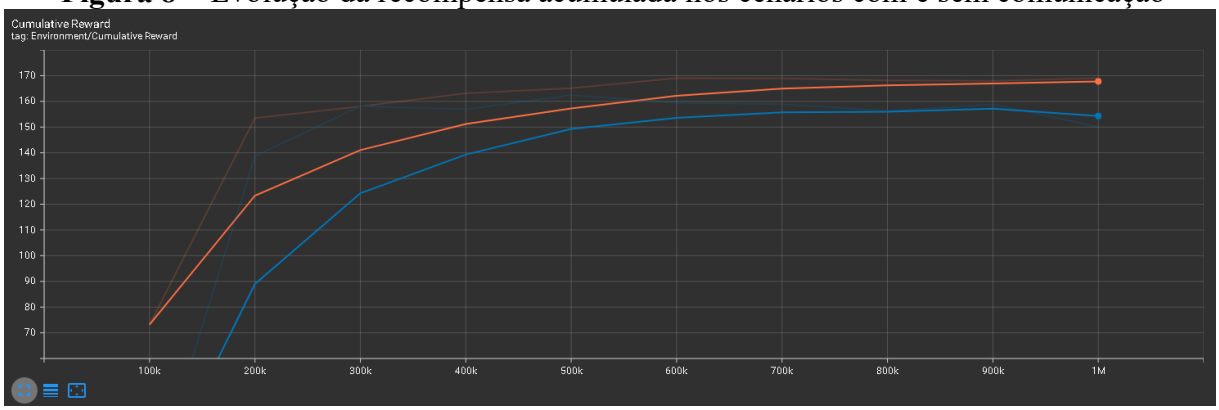


Fonte: Elaborada pelos autores.

4.2.2 Recompensa acumulada

O gráfico de recompensas por episódio, (Figura 8) permite analisar a evolução da recompensa acumulada pelo agente seguidor nos dois cenários.

Figura 8 – Evolução da recompensa acumulada nos cenários com e sem comunicação



Fonte: Elaborada pelos autores.

No cenário sem comunicação, verifica-se que, ao longo do treinamento, o agente alcança recompensas médias superiores às obtidas no cenário com comunicação. Entretanto, nas fases iniciais do treinamento, o agente sem comunicação recebe um número maior de punições, resultando em valores de recompensa mais baixos e, por vezes, negativos.

Esse comportamento inicial indica que, apesar de dispor de mais informações, o agente precisa de um período maior de exploração para aprender a utilizá-las de maneira eficaz. Com

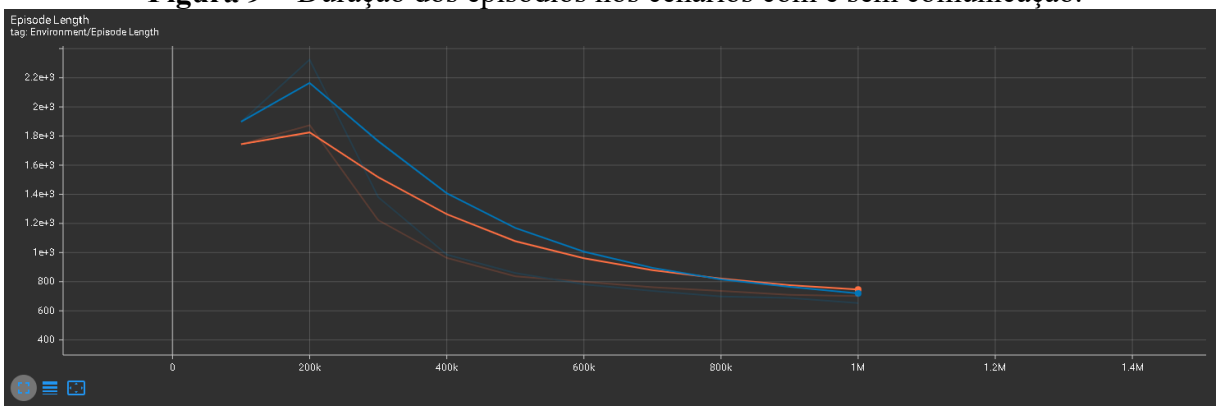
o avanço do treinamento, tanto o agente com comunicação quanto o agente sem comunicação convergem para políticas que maximizam a recompensa acumulada, o que se reflete na presença de trechos mais “estáveis” no gráfico, com valores de recompensa próximos de um patamar máximo para cada cenário.

Esse padrão é coerente com o funcionamento do aprendizado por reforço: após a fase inicial de exploração intensa e punições frequentes, os agentes passam gradualmente a explorar menos e a explorar com maior frequência as ações que levam a trajetórias mais recompensadoras.

4.2.3 Duração dos episódios

O gráfico de duração dos episódios, (Figura 9) apresenta o tempo, em segundos, necessário para a conclusão de cada episódio em ambos os cenários.

Figura 9 – Duração dos episódios nos cenários com e sem comunicação.



Fonte: Elaborada pelos autores.

Observa-se que, durante aproximadamente metade do treinamento, o agente com comunicação leva mais tempo para completar o trajeto, quando comparado ao agente sem comunicação. Esse comportamento reforça a interpretação de que o agente com comunicação enfrenta, inicialmente, maior dificuldade em encontrar uma política eficiente, possivelmente em função do espaço de estados mais rico e das novas possibilidades de ação decorrentes da informação adicional.

No entanto, à medida que o treinamento avança, o agente com comunicação passa a concluir os episódios mais rapidamente, superando o desempenho do agente sem comunicação em termos de tempo mínimo de conclusão. O agente sem comunicação, por sua vez, parece atingir um limite inferior de tempo relativamente cedo, sem apresentar ganhos significativos nas fases finais.

Esse resultado sugere que, com mais tempo de treinamento, a política com comunicação tende a se beneficiar de forma mais expressiva das informações de faixa, combinando maior segurança (menos colisões) com menor duração de episódio, o que indica trajetórias mais fluidas e eficientes.

4.3 Análise qualitativa do comportamento dos agentes

Além das métricas numéricas, foram observados qualitativamente os comportamentos dos agentes durante os episódios de simulação.

No cenário sem comunicação, o agente seguidor apresenta um comportamento predominantemente reativo. Em muitos casos, o agente só inicia a manobra de desvio quando o obstáculo já se encontra muito próximo, o que leva a: paradas bruscas; mudanças de faixa tardias; aumento da probabilidade de colisões ou de trajetórias pouco suaves.

Esse padrão é consistente com o fato de que, sem acesso às informações do TrafficManager, o seguidor baseia suas ações quase exclusivamente nas percepções locais imediatas, reagindo ao ambiente apenas quando o risco já é evidente.

No cenário com comunicação, observa-se o surgimento de um comportamento mais proativo. O agente seguidor tende a: iniciar a troca de faixa com maior antecedência; evitar faixas que foram previamente sinalizadas como perigosas; manter uma trajetória mais contínua, com menos paradas e correções bruscas.

Essa diferença na forma de navegação aparece de forma indireta no gráfico de duração dos episódios, em que o agente com comunicação passa a apresentar tempos mínimos de conclusão inferiores aos do agente sem comunicação. Trajetórias mais fluidas, com menos desacelerações abruptas e desvios tardios, contribuem tanto para a redução do tempo de episódio quanto para a diminuição do número de colisões.

De maneira geral, a análise qualitativa reforça os achados quantitativos: a comunicação entre agentes permite que o seguidor antecipe situações de risco e aja de forma mais coordenada com o líder, resultando em um tráfego simulado mais seguro e eficiente.

4.4 Síntese dos resultados em relação à questão de pesquisa

Os resultados apresentados neste capítulo indicam que a introdução de comunicação entre os veículos autônomos — por meio da troca de mensagens entre o agente líder, o TrafficManager e o agente seguidor — traz benefícios concretos ao desempenho do sistema.

Em termos quantitativos, o cenário com comunicação apresentou: maior estabilidade nas métricas por episódio; recompensas acumuladas superiores na fase final de treinamento; redução da ocorrência de colisões; melhora na duração dos episódios, com tempos mínimos inferiores aos do cenário sem comunicação.

Em termos qualitativos, observou-se que o agente seguidor com comunicação desenvolve um comportamento mais proativo e fluido, enquanto o agente sem comunicação permanece mais reativo, com manobras tardias e maior propensão a paradas bruscas.

Esses resultados dialogam diretamente com a questão de pesquisa formulada na Introdução, sugerindo que, em um cenário de tráfego veicular simulado, a comunicação entre veículos autônomos é capaz de reduzir colisões e melhorar o desempenho do fluxo em comparação com veículos que não compartilham informações sobre o ambiente.

No Capítulo 5, essas evidências serão retomadas para a formulação das conclusões finais, bem como para a discussão das limitações do estudo e proposição de trabalhos futuros que possam ampliar a complexidade do ambiente e a generalidade dos resultados.

5 CONCLUSÕES

Este trabalho investigou o impacto da comunicação entre veículos autônomos em um ambiente de tráfego simulado, utilizando sistemas multiagentes e aprendizado por reforço com Unity e ML-Agents. A partir de um cenário controlado com um veículo líder e um seguidor, compararam-se duas condições: uma sem comunicação e outra em que o seguidor recebia informações adicionais do líder via TrafficManager.

Os resultados mostram que a comunicação V2V melhora o desempenho do tráfego e reduz colisões. No cenário com comunicação, o agente seguidor apresentou maior taxa de

sucesso, menor número de acidentes — especialmente com obstáculos — e trajetórias mais rápidas e fluidas. Observou-se também um comportamento mais proativo, em contraste com o caráter reativo do agente sem comunicação.

Apesar desses resultados positivos, o estudo é limitado por tempo de treinamento reduzido, restrições de hardware e simplicidade do ambiente, que não representa toda a complexidade do tráfego real. Ainda assim, as evidências indicam que a troca de informações entre veículos favorece decisões mais antecipadas, maior segurança e melhor fluxo.

O trabalho oferece uma base sólida para futuras pesquisas, que podem incluir treinamentos mais longos, análise estatística mais robusta, cenários mais complexos, novas estratégias de comunicação e experimentos com algoritmos alternativos de aprendizado por reforço.

Em síntese, conclui-se que a comunicação entre veículos autônomos tem potencial significativo para melhorar a eficiência e a segurança do tráfego, sendo uma abordagem promissora para sistemas de transporte inteligentes.

REFERÊNCIAS

RUSSELL, Stuart J.; NORVIG, Peter. *Artificial Intelligence: A Modern Approach*. 4. ed. Hoboken: Pearson, 2021.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning*. Cambridge, MA: MIT Press, 2016.

MITCHELL, T. *Machine Learning*. New York: McGraw-Hill, 1997.

LECUN, Y.; BENGIO, Y.; HINTON, G. *Deep learning*. Nature, v. 521, 2015.

FERBER, J. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley, 1999.

Wooldridge, Michael. *An Introduction to MultiAgent Systems*. 2.ª ed. John Wiley & Sons, 2009.

SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction*. 2. ed. MIT Press, 2018.

PUTERMAN, M. L. Markov Decision Processes: *Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.

François-Lavet, V.; Henderson, P.; Islam, R.; Bellemare, M. G.; Pineau, J. (2018). *An Introduction to Deep Reinforcement Learning. Foundations and Trends in Machine Learning.*

Mnih, V.; et al. (2015). *Human-level control through deep reinforcement learning.*

Juliani, A.; Berges, V.-P.; Teng, E.; Cohen, A.; Harper, J.; Elion, C.; Goy, C.; Gao, Y.; Henry, H.; Mattar, M.; Lange, D. (2020). *Unity: A general platform for intelligent agents.*

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. (2017). *Proximal Policy Optimization Algorithms.*

Thrun, S. e Montemerlo, M. (2006). *The GraphSLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. International Journal of Robotics Research.*