

UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”

FACULDADE DE ENGENHARIA DE BAURU

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

TRABALHO DE GRADUAÇÃO

ADRIEL BOMBONATO GUIDINI GODINHO

**ESTUDO DE OTIMIZAÇÃO DE ROBÔS MÓVEIS AUTÔNOMOS EM  
AMBIENTE DE SIMULAÇÃO HOSPITALAR COM REINFORCEMENT  
LEARNING**

Bauru, SP  
2024

UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”  
FACULDADE DE ENGENHARIA DE BAURU  
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ADRIEL BOMBONATO GUIDINI GODINHO

**ESTUDO DE OTIMIZAÇÃO DE ROBÔS MÓVEIS AUTÔNOMOS EM AMBIENTE DE  
SIMULAÇÃO HOSPITALAR COM REINFORCEMENT LEARNING**

Monografia apresentada ao Curso de Engenharia Elétrica da Universidade Estadual Paulista “Júlio de Mesquita Filho” como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Elétrica.

**Orientador:** Prof. Dr. Marcelo Nicoletti Franchin

Bauru, SP  
2024

Adriel Bombonato.

Estudo de otimização de robôs móveis autônomos em ambiente de simulação hospitalar com reinforcement learning/ Adriel Bombonato Guidini Godinho. –, 2024-  
56 p. 1 :il. (colors; grafs; tabs).

Orientador: Prof. Dr. Marcelo Nicoletti Franchin

Monografia – Universidade Estadual Paulista “Júlio de Mesquita Filho”,  
Faculdade de Engenharia de Bauru, Departamento de Engenharia Elétrica, 2024.

1. Logística hospitalar. 2. robôs móveis autônomos. 2. aprendizado por reforço. 3. Unity. 4. ML-Agents. I. Prof. Dr. Marcelo Nicoletti Franchin. II. Universidade Estadual Paulista “Júlio de Mesquita Filho”. III. Estudo de otimização de robôs móveis autônomos em ambiente de simulação hospitalar com reinforcement learning.

## FOLHA DE APROVAÇÃO

Aluno: **Adriel Bombonato Guidini Godinho**

Título: **ESTUDO DE OTIMIZAÇÃO DE ROBÔS MÓVEIS AUTÔNOMOS EM AMBIENTE DE SIMULAÇÃO HOSPITALAR COM REINFORCEMENT LEARNING**

Trabalho de Graduação defendido e aprovado em 19 / 06 / 2024, com

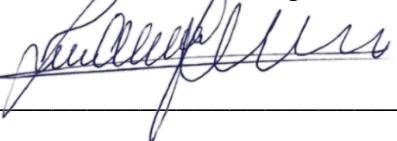
MÉDIA: 9,5 (Nove e Cinco), pela comissão julgadora:



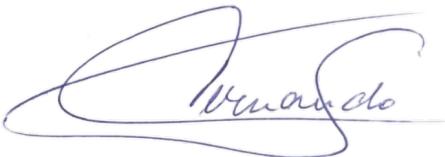
Prof. Dr. Marcelo Nicoletti Franchin



Prof. Dr. José Eduardo Cogo Castanho



Prof. Dr. José Alfredo Covolan Ulson



Prof. Dr. Fernando de Souza Campos

Coordenador do Curso de Graduação em  
Engenharia Elétrica

*Dedico este trabalho aos meus pais e minha família.*

# **Agradecimentos**

Inicialmente, aos meus parentes, Carolina, César, Neiva e Vinícius, por me apoiarem e serem minha principal motivação na vida.

A minha companheira Gláucia e meus amigos por terem me apoiado em momentos difíceis.

Ao meu orientador, Professor Dr. Marcelo Nicoletti Franchin, pelo apoio e por toda a animação junto a mim.

Ao Capítulo Estudantil RAS Unesp Bauru e Ramo Estudantil IEEE de Bauru, pelo acolhimento e pela oportunidade de desenvolvimento.

# Resumo

A crescente demanda por eficiência e segurança nos processos hospitalares tem impulsionado o desenvolvimento de soluções tecnológicas inovadoras. A logística hospitalar, crucial para o funcionamento adequado de instituições de saúde, enfrenta desafios significativos, especialmente na entrega eficiente de instrumentos cirúrgicos em salas de operação. Neste contexto, os Robôs Móveis Autônomos (AMRs) surgem como uma alternativa promissora para automatizar e aperfeiçoar essas tarefas. Este trabalho investiga a utilização de AMRs especializado com técnicas de aprendizado por reforço para a entrega de instrumentos cirúrgicos em hospitais. A pesquisa desenvolve um ambiente de simulação hospitalar em 3D, utilizando a plataforma Unity e a ferramenta ML-Agents, para treinar agentes autônomos em diversas tarefas logísticas. A metodologia abrange a criação de um currículo de aprendizado progressivo, onde os agentes são submetidos a diferentes níveis de complexidade, desde a entrega em uma única sala até a navegação por um hospital completo. Os agentes treinados são capazes de melhorar a eficiência das entregas, reduzindo o tempo de entrega durante o treinamento e evitando obstáculos. Os resultados finais apontam que a aplicação de AMRs com aprendizado por reforço em ambientes hospitalares simulados, com devido refinamento e evolução, podem representar uma solução eficaz para a automação de tarefas logísticas, contribuindo para a melhoria operacional na entrega de instrumentos cirúrgicos.

**Palavras-chave:** logística hospitalar, robôs móveis autônomos, aprendizado por reforço, Unity, ML-Agents.

# Abstract

The growing demand for efficiency and safety in hospital processes has driven the development of innovative technological solutions. Hospital logistics, crucial for the proper functioning of healthcare institutions, face significant challenges, especially in the efficient delivery of surgical instruments to operating rooms. In this context, Autonomous Mobile Robots (AMRs) emerge as a promising alternative to automate and improve these tasks. This work investigates the use of specialized AMRs with reinforcement learning techniques for the delivery of surgical instruments in hospitals. The research develops a 3D hospital simulation, using the Unity platform and the ML-Agents tool, to train autonomous agents in various logistical tasks. The methodology encompasses the creation of a progressive learning curriculum, where agents are subjected to different levels of complexity, from delivery to a single room to navigating an entire hospital. Trained agents are able to improve delivery efficiency, reducing delivery time during training and avoiding obstacles. The final results indicate that the application of AMRs with reinforcement learning in simulated hospital environments, with due refinement and evolution, can represent an effective solution for the automation of logistical tasks, contributing to the improvement of operational efficiency in the delivery of surgical instruments.

**Keywords:** hospital logistics, autonomous mobile robots, reinforcement learning, Unity, ML-Agents.

# Listas de ilustrações

Figura 1.1 – Exemplos de arquitetura de uma sala de operação convencional. . . . .	2
Figura 1.2 – Exemplos de cadeia logística para entrega de instrumentos cirúrgicos. . . . .	3
Figura 2.1 – Tipos de Orientações para AGVs e para AMRs. . . . .	6
Figura 2.2 – Sistema com AGVs centralizados e AMRs descentralizados. . . . .	7
Figura 2.3 – Exemplos de AMRs utilizados para transporte de instrumentos cirúrgicos e medicamentos. . . . .	8
Figura 2.4 – Exemplo de ambiente de desenvolvimento do simulador Unity. . . . .	12
Figura 2.5 – Diagrama de funcionamento típico utilizando ML-Agents para treinamento, inferência e heurística. . . . .	14
Figura 3.1 – Exemplo de associação de uma posição e um tipo a um objeto <i>Room</i> . . . . .	17
Figura 3.2 – Exemplo de template de uma sala de operação. . . . .	20
Figura 3.3 – Template de uma sala de operação convertido em uma sala no ambiente de treinamento. . . . .	21
Figura 3.4 – Modelo utilizado para ponto de entrega, à direita, e modelo utilizado para ponto de coleta, à esquerda, de instrumentos cirúrgicos. . . . .	22
Figura 3.5 – Par atribuído a um ponto de entrega e um ponto de coleta, assim como um agente carregando um instrumento cirúrgico. . . . .	23
Figura 3.6 – Objeto primitivo utilizado para o agente e alguns de seus parâmetros. . . . .	24
Figura 3.7 – Componente raycast atrelado ao agente, com raios vermelhos representando contato com outro objeto. . . . .	26
Figura 3.8 – Pontos flutuantes para um conjunto de observação variável de um ponto de coleta. . . . .	29
Figura 3.9 – Exemplo de uma sala instanciada na primeira lição. . . . .	30
Figura 3.10–Exemplo de uma sala grande instanciada na segunda lição. . . . .	31
Figura 3.11–Exemplo de um ambiente instanciado na terceira lição. . . . .	32
Figura 3.12–Exemplo de um ambiente instanciado na quarta lição. . . . .	33
Figura 3.13–Exemplo de um ambiente instanciado na quinta lição. . . . .	34
Figura 3.14–Exemplo de um ambiente instanciado na sexta lição. . . . .	35
Figura 3.15–Exemplo de um ambiente instanciado na sétima lição. . . . .	36
Figura 3.16–Exemplo de paralelização de ambiente para a segunda lição. . . . .	37
Figura 4.1 – Gráfico de recompensa acumulada para o grupo durante a primeira lição. . .	38
Figura 4.2 – Gráfico de recompensa acumulada para o grupo durante a segunda lição. .	39
Figura 4.3 – Gráfico da quantidade média de passos para se completar a primeira lição. .	39
Figura 4.4 – Gráfico da quantidade média de passos para se completar a segunda lição. .	40

Figura 4.5 – Gráfico de recompensa acumulada para o grupo durante a terceira lição. . . .	41
Figura 4.6 – Gráfico da quantidade média de passos para se completar a terceira lição. . . .	41
Figura 4.7 – Gráfico de recompensa acumulada para o grupo durante a quarta lição. . . .	42
Figura 4.8 – Gráfico da quantidade média de passos para se completar a quarta lição. . . .	42
Figura 4.9 – Gráfico de recompensa acumulada para o grupo durante a quinta lição. . . .	43
Figura 4.10–Gráfico da quantidade média de passos para se completar a quinta lição. . . .	43
Figura 4.11–Gráfico de recompensa acumulada para o grupo durante a sexta lição. . . . .	44
Figura 4.12–Gráfico da quantidade média de passos para se completar a sexta lição. . . .	44
Figura 4.13–Gráfico de recompensa acumulada para o grupo durante a quinta lição. . . .	45
Figura 4.14–Gráfico da quantidade média de passos para se completar a quinta lição. . . .	45
Figura 4.15–Gráfico de recompensa acumulada para o grupo em todas as lições. . . . .	46
Figura 4.16–Gráfico da quantidade média de passos para se completar cada lição. . . . .	47

# **Lista de tabelas**

Tabela 3.1 – Relação entre tipo e sala hospitalar equivalente. . . . .	18
Tabela 3.2 – Valores dos principais parâmetros para cada lição. . . . .	36

# **Lista de abreviaturas e siglas**

A*	A-estrela
AGV	Automated Guided Vehicle
AMR	Autonomous Mobile Robot
CS	Central Storage
D*	D-estrela
IA	Inteligência Artificial
MA-POCA	Multi-Agent Posthumous Credit Assignment
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
SLAM	Simultaneous Localization and Mapping
USD	Universal Scene Description

# Sumário

<b>1</b>	<b>Introdução e justificativa</b>	<b>1</b>
1.1	Objetivos	5
<b>2</b>	<b>Revisão Bibliográfica</b>	<b>6</b>
2.1	AMRs em hospitais	6
2.2	Otimização de métricas com aprendizado de máquina por reforço	9
2.3	Ambientes de simulação para sistemas multiagente	10
2.4	Unity e ML-Agents para MARL	12
2.4.1	Funcionamento do ML-Agents	12
2.4.2	Parâmetros de treinamento	14
<b>3</b>	<b>Metodologia</b>	<b>16</b>
3.1	Equipamentos	16
3.2	Ambiente de simulação	16
3.3	Construção e Operação do Ambiente de Treino	16
3.3.1	Criação do esboço do mapa	16
3.3.2	Instanciação de elementos 3D	19
3.3.3	Seleção e Utilização de pontos de entrega e coleta de instrumentos cirúrgicos	21
3.4	Construção e Operação do Agente	23
3.4.1	Ações do Agente	24
3.4.2	Recompensas do Agente	24
3.4.3	Observações do Agente	25
3.5	Etapas do Curriculum Learning	29
3.5.1	Primeira Lição (Sala Única)	30
3.5.2	Segunda Lição (Grande Sala Única)	31
3.5.3	Terceira Lição (Duas Salas)	32
3.5.4	Quarta Lição (Três Salas)	32
3.5.5	Quinta Lição (Hospital Pequeno)	33
3.5.6	Sexta Lição (Hospital Médio)	34
3.5.7	Sétima Lição (Hospital Grande)	35
3.6	Configuração para treinamento	36
<b>4</b>	<b>Resultados</b>	<b>38</b>
4.1	Resultados de Cada lição	38
4.1.1	Resultados da Primeira e Segunda Lição	38
4.1.2	Resultados da Terceira Lição	40
4.1.3	Resultados da Quarta Lição	41

4.1.4	Resultados da Quinta Lição . . . . .	42
4.1.5	Resultados da Sexta Lição . . . . .	43
4.1.6	Resultados da Sétima Lição . . . . .	45
4.2	Observações abandonadas . . . . .	47
4.3	Comportamentos Emergentes . . . . .	48
4.4	Discussão . . . . .	50
<b>Referências</b>	. . . . .	<b>53</b>

# 1 Introdução e justificativa

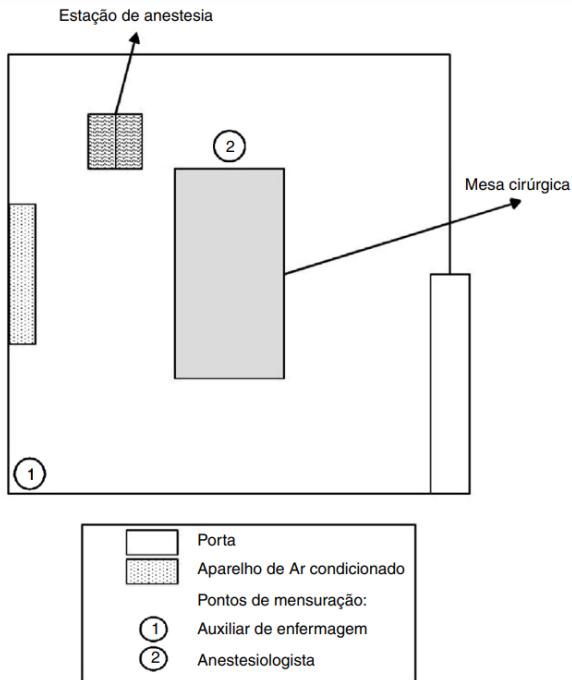
A Constituição Brasileira assegura a todos o direito à saúde (BRASIL, 1988). Porém, para que sua completa implementação, dentro do contexto Brasileiro, é necessário que haja um sistema que englobe a prevenção de doenças, promoção, assistência e a recuperação da saúde populacional.

Essa junção de deveres torna a estrutura dos sistema de saúde complexa, custosa e desafiadora. De forma que é necessário que haja constante adaptação à novas tecnologias e mudanças no ambiente hospitalar(GOMES; ROMÃO; CARVALHO, 2016). A pesquisa, a avaliação e a criação de inovações tecnológicas se tornam a principal ferramenta de resistência em meio a constantes alterações nos cenários ambientais, socioculturais e econômicos (BITTAR; MENDES, 2019). Assim, existe uma grande reflexão para os produtos e serviços hospitalares, cuja qualidade é essencial.

Conforme as atividades exercidas, Raimundo, Dias e Guerra (2015) consideram que os materiais, recursos humanos, administração financeira e em especial a logística de suprimentos são os fatores críticos para o desenvolvimento de atividades de atenção à saúde e para a excelência operacional de uma organização hospitalar.

Dentro de logística de suprimentos, é possível citar em específico as salas de operações, que são responsáveis por aproximadamente 60% dos custos hospitalares (WEISS et al., 2016). A figura 1.1 mostra um exemplo de arquitetura de uma sala de operação convencional.

Figura 1.1 – Exemplos de arquitetura de uma sala de operação convencional.

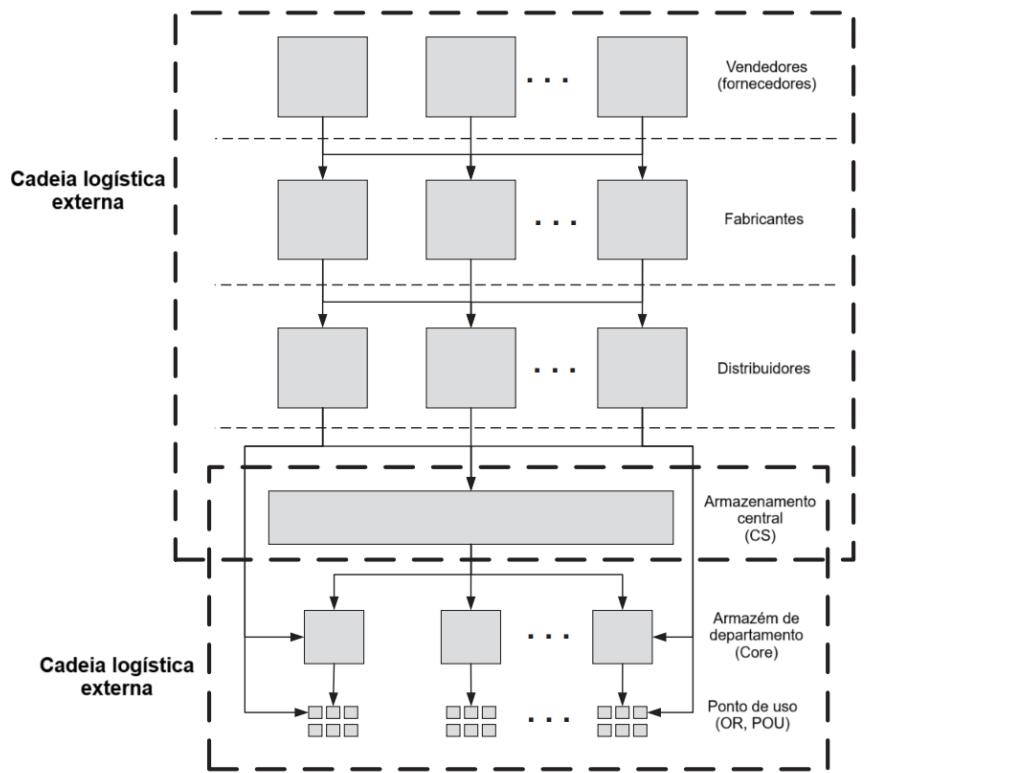


Fonte: Adaptado de Braz et al. (2017)

As salas de operações são espaços que desempenham papel fundamental para a realização de intervenções cirúrgicas de diversos tipos. Para isso, precisam possuir, em demanda, diferentes equipamentos de monitoramento, instrumentos cirúrgicos, medicamentos e quaisquer outras necessidades das equipes cirúrgicas. Essas equipes também devem ser qualificadas e geralmente incluem cirurgiões, anestesiologistas e enfermeiros. Todos trabalham juntos para garantir que os procedimentos cirúrgicos sejam realizados com segurança e eficácia.

A logística de materiais dentro das salas de operação requerem um fluxo constante de suprimentos médicos e cirúrgicos. O início do fluxo acontece nas armazéns centrais, no qual serão obtidos quaisquer materiais necessários. Logo em seguida, eles são direcionados para pontos de uso, como as salas de operação, e, após finalizada a cirurgia, são encaminhados para o departamento de processamento de instrumentos estéreis, onde será realizada a limpeza e esterilizados de instrumentos passivos de reutilização (AHMADI et al., 2019). Dependendo do tamanho do hospital, a cadeia pode ser expandida com a inclusão de armazéns de departamentos, que ficam entre os armazéns centrais e os pontos de uso. Para o trabalho atual, foi considerado somente a utilização de um armazém central, salas de operação como pontos de uso e um departamento de processamento de instrumentos estéreis. A figura 1.2 mostra a cadeia logística formada na entrega de instrumentos cirúrgicos.

Figura 1.2 – Exemplos de cadeia logística para entrega de instrumentos cirúrgicos.



Fonte: Adaptado de Ahmadi et al. (2019)

Os materiais são normalmente transportados pelos enfermeiros e podem incluir desde instrumentos cirúrgicos e equipamentos de monitoramento até medicamentos e materiais descartáveis. Uma boa gestão desses recursos deve sempre garantir que a sala de operação esteja pronta para o próximo procedimento. Para isso, seria benéfico a redução do tempo gasto em atividades de deslocamento, visto que, em uma visão geral, os enfermeiros dedicam grande parte do seu tempo (cerca de 40%) a tarefas que não estão diretamente relacionadas à enfermagem, como transporte e entrega de suprimentos, serviços de limpeza e outras atividades não relacionadas à enfermagem (FRAGAPANE et al., 2020).

Os Robôs Móveis Autônomos, no inglês referidos como *Mobile Autonomous Robots (AMRs)*, têm se tornado cada vez mais úteis para cenários de transporte de suprimentos e desinfecção de salas, isso devido à sua capacidade de evitar obstáculos, localização dinâmica de caminhos e menores dimensões dos veículos. Os AMRs podem ser implementados em ambientes hospitalares movimentados, como áreas com pacientes presentes e corredores e portas estreitas, levando a um maior grau de integração em hospitais (FRAGAPANE et al., 2023).

Com isso, é possível a utilização de AMRs em entregas e, consequentemente, no gerenciamento de inventário de suprimentos cirúrgicos e instrumentos estéreis. Acelerando, assim, o

processo de entrega desses materiais e economizando tempo de trabalho de enfermeiros gasto com transporte. Sempre levando em consideração todos tipos de itens e suas quantidades que o cirurgião requer.

A navegação do AMR é baseada em três principais problemas, a localização, o planejamento de rota e o controle de movimento. O planejamento de rota, em específico, tem grande importância por permitir a identificação e seleção dos melhores caminhos para o robô realizar dentro do ambiente (SARIFF; BUNIYAMIN, 2006).

A implementação do planejamento pode ser atingido tanto pelo uso de algoritmos clássicos como pela utilização de algoritmos de inteligência artificial. Dentro de algoritmos clássicos, uma das formas mais utilizadas e a realização de um mapa probabilístico de caminhos seguida do uso de algoritmos básicos de pesquisa de grafos, como A\* (pronunciado como "A-estrela" ou "A-star" em inglês) e D\* (pronunciado como "D-estrela" ou "D-star" em inglês). Para algoritmos de inteligência artificial, uma das principais formas é a utilização de redes neurais, com destaque para utilização de *Reinforcement Learning* (RL), um método de aprendizado por reforço que conta com a capacidade de aprender estratégias por interações autônomas. Tal característica é muito útil para ambientes dinâmicos (DONG; ZOU, 2020). Duguleana e Mogan (2016) demonstram a utilização de um algoritmo de RL, o algoritmo Q-learning, para solução de planejamento de rotas em ambientes incertos aos agentes e Dong e Zou (2020) aprimoraram o planejamento através do algoritmo de gradiente de política determinística profunda, normalmente referido pela sigla DDPG.

Tomando-se em conta as diferentes abordagens para otimização de entrega e recolhimento de instrumentos cirúrgicos em salas de operações, esse trabalho busca avaliar a eficácia do uso de AMRs acoplados por um sistema de planejamento de rota totalmente fornecido por uma rede neural artificial treinada a partir do uso de *Multi-Agent Posthumous Credit Assignment (MA-POCA)* (COHEN et al., 2021), um algoritmo de RL adaptado para cenários multiagentes. Seu uso mescla a utilização de mecanismos de atenção (VASWANI et al., 2017) com a utilização de uma arquitetura ator-crítico (LYU et al., 2021). Utiliza-se também um sistema de aprendizado progressivo para os agentes, chamado de aprendizado por curriculum. Com seu uso, o progresso do agente é feita de forma incremental e em diferentes tarefas, o que evita a dificuldade de treinar para adquirir comportamentos complexos de uma só vez (NARVEKAR et al., 2020). A palavra curriculum se refere ao conjunto de tarefas.

O método de avaliação foi baseado em treinamentos e testes dentro de um ambiente simulado na plataforma Unity (UNITY, 2024). Um ambiente simulado oferece total segurança durante o aprendizado dos agentes, que são os AMRs neste trabalho, principalmente pelo fato de colidirem muito com outros obstáculos no começo no treinamento. Além disso, a plataforma Unity possui uma biblioteca própria chamada de ML-Agents que permite acesso e uso facilitado

dos principais algoritmos de cenários de RL (UNITY, 2024).

Conforme a estrutura da monografia, a seção 1.1 apresenta os principais objetivos do trabalho, a seção 2 possui uma revisão bibliográfica do assunto, que trata dos principais usos de AMRs em hospitais (seção 2.1), da utilização de RL para construção de agentes inteligentes (seção 2.2), dos tipos de simulação para sistemas multiagentes (seção 2.3) e do funcionamento da plataforma Unity integrada com sua biblioteca de aprendizado de máquina, o ML-Agents (seção 2.4). A seção 3 demonstra como foi o processo de construção do ambiente de simulação e do agente (seções 3.3 e 3.4, respectivamente), da formação do currículum de aprendizagem (seção 3.5) e do treinamento dos agentes (seção 3.6). A seção 4 demonstra os dados obtidos para cada lição no currículum (seção 4.1) e os comportamentos emergentes obtidos (seção 4.3).

## 1.1 Objetivos

A partir do desafio de entrega de suprimentos cirúrgicos e instrumentos estéreis em ambientes hospitalares e da utilização de AMRs com trajetória otimizada por algoritmos de RL, este trabalho tem como principal objetivo testar a viabilidade de entregas feitas com AMRs treinados a partir de algoritmo de RL dentro de um ambiente simulado através da plataforma Unity. Ademais, criar um sistema de geração de hospitais que reduza a chance do agente desenvolver viés conforme a planta hospitalar e o posicionamento de pontos de coleta e de entrega de instrumentos cirúrgicos.

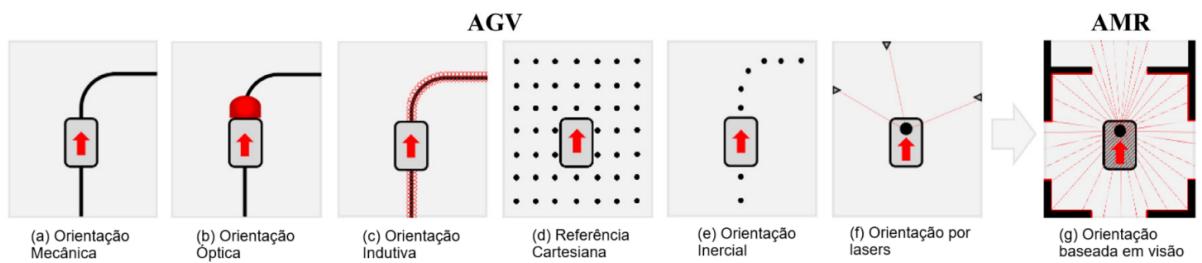
Por fim, também tomou-se como objetivo disponibilizar o código utilizado de forma *open-source* em um repositório no github. O que foi feito através da seguinte url: <<https://github.com/Adribom/TG-RL-Hospital>>

## 2 Revisão Bibliográfica

### 2.1 AMRs em hospitais

O uso de veículos guiados automaticamente, em inglês referidos como *Automated Guided Vehicles (AGVs)*, desde 1995, tem sido uma forma de facilitar a operações intralogísticas de diversos setores, como em armazéns e em linhas de produção (MULLER, 1983). O sistema de orientação que forma a parte central dos sistemas de manuseio de materiais dos AGVs evoluiu ao longo de vários estágios de orientação mecânica, óptica, indutiva, cartesiana, inercial e a laser (figura 2.1 (a)-(f)) para a orientação baseada na visão (figura 2.1 (g)). Este sistema baseado em visão usa sensores onipresentes, poderosos computadores de bordo, inteligência artificial (IA) e tecnologia de localização e mapeamento simultâneos, referido no inglês como *Simultaneous Localization and Mapping (SLAM)*, permitindo que o dispositivo entenda seu ambiente operacional e navegue em instalações sem a necessidade de definir e implementar pontos de referência com antecedência (FRAGAPANE et al., 2021).

Figura 2.1 – Tipos de Orientações para AGVs e para AMRs.



Fonte: Adaptado de Fragapane et al. (2021)

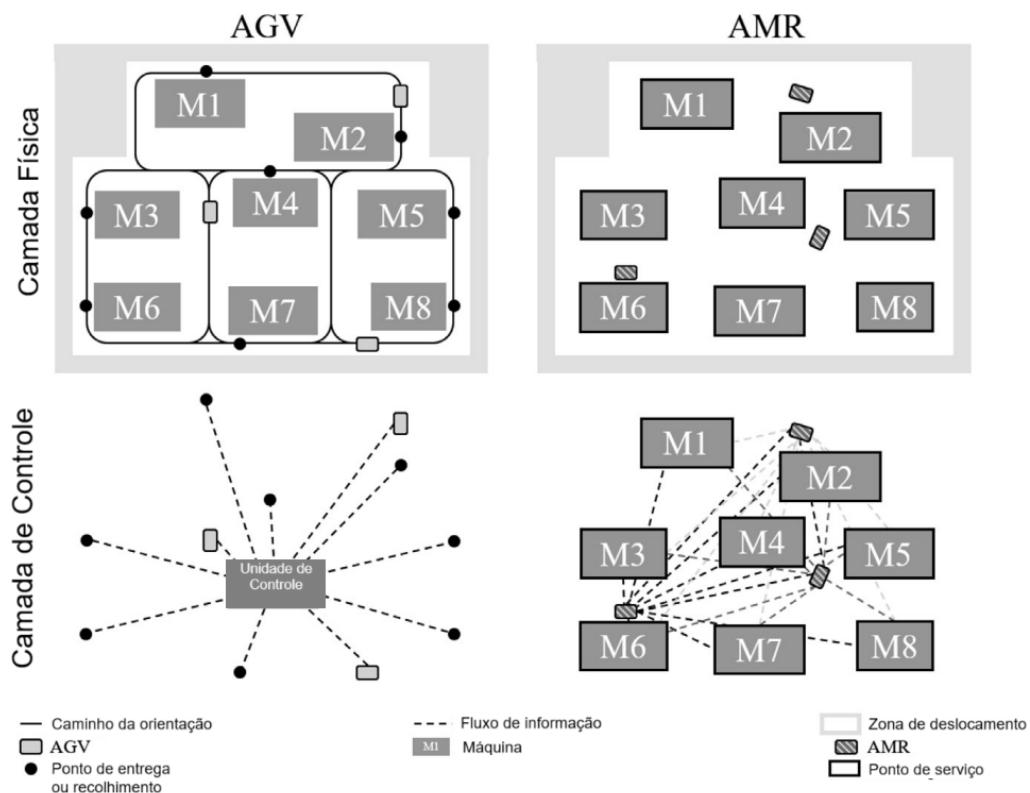
Em casos de pequenas mudança de posicionamento dos robôs ou de alterações na estrutura de seu local de operação, a recalibração do sistema de orientação de AGVs pode tomar um tempo considerável, fomentando períodos de inatividade e, consequentemente, perda de eficiência na logística atribuída a eles (FRAGAPANE et al., 2021). Os AMRs conseguem evitar essas perdas devido a sua principal mudança em referência aos AGVs, a capacidade de tomada de decisões autônomas. Essa habilidade permite que os AMRs tenham a capacidade de continuamente tomar decisões sobre como se comportar em um ambiente variável, mas ainda assim respeitando limitações impostas durante sua programação.

Além disso, AGVs não conseguem evitar obstáculos ou entrar em enfermarias e departamentos devido a segurança e preocupações de espaço. Sistemas de coleção de vácuo automáticos,

sistemas de tubos pneumáticos e sistemas de transporte suspenso normalmente possuem apenas pontos de coleta e entrega fixos em departamentos. Essa pequena flexibilidade na mobilidade e dependência com ajuda humana faz com seja difícil automatizar o espaço entre os departamentos e os pacientes (FRAGAPANE et al., 2020), o que reforça a flexibilidade oferecida pelos AMRs.

Com isso, AMRs também ganham maior liberdade de operação descentralizada. As estruturas de controle centralizado estão profundamente enraizadas na indústria e podem acessar informações globais para obter desempenho de objetivo único ideal para sistemas simples de pequena escala. O controle descentralizado normalmente tem acesso apenas a informações geradas internamente, como dados de seus próprios sensores e equipamentos, e encontra soluções dentro desse contexto local. No entanto, em sistemas complexos de grande escala, o uso de abordagens descentralizadas pode ser benéfico (RYCK; VERSTEYHE; DEBROUWERE, 2020). O sistema descentralizado na tomada de decisões dos AMRs permite que eles reajam dinamicamente a demandas ou mudanças e que consigam constantemente se optimizar. A figura 2.2 demonstra os diferentes fluxos de informações para os casos descentralizado e centralizado.

Figura 2.2 – Sistema com AGVs centralizados e AMRs descentralizados.



Fonte: Adaptado de Fragapane et al. (2021)

Para cenários de multiagentes, porém, existem técnicas de RL que utilizam um mesclado

entre sistemas centralizados e descentralizados. Para isso, é utilizado o método descentralizado para execução e centralizado para treinamento (LOWE et al., 2017). Com isso, os agentes utilizam apenas observações locais, mas o treinamento em si acontece com toda informação globalmente disponível. Cohen et al. (2021) utilizam da mescla de sistemas centralizados e descentralizados para a produção do MA-POCA, o que permite que haja adição e remoção de agentes durante o treinamento, sem que a experiência do agente removido seja perdida em sua próxima atualização de parâmetros. Para o caso de AMRs em hospitais, isso ajuda em casos onde há acidentes ou mal funcionamento de componentes que danifiquem o robô ou em casos onde há adição de novos agentes.

Com esses diferentes atributos e vantagens de AMRs, dentro de um ambiente hospitalar, podemos encontrá-los atuando em tele-atendimentos, desinfecção de salas e, principalmente, transporte de diferentes tipos de objetos pelos cômodos e corredores, como é mostrado nos exemplos da figura 2.3.

Figura 2.3 – Exemplos de AMRs utilizados para transporte de instrumentos cirúrgicos e medicamentos.



Fonte: Adaptado de Fragapane et al. (2020)

Para a logística no transporte de instrumentos cirúrgicos e estéreis, deve ser considerado que tais materiais são necessários para operações, tratamentos e cirurgias planejadas ou emergenciais. Para possibilitar suas entregas em demanda e manter um menor custo de operação, é circulado um quantidade específica de carga durante o dia. Isso acontece entre pontos de uso, departamentos, almoxarifados de departamentos e centrais e armazenamentos centrais (CHOBIN; SWANSON, 2012). O local de retorno de equipamentos reutilizáveis é o departamento de esterilização, onde é garantido que os instrumentos são de alta qualidade, estéreis e disponíveis. Porém, muitos hospitais ainda têm dificuldade na logística de transporte desses materiais. Wubben

et al. (2010) cita que 46% das causas de atrasos em salas de operação são relacionadas a não disponibilidade de instrumentos estéreis.

AMRs podem ser considerados uma alternativa adequada a formas atuais de automatização de hospitais quanto a transporte de instrumentos cirúrgicos e estéreis, fornecendo transporte altamente flexível e econômico (FRAGAPANE et al., 2023). Mesmo assim, alguns aspectos ainda podem ser melhorados quanto a essa abordagem. Fragapane et al. (2020) cita alguns fatores passivos de melhoria com futuros trabalhos. Principalmente, como instrumentos devem ser transportados pelos AMR para alcançar maior robustez, produtividade, segurança e qualidade, como deve o planejamento de caminho e de movimentação ser adaptado para o ambiente hospitalar e como determinar o número ótimo de AMRs dentro da logística hospitalar.

## 2.2 Otimização de métricas com aprendizado de máquina por reforço

Aprendizado de máquina por reforço se trata em mapear situações e ações para se maximizar um sinal numérico de recompensa. O aprendizado não se trata de mostrar ações ótimas para o agente, mas sim deixá-lo decidir, por tentativa e erro, quais ações se desdobram no maior acúmulo de recompensa (SUTTON; BARTO, 2018). Tudo aquilo que não seja o próprio agente é considerado como ambiente. Essa definição deve ser seguida de forma literal, pois até em casos em que há vários agentes presentes, um agente considera outros agentes que não seja ele mesmo como parte do ambiente.

O agente, portanto, tem o papel de observar o ambiente e formular ações que irão alterá-lo no caminho de acúmulo máximo de recompensa. As regras utilizadas pelo agente para gerar o mapeamento de observações para ações é chamado de política. A política pode ser considerada o centro de um agente em um ambiente de aprendizado por reforço, pois ela sozinha já é capaz de determinar comportamentos (SUTTON; BARTO, 2018).

Romero-Martí et al. (2016) utilizam a formulação de ações através de um algoritmo de RL, Q-learning, para que um robô Roomba consiga determinar por conta própria uma política que englobe os possíveis caminhos de cada estado para cada estado objetivo. Nesse caso, o Roomba consegue se locomover de um ponto de uma sala até outro ponto desejado através de uma representação topológica de mapa. As recompensas do sistema são baseadas na transição de salas feitas pelo robô, porém, para um caso hospitalar com ARMs, essas recompensas podem ser expandidas para englobar aspectos como tempo de entrega, tempo em que o AMR está ativo, distância percorrida e eficiência energética.

Balachandran, Lal e Sreedharan (2022) providenciam um AMR para navegar em um

ambiente de armazém para seu local de destino usando sensores LIDAR. O método usado para resolver este problema é um algoritmo de aprendizado por reforço profundo (DRL) chamado deep Q-network (DQN) para detectar e evitar obstáculos e alcançar o local de destino. Escobar-Naranjo et al. (2023) também utilizam a mesma abordagem para relacionar esse algoritmo com uma simulação em Gazebo.

Ainda com RL, Yu et al. (2023) implementam um sistema com 2 agentes, com objetivo de explorarem um ambiente desconhecido para o mais rápido possível. Para isso, se baseiam no algoritmo *Proximal Policy Optimization (PPO)*, que garante que a política atualizada permaneça próxima da política original, evitando mudanças drásticas que possam impactar negativamente o processo de aprendizagem (SCHULMAN et al., 2017). Esse algoritmo foi adaptado para cenários multiagentes, além de ter sido somado com uma técnica de elaboração de recompensas em paralelo de forma assíncrona, para que um agente não precise esperar outro terminar sua ação para gerar um sinal de recompensa. Este exemplo também demonstra a complexidade de se realizar tais treinamentos em um cenário na vida real, devido a fatores como a natureza assíncrona dos agentes, que operarem em diferentes velocidades ou podem apresentar problemas de conexão, a necessidade de protocolos de comunicação efetivos e a elaboração de técnicas de aleatorização de ambiente.

## 2.3 Ambientes de simulação para sistemas multiagente

Ambientes de simulação para sistemas com um agente estão ficando bem mais numerosos e acessíveis. Porém, cada um possui uma particularidade e diferentes aplicações podem se beneficiar de diferentes tipos de simuladores. Juliani et al. (2018) dividem os principais atributos de um bom simulador, para casos de aprendizado de máquina, em duas principais categorias: propriedades de ambiente, com ambiente referenciando o espaço em que o agente age, e propriedades de simulação, com simulador sendo a plataforma que gera o ambiente. Esses atributos são mais requeridos conforme o aumento da complexidade do comportamento desejado dos agentes e podem ser resumidos nos seguintes pontos:

- Propriedades de ambiente
  - **Complexidade sensorial:** trata como o ambiente interpreta informações sensoriais que representem visão, audição, processamento textual, e mais. quanto maiores as capacidade de percepção sensorial do ambiente simulado, mais fácil é adequar a simulação para vida real;
  - **Complexidade física:** como os agentes interagem dinamicamente com o ambiente em que são colocados é necessária a presença de simuladores que gerem ambientes

que repliquem propriedades físicas do mundo real. O realismo é vital para transferir as políticas aprendidas para o mundo real;

- **Complexidade da lógica de tarefas:** considera a complexidade do espaço de procura que o agente detém para decidir uma ação;
- **Complexidade social:** capacidade de interação entre agentes, facilitando a criação e estudo de comportamentos sociais e comunicação.

- Propriedades de simulação

- **Flexibilidade de controle:** para um design efetivo e rápido de ambientes experimentais, os pesquisadores devem ter controle fácil sobre os ambientes simulados;
- **Execução rápida e distribuída:** para apoiar a rápida iteração necessária na pesquisa experimental, os simuladores precisam operar de maneira fluida, rápida e distribuída.

Considerando esses atributos, Oroojlooy e Hajinezhad (2022) apresentam vários simuladores e suas diferentes aplicações, dentre eles, é possível salientar **MuJoCo**, um mecanismo de física gratuito e de código aberto para pesquisas em robótica (DEEPMIND, 2021), **ROS** em conjunto com **Gazebo**, que permitem a criação e simulação de robôs em um ambiente 3D como forma de validar sistemas robóticos antes de implantá-los fisicamente (ROBOTICS, 2024), **OpenAI Gym**, que une aspectos do MuJoCo para simulação de ambiente com RL (BROCKMAN et al., 2016) e **Unity** com ML-Agents (UNITY, 2022). Além desses simuladores, **NVIDIA Isaac Sim** oferece um ambiente para criação de robôs baseados em IA e autônomos, contando também com a importação de projetos de vários formatos, por conta da utilização do *Universal Scene Description (USD)* (NVIDIA, 2024).

Porém, ao se passar para casos com mais de um agente, existe um número menor de ambientes de simulação adequados. Tomando ainda em conta que cada ambiente pode ter um tipo de mecanismo diferente para comunicação entre seus agentes. Alguns dos ambientes citados anteriormente, como o OpenAI Gym, podem ser expandidos para comportar cenários multiagentes (OROOJLOOY; HAJINEZHAD, 2022).

Juliani et al. (2018) Também fornecem classificação para alguns simuladores, que são divididos entre ambiente único, que é específico para um estudo apenas, conjunto de ambientes, normalmente conjuntos compactados que são usados para *benchmarks*, plataforma específica de domínio, que permitem criação de ambientes para domínios específicos como locomoção e primeira pessoa e, por fim, plataforma geral, que permite a criação dos ambientes citados nas categorias passadas. O simulador Unity com ML-Agents é classificado como uma opção que consegue suportar aplicações multiagentes e, além disso, permite criar qualquer tipo de ambiente

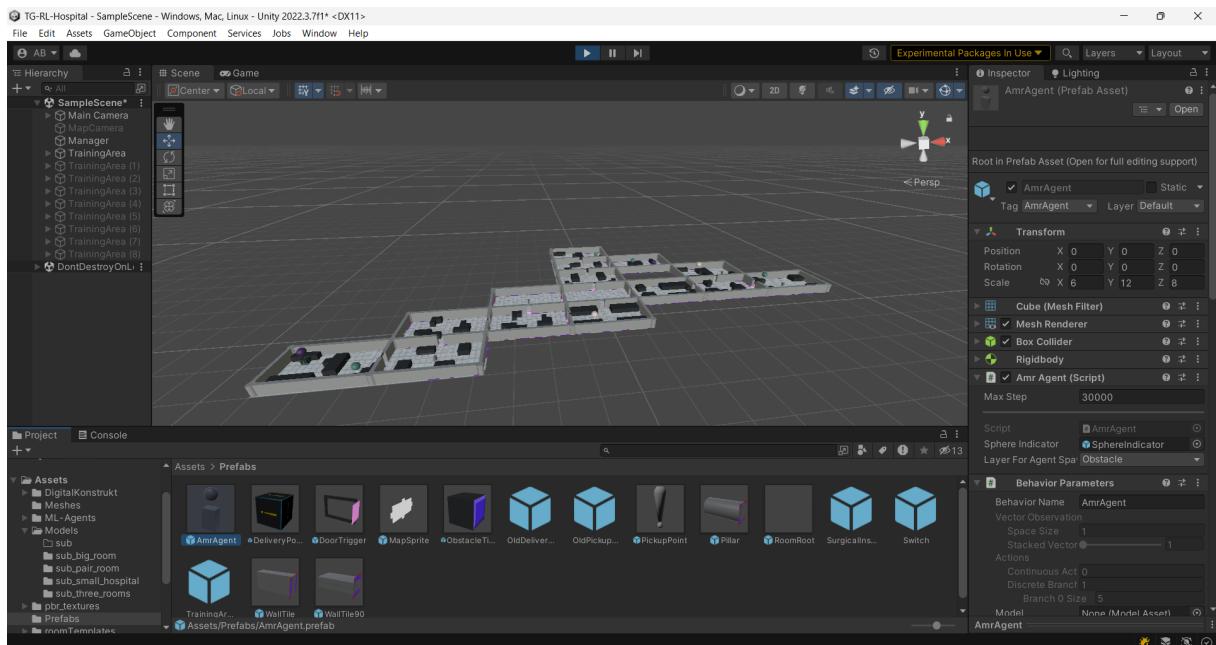
multiagente, sendo ele cooperativo ou competitivo (OROOJLOOY; HAJINEZHAD, 2022). Tanto o Unity como ML-Agents são discutidos na seção 2.4.

## 2.4 Unity e ML-Agents para MARL

### 2.4.1 Funcionamento do ML-Agents

O Unity é uma plataforma para desenvolvimento de jogos, que podem ser simples grid-world ou jogos estratégicos complexos com múltiplos agentes. Qualquer configuração de ambiente pode ser usado para treinar agentes de aprendizado de máquina (JULIANI et al., 2018). A figura 2.4 demonstra um exemplo de ambiente de desenvolvimento no simulador Unity. Ademais, o Unity pode fornecer, para casos de aprendizado de navegação sem mapa, treinamentos mais rápidos que outros ambiente de simulação, como Gazebo (MARCHESINI; FARINELLI, 2020). Essa plataforma, portanto, se torna uma das mais adequadas para simulações de cenários multiagentes baseados em algoritmos de RL.

Figura 2.4 – Exemplo de ambiente de desenvolvimento do simulador Unity.



Fonte: Elaborado pelo próprio autor.

O ML-Agents, por sua vez, pode ser dividido em cinco principais partes:

- **Ambiente de aprendizado:** possui a cena, local onde é instanciado o componentes 3D do ambiente e onde o agente atua e observa;

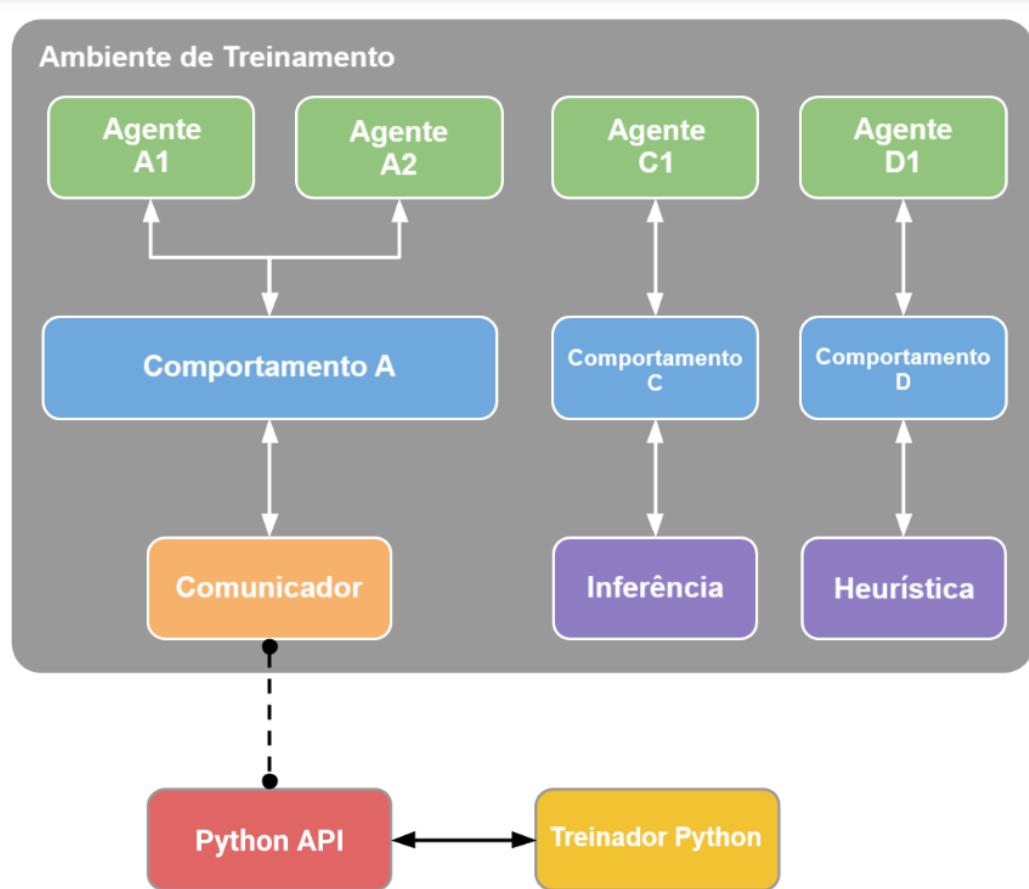
- **API Python de baixo nível:** interface de interação entre códigos Python e o ambiente de aprendizado no Unity. Esta API não faz parte do Unity, mas sim do ML-Agents, então devem fazer sua conexão através de um comunicador próprio do ML-Agents que é instalado dentro do ambiente Unity;
- **Comunicador externo ao Unity:** Fica situado no ambiente de aprendizado a partir de um pacote do ML-Agents instalado dentro do Unity e interliga o ambiente com a API Python;
- **Treinadores Python:** Contempla todos os algoritmos e códigos de aprendizado de máquina para treinamento dos agentes. Para acessá-lo, deve-se baixar um pacote próprio do ML-Agents na máquina através de um terminal. Se comunicam apenas com a API Python;
- **Wrappers:** APIs para comunicação com outros simuladores, como Gym e PettingZoo.

Para gerenciamento de treinamentos e tomada de decisões dos agentes, o ML-Agents se baseia em uma classe chamada *Academy*, que é inicializada ao ser chamada no começo de cada seção de treino. Ela que realiza a conexão com o processo de treinamento Python, através do comunicador externo.

Ao se construir um ambiente, deve-se associar dois principais componentes em um objeto, no Unity chamado de *GameObject*, para habilitar a conexão com o comunicador. O primeiro deles é o componente *Agents*, que cuida da geração de observações, de executar ações e da atribuição de recompensas. O segundo é o componente *Behavior*, que define outros atributos para o agente. Ele receberá observações e recompensas do agente e retornará ações.

A associação desses componentes é necessária para qualquer tipo de treinamento que seja realizado com ML-Agents. Porém, além do treinamento, a biblioteca oferece opções de inferência, dado que seja colocado um modelo de referência no agente, e heurísticas, para testar com comandos manuais as possíveis ações que o agente pode tomar. A figura 2.5 demonstra essas possibilidades e como elas se encaixam com os componentes e partes mencionados.

Figura 2.5 – Diagrama de funcionamento típico utilizando ML-Agents para treinamento, inferência e heurística.



Fonte: Adaptado de Unity (2022)

Vongbunyong et al. (2020) utiliza a plataforma Unity com ML-Agents para uma simulação de AMR na entrega de comida e medicamentos a pacientes e cita como melhoria futura o modelamento realístico de recarga e consumo de bateria.

## 2.4.2 Parâmetros de treinamento

Para inicialização dos parâmetros da rede e de treinamento, o ML-Agents utiliza um arquivo em formato *yaml* onde é associado todos os valores, além de realizar a especificação do processo de curriculum. Um arquivo no formato *yaml* organiza dados de forma hierárquica usando indentação, com o objetivo de torná-los legíveis a humanos. Os principais parâmetros de rede e de treinamento são:

- **num\_layers:** número de camadas escondidas na rede neural. Mais camadas se tornam necessárias quanto mais complexo é o comportamento desejado;

- **hidden\_units**: número de unidades em cada camada escondida. Assim como o número de camadas, deve ser aumentada quanto mais complexo o comportamento desejado;
- **max\_steps**: número total de passos, observação seguida de ação, que o agente realizará;
- **learning\_rate**: hiperparâmetro de taxa de aprendizado da descida do gradiente;
- **beta**: regula a força de regularização de entropia. Quanto maior seu valor, mais randômico será a política, facilitando a exploração do agente pelo cenário.
- **epsilon**: determina o quão rápido a política pode atualizar. Um valor baixo de epsilon permite atualizações mais estáveis na política, porém reduz a velocidade do processo de treinamento.
- **learning\_rate\_schedule**: taxa de redução do *learning\_rate*. Existem duas opções, constante, que mantém o valor constante em todo o treinamento, e linear, que reduz o valor linearmente a partir do valor de *max\_steps*. Existem também a taxas para os valores de *beta* e *epsilon*, porém, utilizam o mesmo tipo que o *learning\_rate\_schedule* quando não são especificados diretamente.

# 3 Metodologia

O capítulo de metodologia do trabalho apresentará o que foi utilizado e desenvolvido para a realização do trabalho, incluindo equipamentos, simulador para construção do ambiente de treinamento, método de operação do ambiente e do agente, algoritmo de treinamento utilizado, e etapas para formação de currículum. Os resultados obtidos através da metodologia serão apresentados na seção 4.

## 3.1 Equipamentos

A realização do projeto contou com a utilização de um notebook pessoal com processador Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 16,0 GB de RAM, placa de vídeo NVIDIA GeForce GTX 1660 Ti e sistema operacional Windows 11.

## 3.2 Ambiente de simulação

O simulador base escolhido para suportar o ambiente e os treinamentos foi o Unity unido com ML-Agents, devido a sua capacidade de suportar ambientes multiagentes, de gerar propriedades de ambiente complexos e de permitir flexibilidade para criação do ambiente hospitalar além de velocidade para treinamento. Tais como apresentadas na seção 2.3.

A versão do Unity utilizado foi a 2022.3.7f1 e, para o ML-Agents, foi utilizado a versão 21, disponibilizada em 9 de outubro de 2023. O treinador Python foi instalado a partir da biblioteca *mlagents* de versão 1.0.0 disponibilizada pelo Pypi.

## 3.3 Construção e Operação do Ambiente de Treino

O ambiente de treinamento consiste em diferentes combinações de diferentes tipos de salas hospitalares, posicionadas através de um processo que permite aleatorização a cada iteração de treinamento sem que se perda a característica hospitalar desejada. O processo é descrito em mais detalhes nas seções 3.3.1

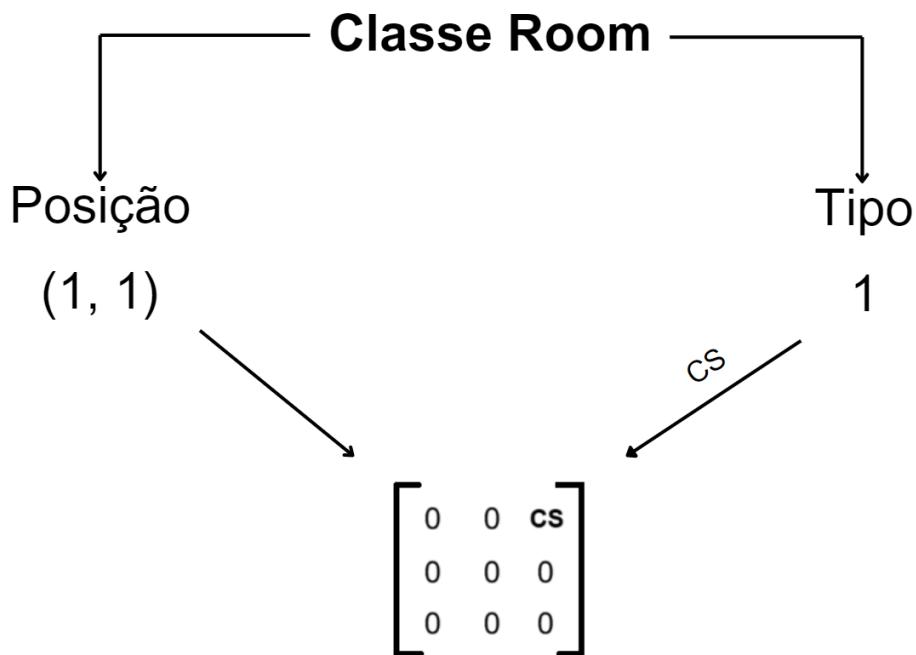
### 3.3.1 Criação do esboço do mapa

A criação do ambiente hospitalar foi baseada no objetivo de torná-lo, ainda respeitando a estrutura de salas hospitalares, aleatório a cada iteração de treinamento do agente, normalmente

chamada de episódio. O motivo de modificar o ambiente a cada iteração é evitar a criação de um viés no agente, que formaria um comportamento baseado na estrutura hospitalar. Novos ambientes tornariam esse agente enviesado inapto a navegar.

Primeiramente é criado um esboço do mapa, sem se preocupar com o que estará contido dentro de cada sala. Para isso, utiliza-se de um objeto *Room*, o qual conterá um vetor de duas dimensões com a posição de cada sala e seu devido tipo. O tipo 0 se refere a uma sala comum, o tipo 1 um armazenamento central, o tipo 2 a uma sala de operação e o tipo 3 a um departamento de esterilização. A construção do mapa sempre se inicia com uma *Room* com tipo de armazenamento central, que será associada a posição (0,0) da matriz. Adicionar salas significa adicionar novos objetos *Room* a uma lista de salas. A figura 3.1 mostra um exemplo da classe *Room* ao associar a ela uma posição (1, 1) e o tipo 1, que se refere a um armazenamento central e foi representado na figura como *CS*. Esses dados de posição e tipo são utilizados posteriormente para atribuir a sala um modelo 3D na posição correta.

Figura 3.1 – Exemplo de associação de uma posição e um tipo a um objeto *Room*.



Fonte: Elaborado pelo próprio autor.

A tabela 3.1, por sua vez, deixa explícito a relação de cada tipo com sua sala hospitalar equivalente.

Para criar um layout aleatório, mas ainda parecido com o de um hospital, são utilizados parâmetros chamados *randomPerc* (P), *randomCompareStart* (S) e *randomCompareEnd* (E), que compõem um valor de comparação *randomCompare* (C) através de interpolação linear, que

Tabela 3.1 – Relação entre tipo e sala hospitalar equivalente.

Tipo	Sala hospitalar
0	Sala comum, com ou sem obstáculos
1	Armazenamento central
2	Sala de operação
3	Departamento de esterilização

Fonte: Elaborado pelo próprio autor.

fará com que o posicionamento de salas aumente ou reduza o efeito de ramificação no mapa. Essa operação é feita a cada sala que é adicionada. A construção do valor de comparação C é apresentado na equação 3.1, sendo i o índice da iteração ao adicionar sala a sala.

$$C = S + (E - S) \times \frac{i}{\text{número total de salas}} \quad (3.1)$$

Quanto maior o número de salas totais, maior o valor de C e, portanto, maior será o efeito de ramificação. Conforme for ocorrendo as iterações, o valor de C será reduzido e o efeito de ramificação diminuirá até ser contido. Para efeito de ramificação se asselhando a um hospital, foi utilizado os valores de 0,7 para S e 0,5 para E.

A partir do valor de comparação randomCompare, poderá ocorrer 3 principais formas de posicionamento:

1. **Posicionamento aleatório:** maior chance de ocorrer quanto menor for o valor de randomCompare. Escolhe posições aleatórias para a próxima sala, contanto que esteja conectada com pelo menos uma outra sala.
2. **Posicionamento de ramificação:** maior chance de ocorrer quanto maior o valor de randomCompare. Considera apenas a salas que possuem 1 sala vizinha e adicionam a nova sala em uma nova direção a partir dela, contanto que não colida com outra sala.
3. **Posicionamento de salas de operação:** quando 70% tiverem sido posicionadas, todas as próximas salas serão posicionadas dessa forma. Considera apenas a salas que possuem 1 sala vizinha e adicionam a nova sala apenas em posições que não haverá contato com quaisquer outras salas. Elas salas serão posteriormente classificadas como salas de operação.

O Pseudocódigo retratado no algoritmo 1 resume a forma que acontece a iteração durante a criação do esboço do mapa hospitalar. Existem mais fatores durante o código que foram omitidos do pseudocódigo pois fazem parte de situações específicas de currículum ou não agregariam para o entendimento do algoritmo como um todo.

**Algorithm 1** Pseudocódigo resumido para a função de esboço do mapa

**Result:** Criação de salas para um mapa hospitalar

- 1 Inserir a posição do centro na lista de posições ocupadas
- 2 Inicializar a variável checkPos como um vetor zero
- 3 Definir os valores iniciais para randomCompare, randomCompareStart e randomCompareEnd
- 4 **for** cada sala que precisamos adicionar (*i* de 0 até número de salas - 1) **do**
- 5     Calcular randomCompare usando a equação 3.1
- 6     Obter uma nova posição checkPos
- 7     **if** *i* for igual a última sala **then**
- 8         Definir checkPos como uma posição solitária com tag de departamento de esterilização
- 9     **else**
- 10         **if** a razão entre *i* e número de salas for maior que 0.7 **then**
- 11             Definir checkPos através de Posicionamento de salas de operação
- 12         **else**
- 13             **if** o número de vizinhos de checkPos for maior que 1 e um valor aleatório for maior que randomCompare **then**
- 14                 **while** o número de vizinhos de checkPos for maior que 1 **do**
- 15                     Definir checkPos através de posicionamento de ramificação
- 16     Adicionar uma nova sala com seu devido tipo na posição checkPos na matriz de salas
- 17     Inserir checkPos no início da lista de posições ocupadas

### 3.3.2 Instanciação de elementos 3D

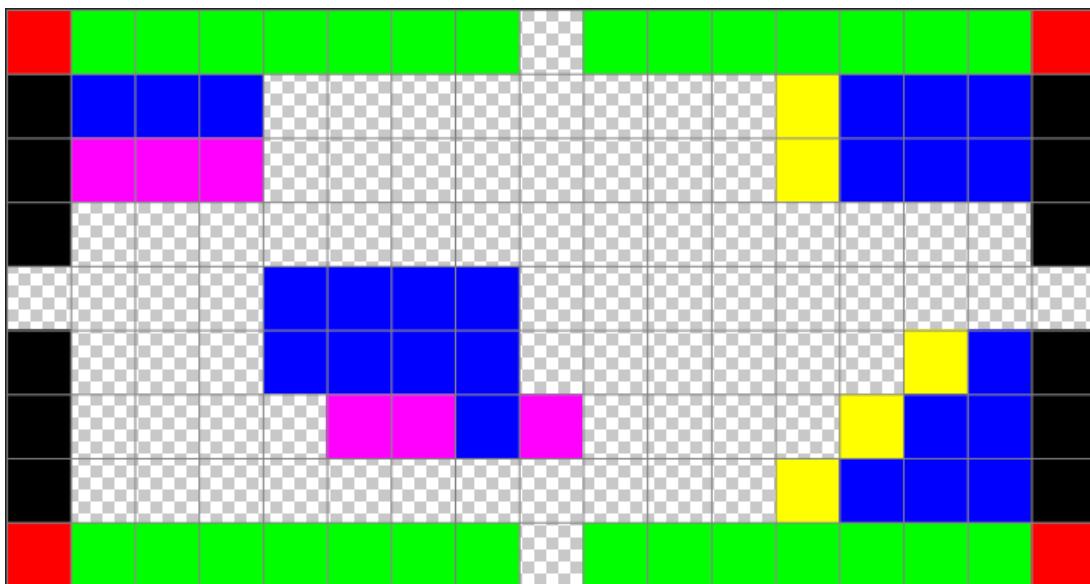
Após criação do esboço do mapa, a seguinte etapa é alocar a posição de objetos como obstáculos, paredes, colunas e portas dentro de cada sala. Para isso, é utilizado outro algoritmo que realiza a alocação seguido da importação de componentes 3D. Seu funcionamento é dividida na associação dos objetos por templates feitos em photoshop, instanciação de componentes 3D nas posições e rotações corretas e geração de corredores.

Em primeiro lugar, o posicionamento dos objetos é feita a partir de templates com diferentes desenhos de posicionamento, representados por diferentes cores. Os templates são arquivos arquivos no formato Photoshop com extensão .psd com uma arte em pixels representando onde na sala deve ser colocado os diferentes componentes 3D. Um exemplo de template é demonstrado na Figura 3.2. As representações de cores foram utilizadas como a seguir:

- **Preto:** Porta;
- **Verde:** Porta, porém com rotação de 90°;
- **Vermelho:** Pilar;
- **Azul:** Obstáculo;

- **Roxo:** Ponto de entrega de instrumentos cirúrgicos;
- **Amarelo:** Ponto de coleta de instrumentos cirúrgicos;

Figura 3.2 – Exemplo de template de uma sala de operação.



Fonte: Elaborado pelo próprio autor.

Um obstáculo é uma região de 1 unidade Unity por 1 unidade Unity ocupada por um objeto com colisão ativada, todos os agentes que se colidirem com ele terão suas velocidades e recompensas afetadas.

Os templates estão classificados por categorias, sendo elas: templates para armazenamento central, templates para departamentos de esterilização, templates de salas com apenas um vizinho, templates para corredores e templates para salas com mais de um vizinho. Posteriormente foi associado mais uma categoria para uma lição específica de curriculum, o que será apresentado na seção 3.5.

Para finalizar o funcionamento dos templates com o Unity, tem-se uma configuração na cena do Unity que cria a relação de cada cor com seu respectivo objeto. Isso é feito a partir de valores em HSV.

Com os templates e a configuração construídos, segue-se o seguinte passo a passo para ser possível instanciar os componentes 3D nas posições e rotações corretas e gerar corredores:

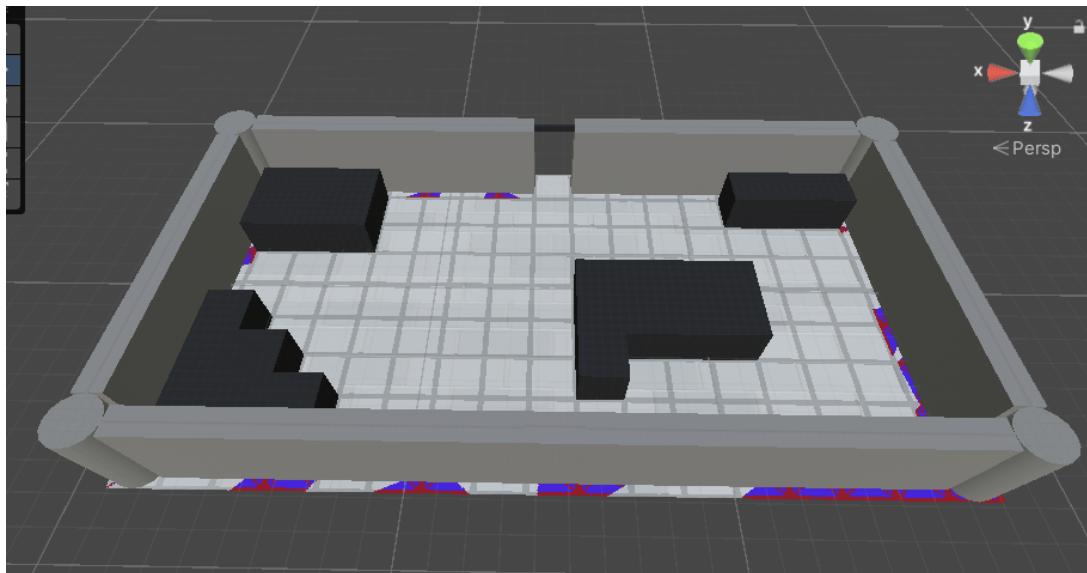
1. Importar os arquivos .psd com os templates para o Unity e gerar divisões de pixels através de um método próprio do Unity. Com isso teremos um objeto próprio do Unity, chamado *sprite*, que conterá cada pixel e sua devida cor;

2. A partir das informações atribuídas para cada sala, como sua posição, seu tipo e o número de vizinhos ao seu redor, determinar qual categoria de template usará e escolher aleatoriamente um desses templates;
3. Para cada pixel, identificar a partir seu valor HSV instanciar seu respectivo componentes 3D nas suas devida posição dentro do ambiente de treino.

No caso específico de haver salas com 4 vizinhos, essas salas terão suas paredes e portas removidas e será utilizado templates de corredores para instanciação, com objetivo de gerar corredores. Isso, porém, não acontece com a armazenamento central.

A figura 3.3 demonstra o template da figura 3.2 instanciado no ambiente de treinamento. Os pontos de entrega e coleta de instrumentos cirúrgicos não estão representados pois são desativados logo depois de sua criação. Ele serão atribuídos posteriormente e em quantidades limitadas. A seção 3.3.3 explica como é feita suas criações.

Figura 3.3 – Template de uma sala de operação convertido em uma sala no ambiente de treinamento.

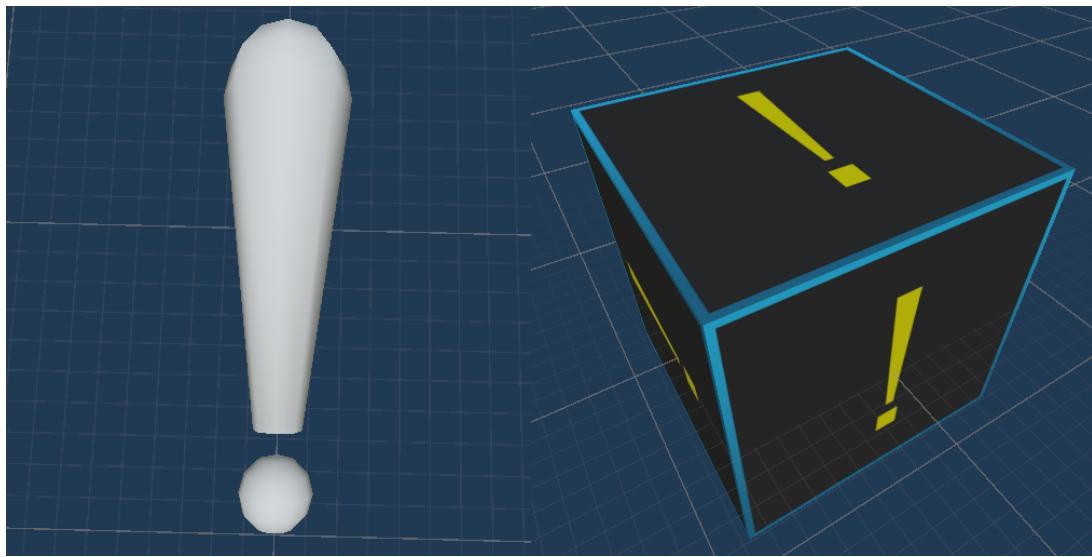


Fonte: Elaborado pelo próprio autor.

### 3.3.3 Seleção e Utilização de pontos de entrega e coleta de instrumentos cirúrgicos

Para simular pontos de entrega e coleta de instrumentos cirúrgicos, utilizou-se modelos gratuitos fornecidos pela loja de modelos do Unity. Sendo o ponto de entrega um bloco com um sinal de ponto de exclamação, enquanto o ponto de coleta é um modelo 3D de ponto de exclamação. A figura 3.4 demonstra ambos os modelos utilizados.

Figura 3.4 – Modelo utilizado para ponto de entrega, à direita, e modelo utilizado para ponto de coleta, à esquerda, de instrumentos cirúrgicos.



Fonte: Elaborado pelo próprio autor.

Toda lógica de operação dos pontos de entrega e coleta é feita considerando-os como pares. Para cada ponto de coleta sempre haverá um ponto de entrega específico para ele. Para seleção, é definido em cada lição do curriculum learning a quantidade de pares que haverá dentro do ambiente ao ser criado. Novos pontos de coleta ou entrega não são criados durante o treinamento, apenas no momento de criação do cenário.

Após a criação e instanciação das salas no ambiente de treinamento, são coletados os dados do armazenamento central, do departamento de esterilização e, aleatoriamente, de algumas salas de operação. A quantidade salas de operação é o valor da quantidade de pares para pontos de entrega e coleta reduzido em um, pois a última posição sempre é reservada para o armazenamento central ou departamento de esterilização. Como exemplo, caso seja determinada a utilização de 3 pares, haverá seleção de 2 salas de operação mais o departamento de esterilização para pontos de entrega e haverá a seleção de 2 salas de operação mais o armazenamento central para pontos de coleta, sendo que é permitido a repetição de uma sala de operação para ponto de entrega e coleta.

Para cada sala selecionada, é escolhido aleatoriamente um dos pixels de ponto de coleta ou de entrega do template para ser ativado e instanciado na cena.

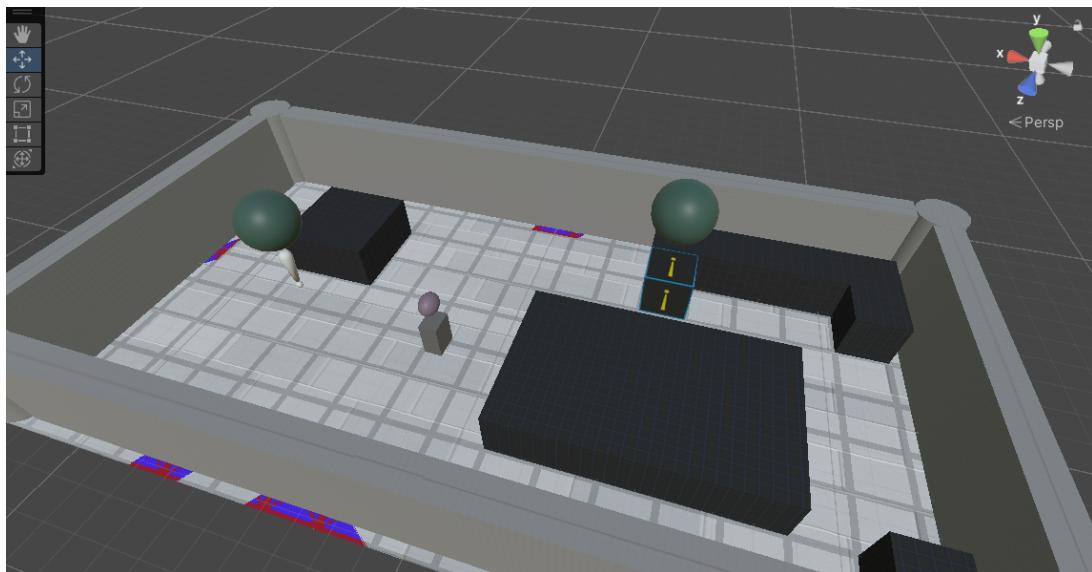
Para criar a relação de pontos de entrega específicos para pontos e coleta, todos os pontos selecionados são unidos em pares aleatoriamente em uma variável nova. Essa variável então será passada por uma função que criará a relação entre os dois da seguinte forma: para cada par, é criado uma esfera, a qual o Unity categoriza como objeto primitivo, que sempre permanecerá em cima dos modelos de ambos os pontos. Logo após, é feita a seleção aleatória de uma cor, em

escala HSV, para essas duas esferas.

Dessa forma, pode-se criar outro algoritmo que transfere o valor HSV da esfera do ponto de coleta para o agente e do agente para o ponto de entrega. Essa transferência acontece quando o Unity reconhece uma colisão entre os modelos e também serve para indicar se o agente possui ou não um instrumento cirúrgico com ele. Caso o agente já possua um instrumento associado a ele, colisões com outros pontos de coleta serão ignoradas. O mesmo vale para colisões com pontos de entrega quando o agente não possuir um instrumento.

A entrega de um instrumento cirúrgico só é feita quando houver o contato do agente carregando um instrumento com o devido ponto de entrega. Tanto o ponto de entrega quanto a representação do instrumento no agente desaparecem após a entrega. A figura 3.5 apresenta um par criado e um agente carregando um instrumento.

Figura 3.5 – Par atribuído a um ponto de entrega e um ponto de coleta, assim como um agente carregando um instrumento cirúrgico.

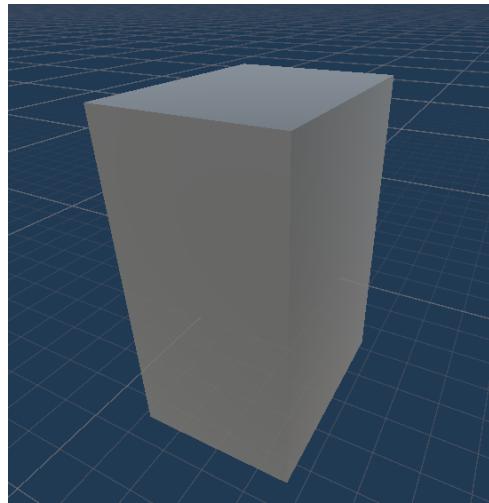


Fonte: Elaborado pelo próprio autor.

## 3.4 Construção e Operação do Agente

O agente é representado com um objeto primitivo de retângulo e pode ser visto através da figura 3.6.

Figura 3.6 – Objeto primitivo utilizado para o agente e alguns de seus parâmetros.



Fonte: Elaborado pelo próprio autor.

Seus atributos são discutidos nas seções 3.4.1, 3.4.2 e 3.4.3.

### 3.4.1 Ações do Agente

Dentro do ML-Agents, uma ação é uma instrução da política que o agente executa. A ação é passada para um receptor, como um objeto com componente de agente ou um atuador, a cada iteração feita pela *Academy*. Existem dois tipos de ações suportadas: contínuas e discretas.

O agente construído utiliza cinco ações discretas que realizam as seguintes funções: duas ações discretas para rotação no eixo perpendicular ao chão (rotacionar para esquerda ou para direita), duas ações para locomoção no eixo paralelo ao chão (avançar ou recuar) e uma ação vazia, que representa o agente escolher permanecer parado.

### 3.4.2 Recompensas do Agente

A recompensa retrata um valor escalar que indica o desempenho do agente. Ela pode ser atribuída a qualquer momento durante o treinamento, desde de que estejam configurados de uma forma que sua maximização gere o comportamento ideal desejado.

Com a utilização do algoritmo MA-POCA, é disponibilizado a opção de utilizar recompensas tanto para um agente individual, como também para um time inteiro. O intuito é fazer com que os agentes aprendam a melhor forma de contribuir para alcançar a recompensa global do time. Porém, com as recompensas recebidas individualmente, a equipe poderá ter a chance de trabalhar em conjunto para ajudar o indivíduo a atingir certos objetivos ou evitar certas condições.

Para o trabalho, são atribuídos as seguintes recompensas para os agentes:

- **Recompensa por entrega de instrumento:** é atribuído uma recompensa ao grupo inteiro quando um agente entrega um instrumento em um ponto de entrega. O valor da recompensa é 1 dividido pelo número total de pares de pontos de coleta e entrega instanciados no início do treinamento. Não é dado nenhum tipo de recompensa quando o agente adquire o instrumento no ponto de coleta;
- **Recompensa por colisão:** recompensa negativa que é atribuída a apenas um agente, quando o mesmo colide com um parede, um obstáculo ou um outro agente. Possui valor de  $-0,05$  por colisão.

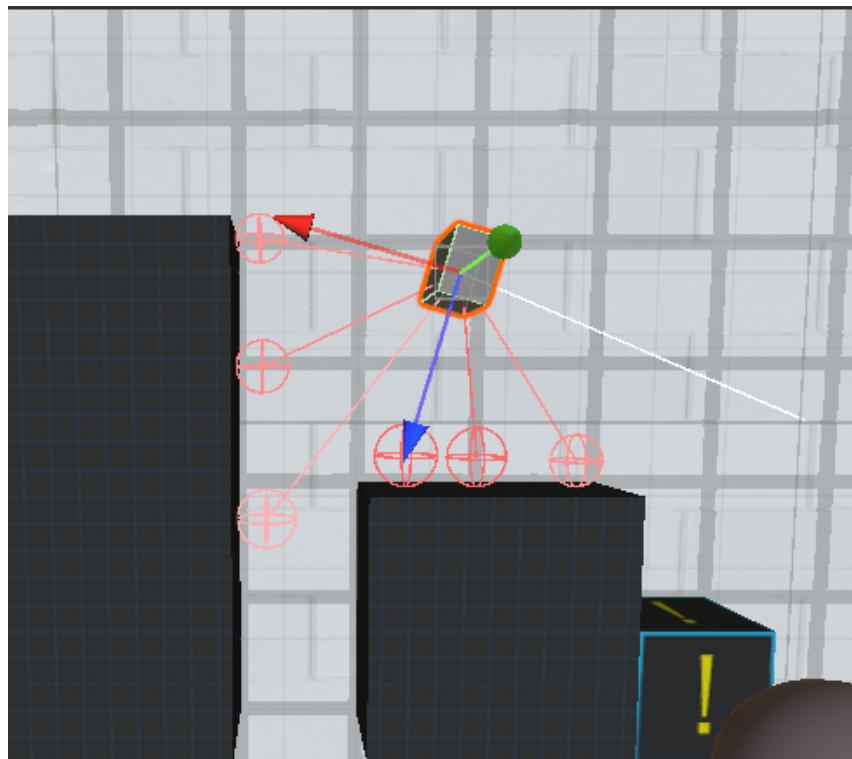
### 3.4.3 Observações do Agente

Observações são informações sobre o ambiente passados pelos sensores ou componentes associados a ele. Elas podem ser numéricas ou visuais. As observações numéricas medem atributos do ambiente do ponto de vista do agente. Para ambientes mais complexos, um agente exigirá diversas observações numéricas contínuas. As observações visuais, por outro lado, são imagens geradas a partir das câmeras acopladas ao agente e representam o que o agente está vendo naquele momento. No geral, é recomendado que gere observações para o agente pensando no que seria preciso para que fosse feita uma solução analítica para o problema.

Para o agente do trabalho, são utilizadas apenas observações numéricas, porém de dois tipos diferentes, três conjuntos de observações de tamanho fixo e outra de tamanho variável. O primeiro conjunto de observações fixas contém apenas uma observação de tamanho fixo, a qual sempre está presente em todos os agentes e contém apenas uma representação booleana, que é verdadeira quando o agente está com um instrumento em sua posse. Tal informação é importante para o agente determinar se deve prosseguir para um ponto de coleta ou entrega.

O segundo conjunto de observações fixas é realizada ao anexar um componente ao agente chamado de *raycast observation*. Através desse componente, o agente emite raios que identificam contato com algum tipo de superfície, se assemelhando a sensores LiDAR. Para o agente do trabalho, foi anexado ao agente um componente de *raycast* que emite sete raios, cada um como 28,3 graus de distância um do outro, sendo que há um raio de referência que é sempre emitido na frente do agente. Os raios possuem distância de operação de 8 medidas do Unity e identificam apenas paredes, obstáculos, outros agentes e pontos de entrega e coleta. Com a adição desse componente, o número real de observações adicionadas é a multiplicação do número de raios com o número de possíveis pontos de identificação. Portanto, o componente adiciona para o agente do trabalho 35 observações. A figura 3.7 possui o exemplo de um agente com um componente *raycast* associado a ele, seguindo as configurações citadas anteriormente.

Figura 3.7 – Componente raycast atrelado ao agente, com raios vermelhos representando contato com outro objeto.



Fonte: Elaborado pelo próprio autor.

O terceiro conjunto de observações fixas é a distância e o alinhamento de outros agentes no mapa. O intuito dessa observação é permitir que o agente decida qual ponto de coleta ou entrega perseguir, levando em conta onde outros agentes estão. Isso possibilita ao agente selecionar pontos que estão mais próximo e que não estejam no caminho de outros agentes. Para essa observação, ocorreu uma dificuldade de implementação, pois o Unity só permite ter um conjunto de observação variável por agente. A ideia inicial seria ter um conjunto variável para o número de pontos e outro para o número de agentes, mas devido a essa limitação, foi utilizado um terceiro conjunto fixo que adiciona 3 pares de distância mais alinhamento de outros agentes. Isso limita a utilização da política para, no máximo, 4 agentes na cena, pois acima de 5 o modelo não suporta adição de mais observações. No caso de haver menos de 4 agentes, os pares são zerados até se igualarem com o número de agentes.

O conjunto de observações variável é dependente da quantidade de pontos de coleta e entrega presentes no mapa. Caso haja dois pares de coleta e entrega, o conjunto terá tamanho dois. Além disso, cada conjunto é uma lista de quatro pontos flutuantes, que representam diferentes características do ambiente.

O primeiro número real determina a distância do agente para a próxima porta que o

levará a caminho de um ponto de entrega ou coleta. Essa observação surgiu da ideia de remover a dificuldade do agente procurar o caminho para os pontos por tentativa e erro, evitando um cenário de labirinto no hospital. Para ser possível determinar o caminho de menor distância a um ponto de coleta ou entrega, foi elaborado um algoritmo de busca A\* adaptado para matriz de posição das salas.

Primeiramente, é pressuposto para o algoritmo que o movimento é em nós com possibilidade de até 8 direções, contando que não haja nenhuma parede bloqueando o caminho para uma movimentação direta entre salas diagonais. Em seguida, define-se 3 variáveis de custo para cada nó:

- **Custo G:** custo de locomoção entre um nó e outro. Para locomoção horizontal e vertical, seu valor é 10, enquanto que para locomoção diagonal, seu valor é 14.
- **Custo H:** custo heurístico para se chegar até o nó final. Ele representa uma estimativa do custo mínimo para ir de um nó específico até o nó objetivo
- **Custo F:** custo total, sendo a soma dos custos G e H.

Através desses custos, é calculado o melhor caminho de uma determinada sala até o ponto de entrega ou de coleta. O algoritmo 2 representa a lógica para obtenção do melhor caminho através desses custos.

**Algorithm 2** Algoritmo de Busca A\*

**Requer** :PosiçãoInicial, PosiçãoFinal

**Resultado** :Caminho de PosiçãoInicial a PosiçãoFinal

18 Inicialize a listaAberta com o nóInicial

19 Inicialize a listaFechada como vazia

20 **for** cada nó na grade **do**

21   | Inicialize o custoG como  $\infty$ , custoH e custoF

22   | Defina nóAnterior como nulo

23 Defina o custoG do nóInicial como 0

24 Calcule o custoH e o custoF do nóInicial

25 **while** listaAberta não está vazia **do**

26   | nóAtual  $\leftarrow$  nó na listaAberta com o menor custoF

27   | **if** nóAtual é nóFinal **then**

28   |   | **return** caminho de nóInicial a nóFinal

29 Mova o nóAtual da listaAberta para a listaFechada

30 **for** cada vizinho do nóAtual **do**

31   | **if** vizinho está na listaFechada **then**

32   |   | Pule a iteração atual

33   | custoGTentativo  $\leftarrow$  custoG do nóAtual + distância do nóAtual para o vizinho

34   | **if** custoGTentativo < custoG do vizinho **then**

35   |   | Defina nóAnterior do vizinho como nóAtual

36   |   | Defina custoG do vizinho como custoGTentativo

37   |   | Calcule o custoH e custoF do vizinho

38   |   | **if** vizinho não está na listaAberta **then**

39   |   |   | Adicione o vizinho à listaAberta

40 **return** null

Através do resultado do algoritmo 2, é possível obter qual a próxima sala no caminho traçado pelo A\*, e portanto, sua posição dentro do ambiente de simulação. Basta então obter a distância da porta que o agente deverá passar. Porém, antes de ser adicionado a observação, essa distância é normalizada com um valor entre 0 e 1.

O segundo número real se refere ao alinhamento do agente com a porta. Caso o agente esteja apontando sua parte frontal para a porta, o valor será 1 e, caso esteja apontando para o sentido oposto, o valor será -1.

Para obter este valor, foi utilizado uma função do Unity que retorna um vetor unitário apontado para a direção frontal do agente. Além disso, com as coordenadas do agente e da porta disponíveis, devido a etapa do A\*, é possível realizar uma subtração dessas coordenadas para obter outro vetor, porém apontando do agente para a porta.

Após normalizar o vetor do agente a porta, é feito um produto vetorial entre os dois

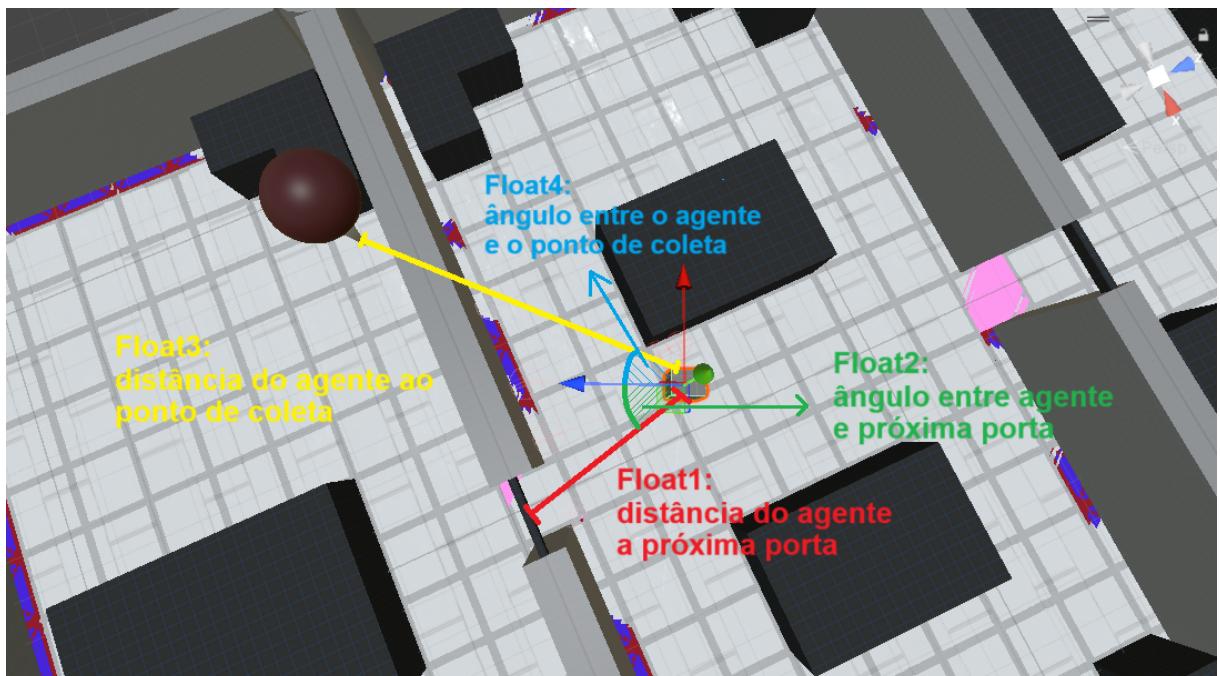
vetores, o que resulta no valor de alinhamento do agente com a porta.

O terceiro número real também é uma distância, porém do agente ao ponto de coleta ou de entrega, que é obtido diretamente com as posições de ambos no ambiente.

O quarto e último ponto é o alinhamento entre o agente e o ponto de coleto ou de entrega. O cálculo é o mesmo utilizado no segundo valor flutuante, porém agora tomando em conta o alinhamento do agente com o ponto.

A figura 3.8 demonstra visualmente os valores flutuantes dentro da observação variável para o caso de um ponto de coleta.

Figura 3.8 – Pontos flutuantes para um conjunto de observação variável de um ponto de coleta.



Fonte: Elaborado pelo próprio autor.

## 3.5 Etapas do Curriculum Learning

A metodologia de curriculum para aprendizado por reforço fundamenta-se na capacitação de agentes através de uma sequência de tarefas de origem, que se tornam progressivamente mais desafiadoras, culminando na realização de uma tarefa-alvo (NARVEKAR; STONE, 2020). Cada tarefa subsequente nesta sequência capitaliza as habilidades adquiridas nas tarefas anteriores, permitindo o desenvolvimento gradual do conjunto de habilidades necessário para a resolução da tarefa final.

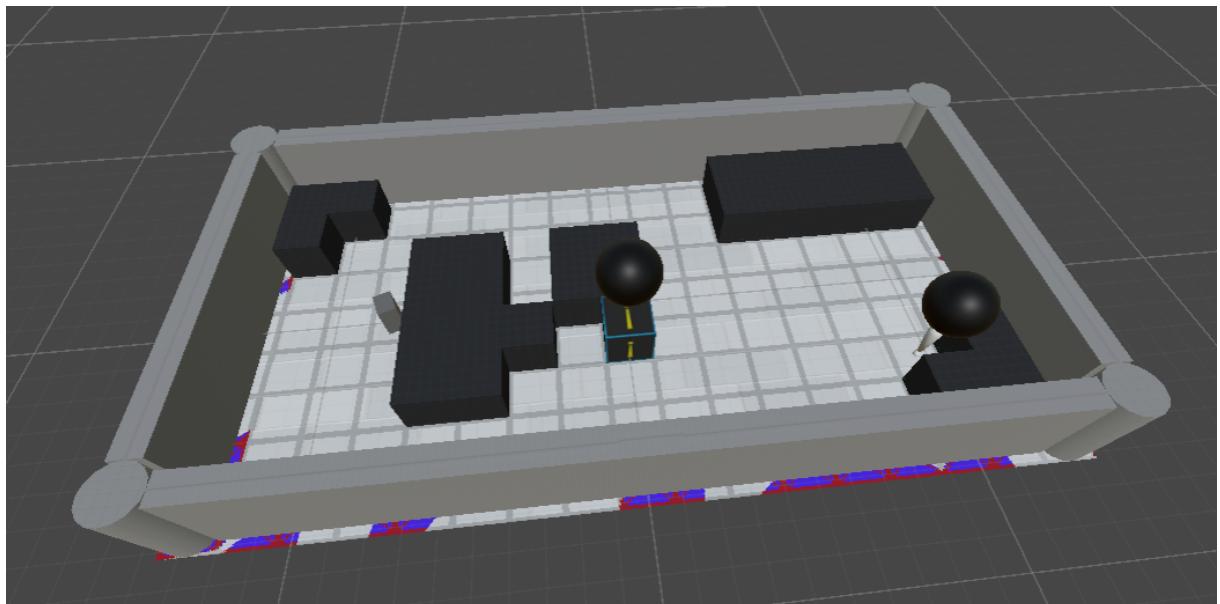
Este método de aprendizado espelha a dinâmica de aprendizado observada em seres vivos. Ambientes como o criado por Leibo et al. (2019) consideram alteração da dinâmica ambiental, gerando um novo conjunto de desafios adaptativos e um currículum que surge espontaneamente de dinâmicas não estacionárias, classificada como autocurricula.

Para este trabalho, foi utilizado a formação de 7 diferentes ambiente, cada um com suas devidas qualidades e propostas para o agente, chamadas de lições. Todos esses ambientes serão discutidos nas seções 3.5.1 à 3.5.7.

### 3.5.1 Primeira Lição (Sala Única)

A primeira lição utiliza apenas uma sala hospitalar, com um ponto de entrega, um ponto de coleta e um agente. São apenas utilizados os templates de salas de operação, pois possuem uma boa variação de exemplos e uma boa quantidades de obstáculos para interferir no trajeto do agente. Tanto os pontos de entrega quanto de coleta possuem uma quantidade de possíveis locais de instanciação que varia de 3 a 6 pontos, dependendo do template da sala de operação. A figura 3.9 demonstra uma das possíveis configurações de salas para a primeira lição.

Figura 3.9 – Exemplo de uma sala instanciada na primeira lição.



Fonte: Elaborado pelo próprio autor.

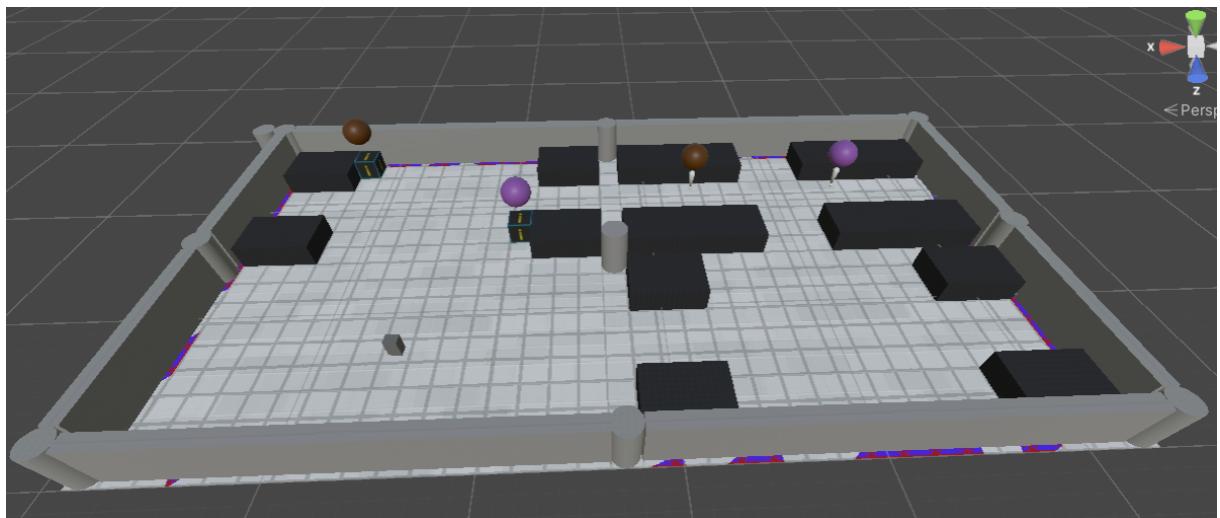
Como a primeira lição lida com o treinamento de uma rede com pesos zerados ou aleatórios, as ações tomadas pelo agente serão totalmente aleatórias, portanto evitar recompensas escassas se torna um tática facilitadora de aprendizado. Para isso, é preciso que ocorra frequentemente a atualização da política para o objetivo desejado utilizando um ambiente menor com maiores chances do agente receber uma recompensa através de ações aleatórias. O intuito da

primeira lição é, portanto, ensinar ao agente a principal correlação de recompensa do todos os ambiente, o contato em um ponto de entrega ao possuir um instrumento cirúrgico.

### 3.5.2 Segunda Lição (Grande Sala Única)

A segunda lição expande o tamanho da sala da primeira lição, sendo agora a união de quatro salas hospitalares, porém sem paredes entre elas. Sempre há a presença de um armazenamento central e um departamento de esterilização na sala, porém as duas outras salas variam no template utilizado. Para se evitar colocar obstáculos muito impeditivos para a trajetória do agente, foi feito uma nova categoria de templates específica para esta lição, que é utilizada para as salas fora o armazenamento central e departamento de esterilização. Além disso, foi adicionado mais um par ponto de entrega e coleta ao ambiente, mas ainda apenas há um agente. A figura 3.10 mostra um exemplo de ambiente para essa lição.

Figura 3.10 – Exemplo de uma sala grande instanciada na segunda lição.



Fonte: Elaborado pelo próprio autor.

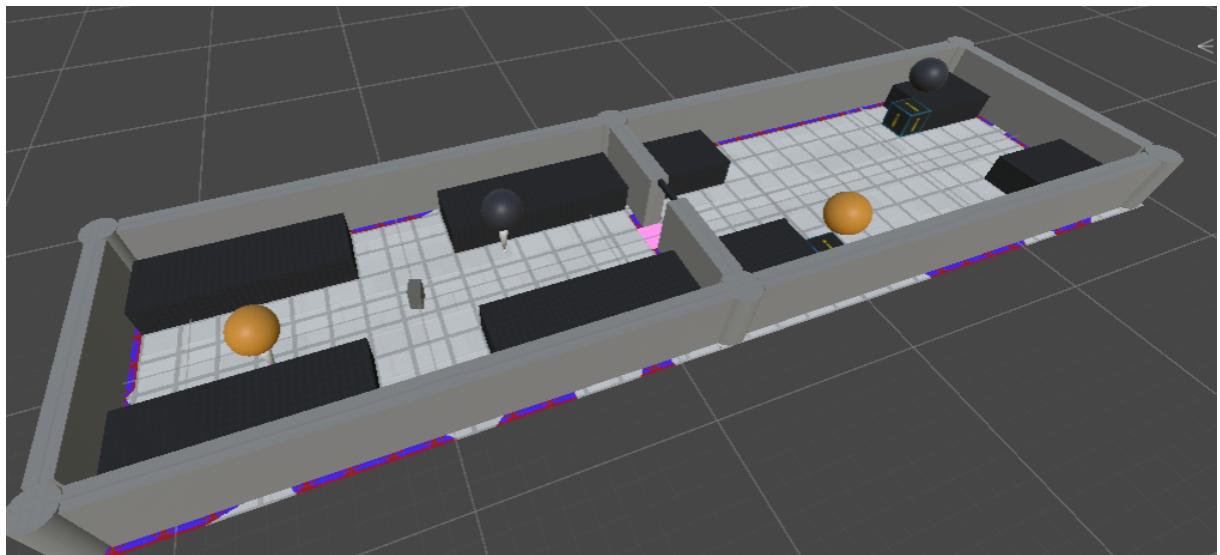
Os principais objetivos dessa lição é aumentar a capacidade de navegação do agente e apresentar a possibilidade de haver mais de um par ponto de entrega e coleta, que culmina em duas observações variáveis. Com isso, o agente deve aprender a ir em direção do ponto de coleta que determinar como melhor.

Apesar de haver quatro salas unidas na segunda lição, as observações de distância e alinhamento da porta mais próxima são zerada, pois não há paredes entre as salas.

### 3.5.3 Terceira Lição (Duas Salas)

Para terceira lição, é utilizado duas salas hospitalares. As salas sempre são o armazenamento central e o departamento de esterilização, mas com variação na localização dos pontos de entrega e coleta. São mantidos dois pares de pontos e um agente no ambiente. A figura 3.11 oferece um exemplo para as salas dessa lição.

Figura 3.11 – Exemplo de um ambiente instanciado na terceira lição.



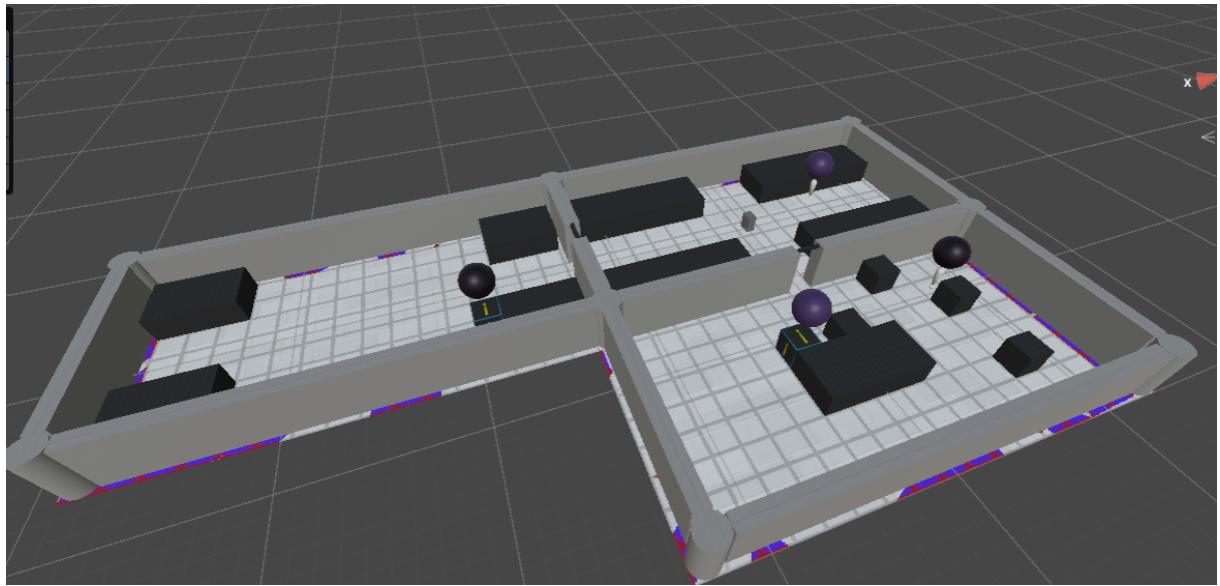
Fonte: Elaborado pelo próprio autor.

A partir dessa lição, é introduzidos as observações de distância e alinhamento com a porta mais próxima.

### 3.5.4 Quarta Lição (Três Salas)

A quarta lição adiciona mais uma sala no processo de criação do ambiente. Apesar de ser adicionado apenas uma sala, já é possível aumentar a variação de configurações de ambiente para o agente. A figura 3.12 demonstra um exemplo do ambiente da quarta lição.

Figura 3.12 – Exemplo de um ambiente instanciado na quarta lição.



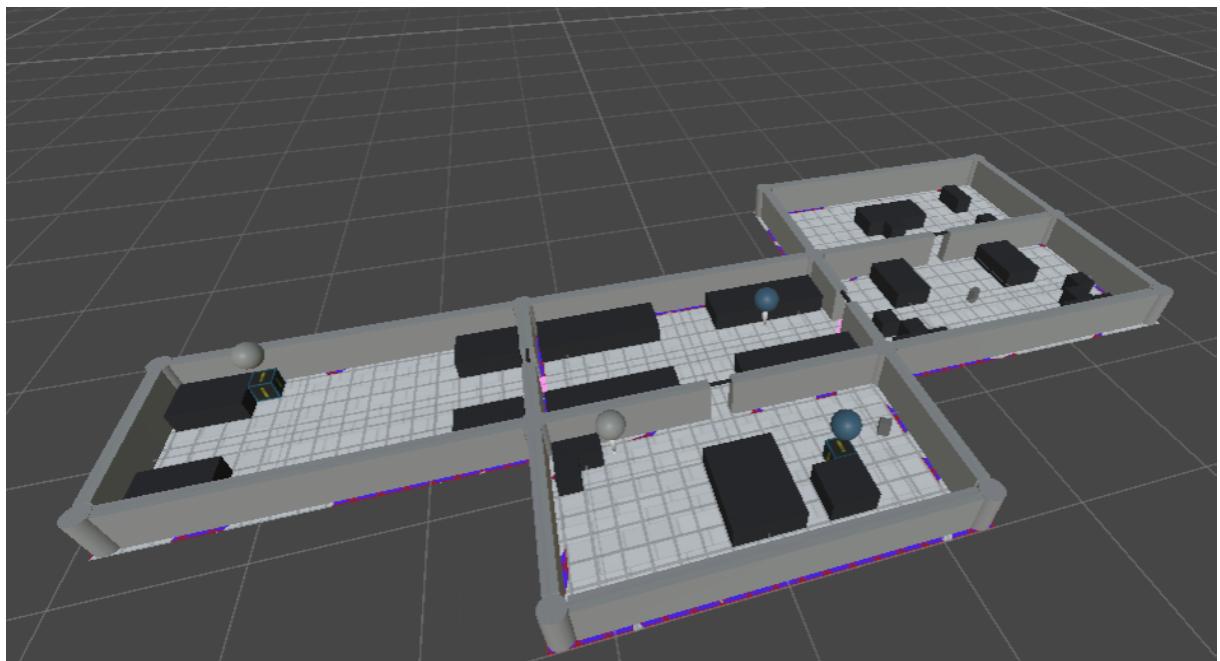
Fonte: Elaborado pelo próprio autor.

Através dessa lição, é possível reduzir possíveis enviesamentos do agente quanto a localização dos pontos de entrega e coleta. Ao migrar diretamente da terceira lição para um hospital pequeno (ambiente utilizado na quinta lição), os agentes muitas vezes não conseguem se locomover a longas distâncias, possivelmente por esperarem que o ponto de entrega ou coleta sempre esteja na próxima sala da que são instanciados. A quarta lição foi criada para amenizar esse problema e, portanto, seu objetivo é a apresentar trajetórias de entrega maiores que uma sala de distância.

### 3.5.5 Quinta Lição (Hospital Pequeno)

A quinta lição instancia 5 salas no ambiente e, com isso, é a primeira a se beneficiar fortemente pelo sistema de criação de mapa discutida nas seções 3.3.1 e 3.3.2. Além do aumento do número de salas, é acrescido um agente no ambiente, totalizando dois agentes ativos. O número de pontos de entrega e coleta, porém, não é alterado. A figura 3.13 apresenta um ambiente construído na quinta lição.

Figura 3.13 – Exemplo de um ambiente instanciado na quinta lição.



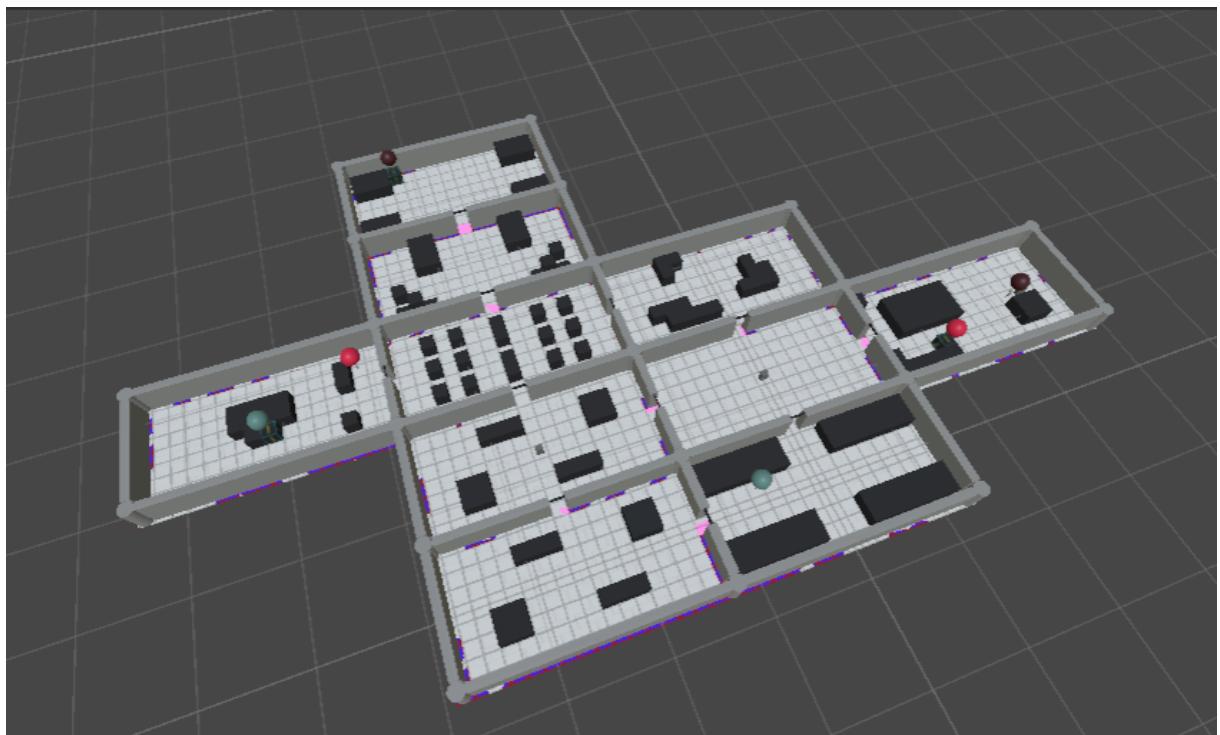
Fonte: Elaborado pelo próprio autor.

Através desta lição é possível introduzir um segundo agente ao ambiente, além de um mapa hospitalar mais complexo.

### 3.5.6 Sexta Lição (Hospital Médio)

A sexta lição apenas expande a quantidade de salas instancias. No caso, são utilizados 10 salas para o treinamento, junto com a adição de um agente, totalizando 3 agentes na cena. É acrescido também um par de ponto de entrega e coleta, totalizando 3 pares. A figura 3.13 apresenta um ambiente construído na sexta lição.

Figura 3.14 – Exemplo de um ambiente instanciado na sexta lição.

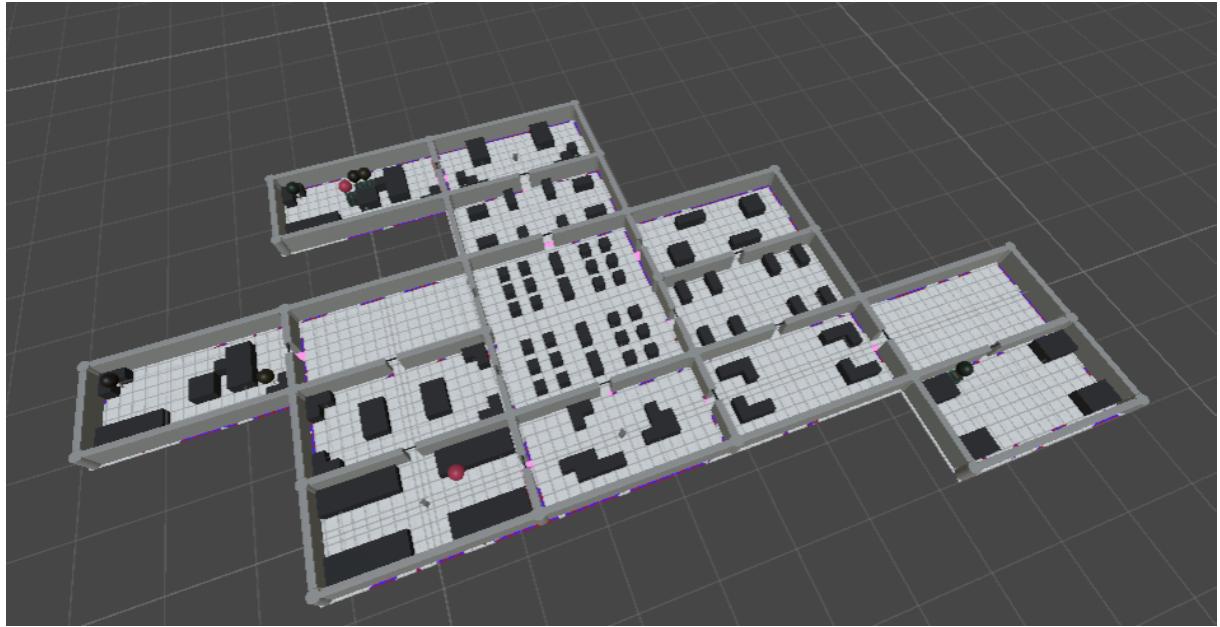


Fonte: Elaborado pelo próprio autor.

### 3.5.7 Sétima Lição (Hospital Grande)

A sétima lição também expande a quantidade de salas instancias. Para ela, são utilizadas 15 salas, junto com a adição de um agente, totalizando 4, e um par de ponto de entrega e coleta, totalizando 4 pares. A figura 3.13 apresenta um ambiente construído na sétima lição.

Figura 3.15 – Exemplo de um ambiente instanciado na sétima lição.



Fonte: Elaborado pelo próprio autor.

## 3.6 Configuração para treinamento

Baseado nos principais parâmetros apresentados na seção 2.4.2, para a rede, foram adicionadas 3 camadas escondidas (*num\_layers*), cada uma com 256 unidades (*hidden\_units*). Os valores dos principais parâmetros de treinamento são citados na tabela 3.2, levando em conta que *schedule* representa o tipo de taxa de redução de valor para *learning\_rate*, *beta* e *epsilon*.

Tabela 3.2 – Valores dos principais parâmetros para cada lição.

<b>Lição</b>	<b>max_steps</b>	<b>learning_rate</b>	<b>beta</b>	<b>epsilon</b>	<b>schedule</b>
Lição 1	$2^6$	$5^{-5}$	$1^{-2}$	0.2	constante
Lição 2	$8^6$	$5^{-5}$	$1^{-2}$	0.2	constante
Lição 3	$12^6$	$8^{-5}$	$1^{-2}$	0.3	linear
Lição 4	$6^6$	$2^{-5}$	$3, 5^{-3}$	0.17	linear
Lição 5	$12^6$	$5^{-5}$	$5^{-3}$	0.2	linear
Lição 6	$12^6$	$3^{-5}$	$1^{-4}$	0.1	linear
Lição 7	$12^6$	$1^{-5}$	$1^{-4}$	0.1	linear

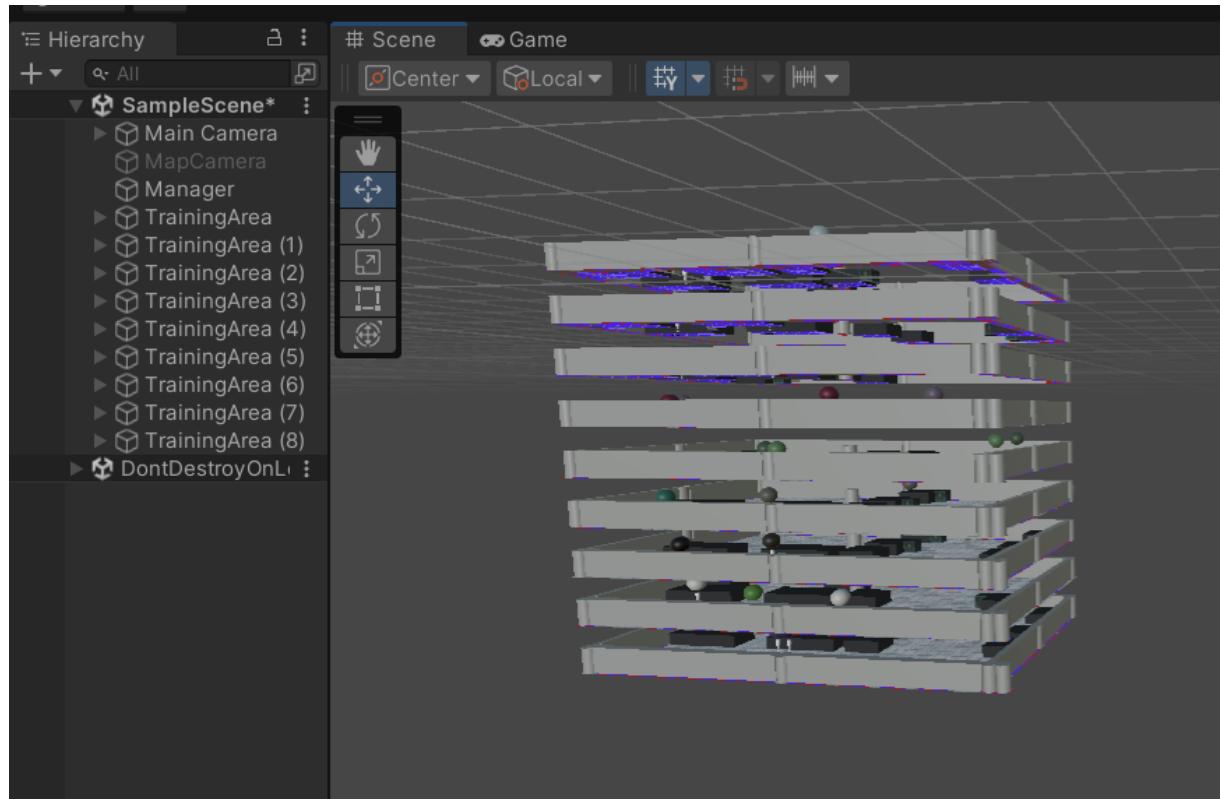
Fonte: Elaborado pelo próprio autor.

Demais parâmetros, para os diferentes casos citados, podem ser visualizados no repositório disponibilizado para o projeto.

Ademais, foi feito a utilização de treinamento com ambientes e computação em paralelo. Para paralelização dos ambientes, uniu-se todo o ambiente de treino em um mesmo objeto Unity

e, logo após, foi replicado em outras instâncias. Caso a criação do ambiente e instanciação dos agentes estejam bem contidos dentro de cada objeto Unity, eles serão utilizados paralelamente para a realização de passos dentro do treinamento da políticas. A figura 3.16 demonstra como fica projetado a paralelização de ambiente para a segunda lição.

Figura 3.16 – Exemplo de paralelização de ambiente para a segunda lição.



Fonte: Elaborado pelo próprio autor.

Ademais, o ML-Agents permite compactar o projeto inteiro em um pacote, chamado de *build*, e realizar o treinamento com vários desses builds ao mesmo tempo. Além disso, surge a opção de não utilizar elementos gráficos, caso não seja utilizado nenhuma observação visual pelo agente. Para o trabalho, foi gerado um *build* contendo 8 ambientes em um mesmo projeto e o treinamento ocorreu com 5 *builds* ao mesmo tempo, totalizando o treinamento de 40 ambientes em paralelo. A retirada dos elementos gráficos também foi realizada, o que diminui a requisição e carga computacional. Para a paralelização computacional, foi utilizado a API da NVIDIA, com suporte a tecnologia CUDA para aceleração gráfica e cálculo de matrizes. A versão utilizada foi a 12.1.

# 4 Resultados

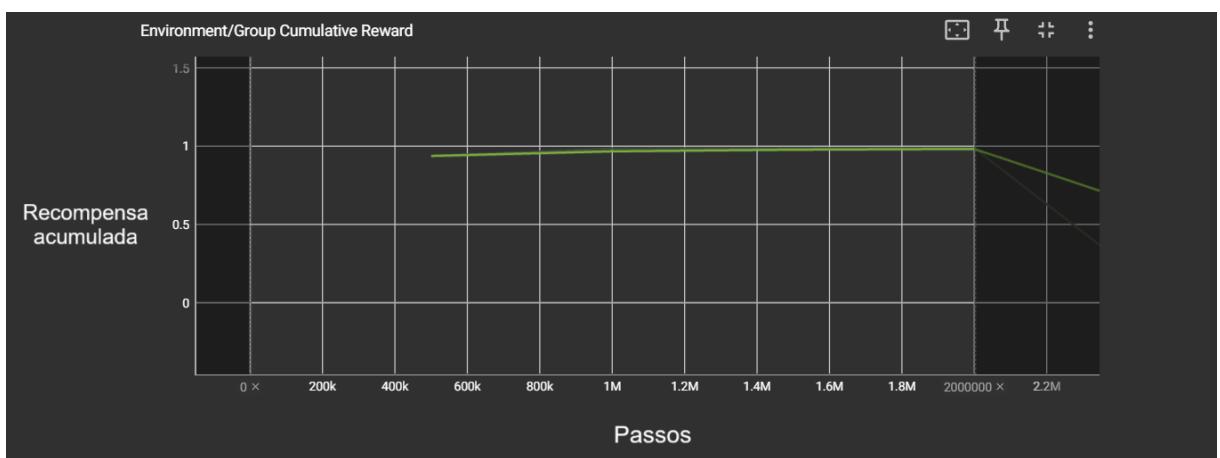
Nesta seção serão discutidos os resultados obtidos para cada lição formulada, levando em consideração o que foi esperado inicialmente, os método de treinamento utilizado e possíveis melhorias. Ademais, serão apresentados comportamentos não desejados do agente que foram observados durante a realização dos experimentos e observações abandonadas durante a realização do trabalho.

## 4.1 Resultados de Cada lição

### 4.1.1 Resultados da Primeira e Segunda Lição

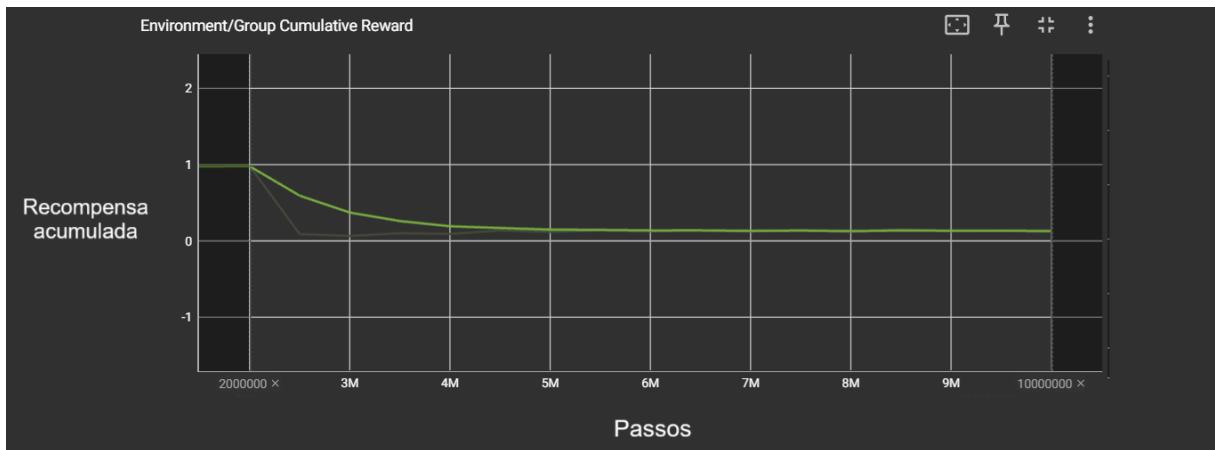
A primeira e segunda lição, por se tratarem de lições mais simples, foram feitas em um treinamento em conjunto. O total de passos utilizados para o treinamento foi de 10.000.000 passos, sendo que a primeira lição foi até 2.000.000 e a segunda até o montante final de 10.000.000. O tempo para realização do treinamento foi de 3 horas e 24 minutos. Como as lições 1 e 2 foram realizadas em conjunto, uma após a outra, é gerado apenas um gráfico para os dois. Porém o gráfico de recompensa e passos médios foram ampliados nas partes que representam cada lição para melhor interpretação. Os gráficos das figuras 4.1 e 4.2 apresentam a recompensa acumulada pelo grupo por passos para a primeira e segunda lição, respectivamente. Como o grupo dessa etapa contém apenas um agente, a recompensa se refere a suas ações.

Figura 4.1 – Gráfico de recompensa acumulada para o grupo durante a primeira lição.



Fonte: Elaborado pelo próprio autor.

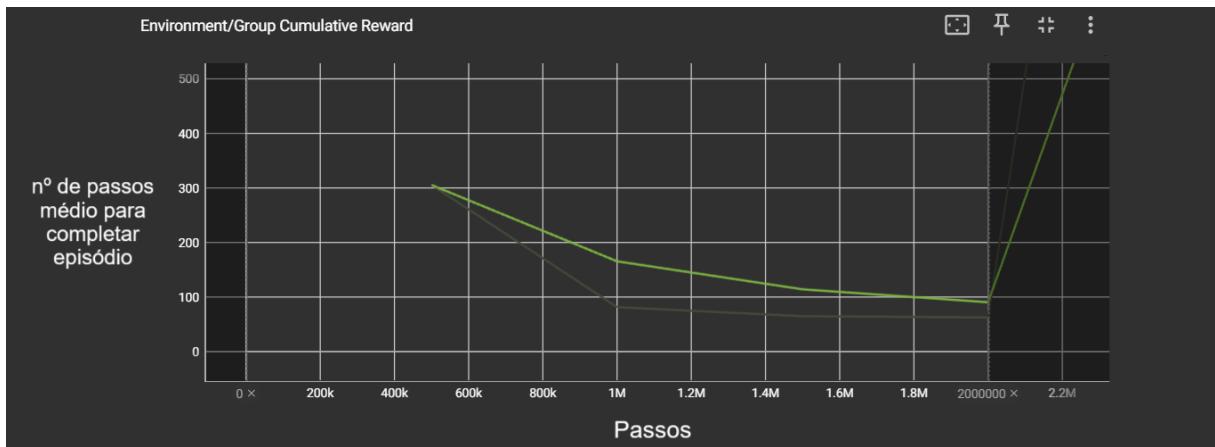
Figura 4.2 – Gráfico de recompensa acumulada para o grupo durante a segunda lição.



Fonte: Elaborado pelo próprio autor.

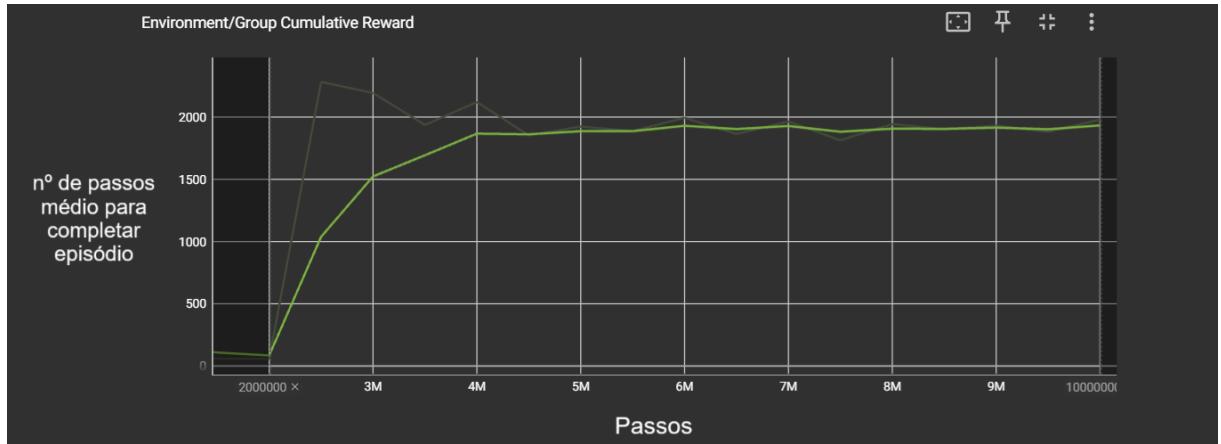
Já as figuras 4.3 e 4.4 demonstram a quantidade média de passos para se completar o episódio, através da realização de todas as entregas, da primeira e segunda lição, respectivamente.

Figura 4.3 – Gráfico da quantidade média de passos para se completar a primeira lição.



Fonte: Elaborado pelo próprio autor.

Figura 4.4 – Gráfico da quantidade média de passos para se completar a segunda lição.



Fonte: Elaborado pelo próprio autor.

Apesar de não ser exibido os 500.000 primeiros passos no gerador de gráficos utilizados, tensorboard, a primeira lição contou com uma rápida ascensão no valor de recompensa, já quase que totalmente se estabilizando a partir de 1.000.000 passos em um valor próximo de 1. Observando o comportamento nesta etapa do modelo com o modo de inferência do agente, claramente percebe-se que foi aprendido a correlação entre ponto de coleta e entrega, o que resultou em uma diminuição do tempo de entrega total. Em primeiras iterações do trabalho, o valor de recompensa negativo por contato com paredes e obstáculos era maior e, por conta disso, o agente criou um comportamento indesejado de rotacionar infinitamente para evitar quaisquer choques. Por conta disso, o valor da recompensa negativa foi diminuída. A seção 4.3 explica melhor a situação.

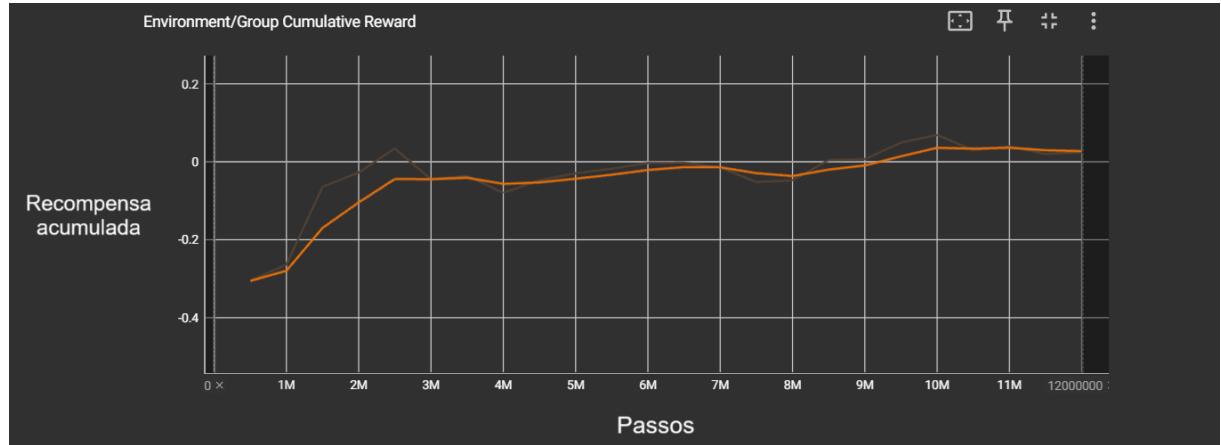
Para a segunda lição, acontece uma queda brusca no valor de recompensa, seguido com uma estabilização perto do valor de 0,13. A queda da recompensa é esperada, devido a incidência de uma recompensa negativa toda vez que o agente toma uma ação. Por ter sido aumentado um par de entrega e coleta, além do aumento na região do mapa, o tempo mínimo de realização das entregas aumenta, o que acarreta em uma média menor de recompensa ótima e um aumento na quantidade média de passos de cada episódio. Apesar da redução, ao observar na inferência, o agente demonstrou melhorar na navegação pelo ambiente, porém criou um comportamento indesejado, no qual se perde ao ter duas opções de pontos próximas e acaba girando infinitamente. Tal comportamento é melhor explicado na seção 4.3.

#### 4.1.2 Resultados da Terceira Lição

A terceira lição ocorreu por 12.000.000 passos, resultando em um tempo total de treino de 3 horas e 53 minutos. O gráfico da figura 4.5 aponta a recompensa acumulada pelo grupo,

contando que ainda só há um agente presente. A figura 4.6 demonstra a quantidade média de passos para completar o episódio.

Figura 4.5 – Gráfico de recompensa acumulada para o grupo durante a terceira lição.



Fonte: Elaborado pelo próprio autor.

Figura 4.6 – Gráfico da quantidade média de passos para se completar a terceira lição.



Fonte: Elaborado pelo próprio autor.

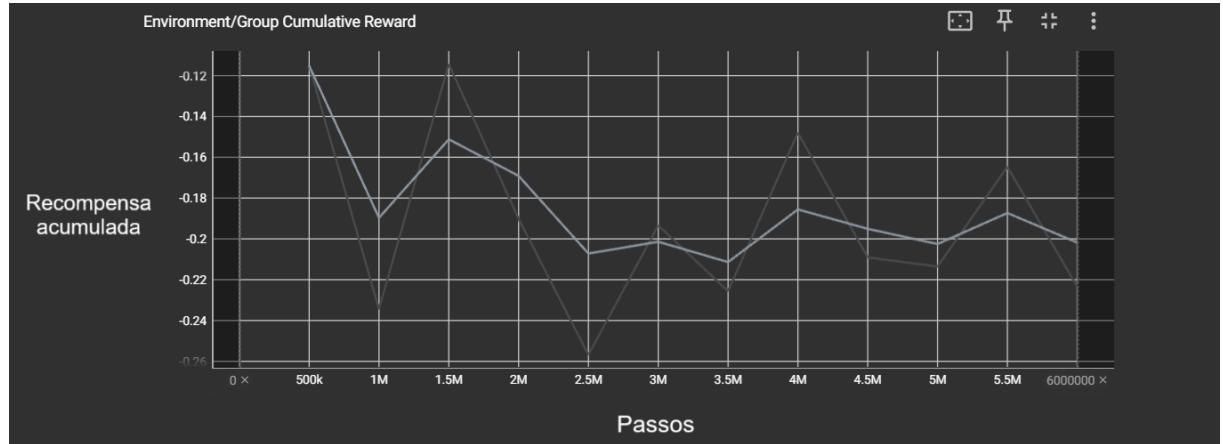
Ao observar a modelo em inferência, o agente mostrou entender a utilidade das observações sobre a distância e o alinhamento da porta. Porém, foi criado mais um comportamento emergente indesejado, no qual o agente rotaciona quase que em 360 graus, próximo a porta, até se alinhar totalmente a ela (valor de 1 na observação de alinhamento) e, a partir disso, só acelera para passar para outra sala. O comportamento é melhor discutido na seção 4.3.

#### 4.1.3 Resultados da Quarta Lição

A quarta lição ocorreu por 6.000.000 passos, totalizando 1 horas e 42 minutos de treinamento, e sua recompensa acumulada está representada na figura 4.7. A figura 4.8 demonstra a

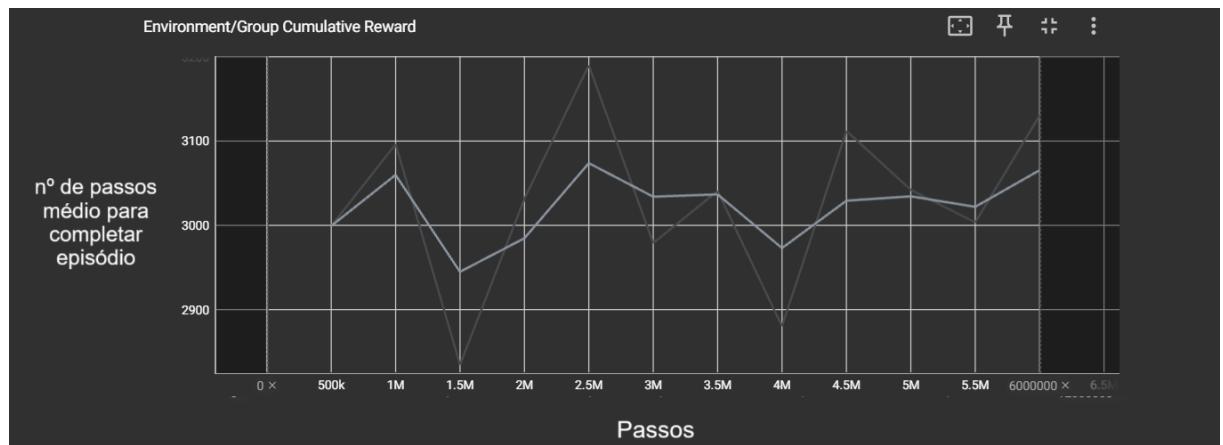
quantidade média de passos para completar o episódio.

Figura 4.7 – Gráfico de recompensa acumulada para o grupo durante a quarta lição.



Fonte: Elaborado pelo próprio autor.

Figura 4.8 – Gráfico da quantidade média de passos para se completar a quarta lição.



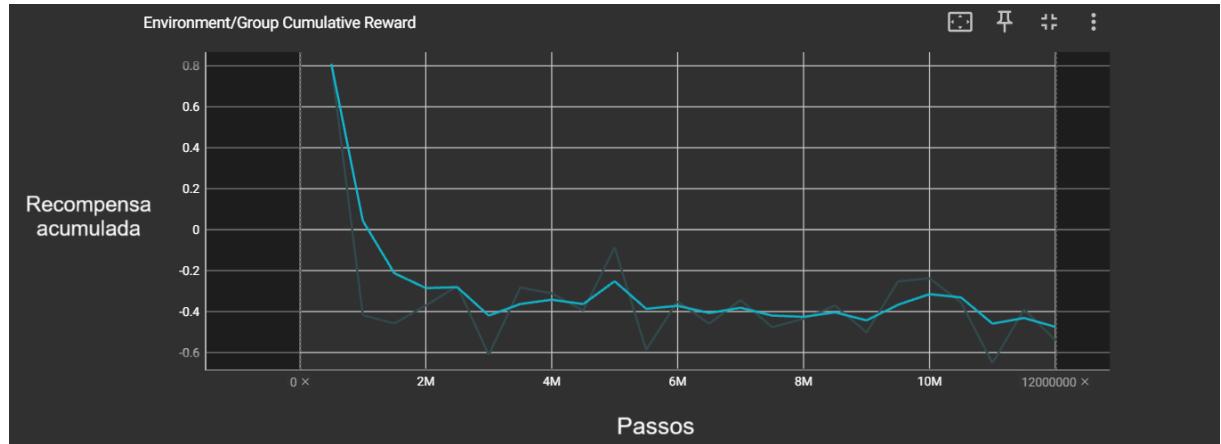
Fonte: Elaborado pelo próprio autor.

Observado a inferência, aumentar em uma sala o ambiente permitiu ao agente ter maior facilidade de navegação entre salas. Porém a maior distância gerada pela adição da sala reduziu a média da recompensa do grupo para  $-0,12$ . A partir desse momento, o problema do agente ficar indeciso entre dois pontos reduziu, porém ainda aconteceu em algumas observações. O problema de uma grande rotação para se alinhar com a porta ainda permaneceu.

#### 4.1.4 Resultados da Quinta Lição

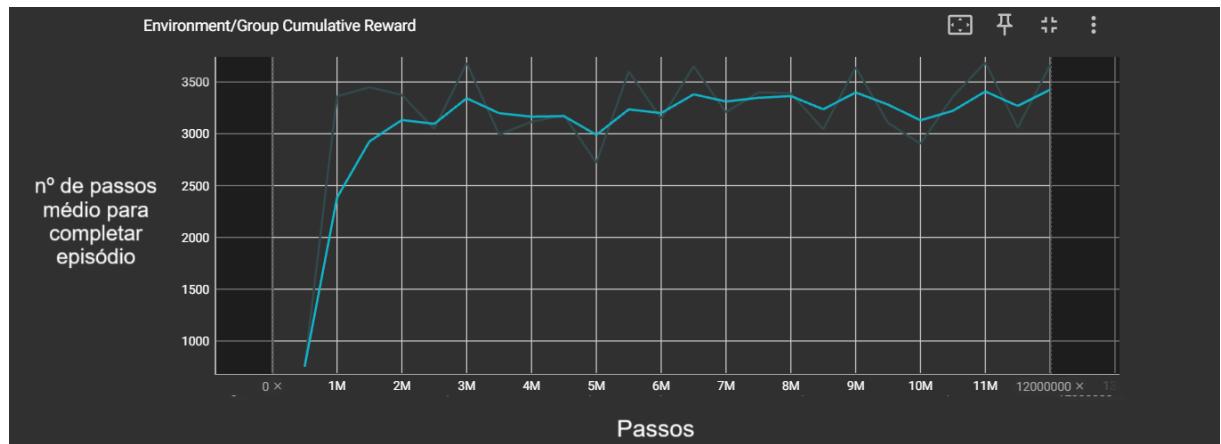
A quinta lição ocorreu por 12.000.000 passos, totalizando 3 horas e 4 minutos de treinamento, e sua recompensa acumulada está representada na figura 4.9. A figura 4.10 demonstra a quantidade média de passos para completar o episódio.

Figura 4.9 – Gráfico de recompensa acumulada para o grupo durante a quinta lição.



Fonte: Elaborado pelo próprio autor.

Figura 4.10 – Gráfico da quantidade média de passos para se completar a quinta lição.



Fonte: Elaborado pelo próprio autor.

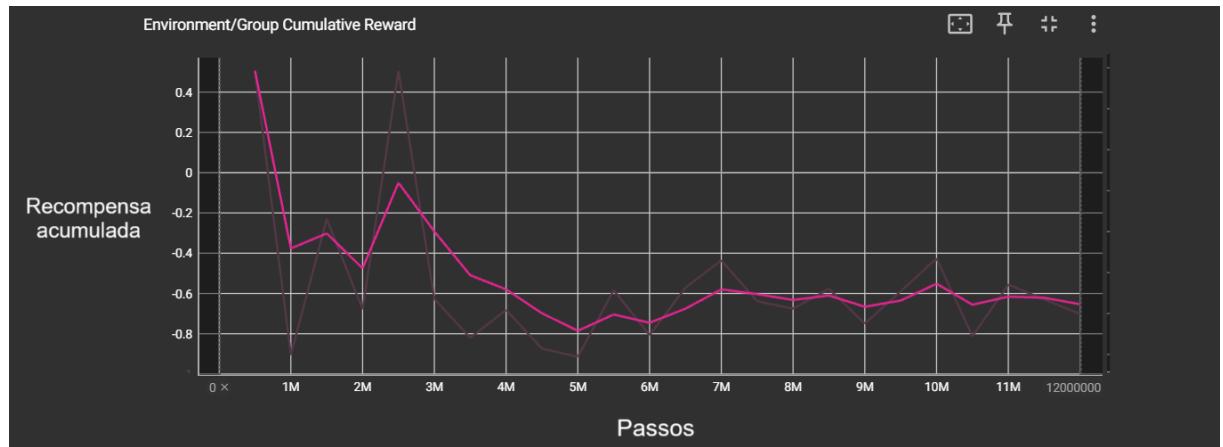
A partir da quinta lição, começa a se utilizar mais de um agente, o que requer do modelo não só planejamento de rota mas também planejamento de logística de entrega. Para a quantidade de passos realizados, o modelo ainda mostrou uma dificuldade de otimizar a logística de entrega, principalmente pelo fato de ter intensificado o comportamento indesejado de se perder ao estar entre dois pontos. Observando a inferência, havia uma incidência de agentes entre dois pontos sem decidir qual ir atrás e, caso um dos agentes conseguisse pegar um ponto, o outro imediatamente prosseguia para o ponto restante.

#### 4.1.5 Resultados da Sexta Lição

A sexta lição ocorreu por 12.000.000 passos, totalizando 4 horas e 12 minutos de treinamento, e sua recompensa acumulada está representada na figura 4.11. A figura 4.12 demonstra a

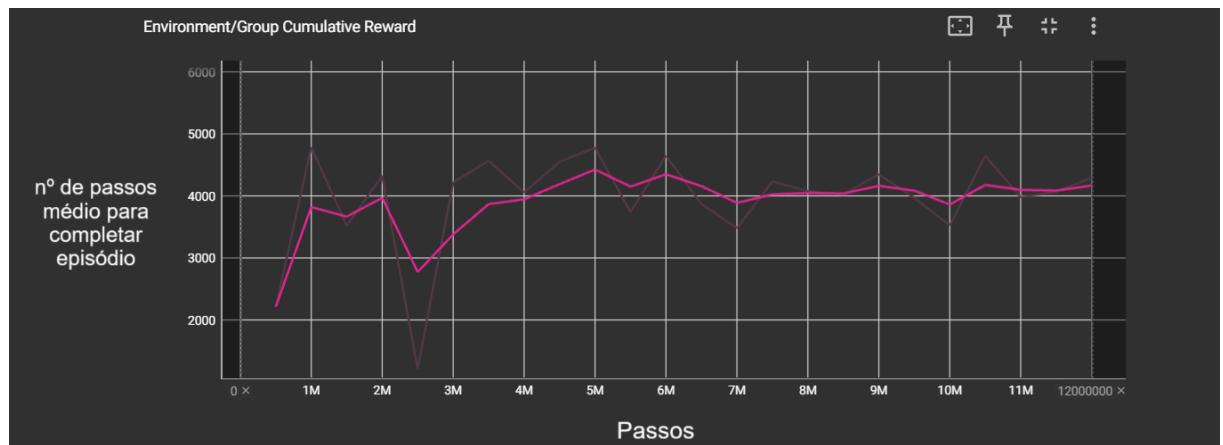
quantidade média de passos para completar o episódio.

Figura 4.11 – Gráfico de recompensa acumulada para o grupo durante a sexta lição.



Fonte: Elaborado pelo próprio autor.

Figura 4.12 – Gráfico da quantidade média de passos para se completar a sexta lição.



Fonte: Elaborado pelo próprio autor.

Devido a maior complexidade do ambiente, a partir da sexta lição o tempo de treinamento é aumentado. A carga computacional não permite realizar o treinamento a velocidade padrão observada nas lições anteriores.

O tempo de realização de cada episódio estabilizou em, aproximadamente, 4000 passos. Um aumento de 500 passos se comparado com a lição passada.

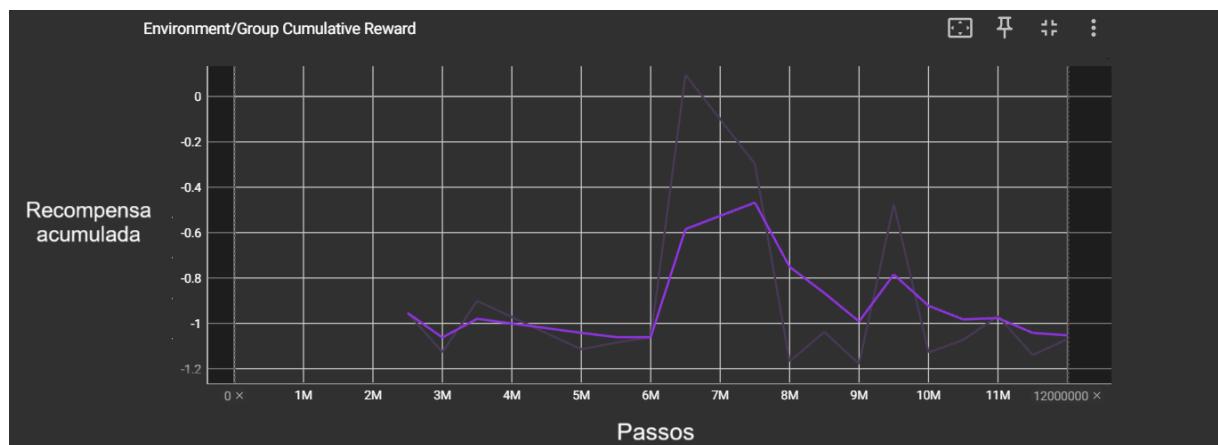
O comportamento dos agentes, observando a inferência, não atende ao necessário para entrega dos instrumentos sem ocorrer muito tempo perdido pelos agentes. Isso se dá principalmente pelo fato dos agentes manterem o comportamento de ficar perdido quando estão entre 2 ou mais pontos. Como ainda há introdução de novos pares de pontos de coleta e entrega, o efeito é

intensificado. Ademais, foi visualizado em pequena escala o comportamento de um agente esperar a passagem de outro em uma porta para não se chocar com ele.

#### 4.1.6 Resultados da Sétima Lição

A sétima lição ocorreu por 12.000.000 passos, totalizando 4 horas e 12 minutos de treinamento, e sua recompensa acumulada está representada na figura 4.13. A figura 4.14 demonstra a quantidade média de passos para completar o episódio.

Figura 4.13 – Gráfico de recompensa acumulada para o grupo durante a quinta lição.



Fonte: Elaborado pelo próprio autor.

Figura 4.14 – Gráfico da quantidade média de passos para se completar a quinta lição.



Fonte: Elaborado pelo próprio autor.

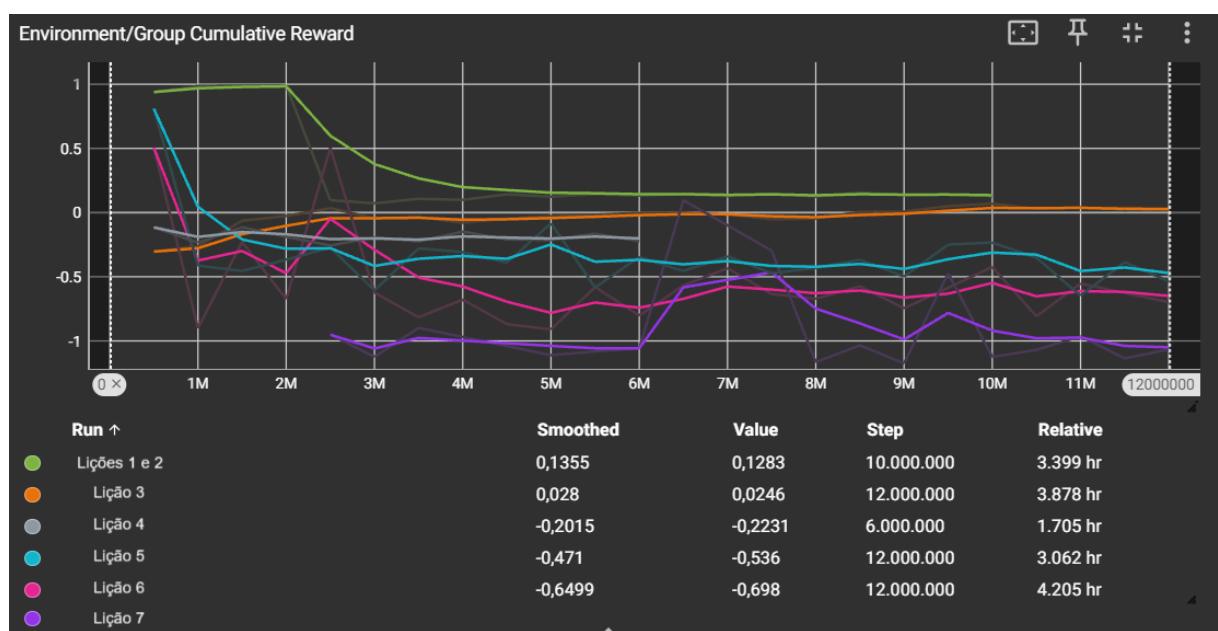
A partir desse ponto, a adição de mais salas começa a gerar um aumento exponencial no tempo de treinamento, o que poderia causar dificuldades para futuros treinamentos em ambientes maiores, caso seja usado um dispositivo de treinamento parecido com do trabalho.

O tempo de conclusão médio de episódios teve um aumento considerável, com valores médios próximos a 5500 passos, porém ainda instável.

Pela inferência, não houve melhoria nos comportamentos dos agentes. O padrão de entrega com confusão ao estar entre pontos se tornou um aspecto padronizado na política. A seção 4.4 comenta possíveis causas e possíveis soluções a esse problema.

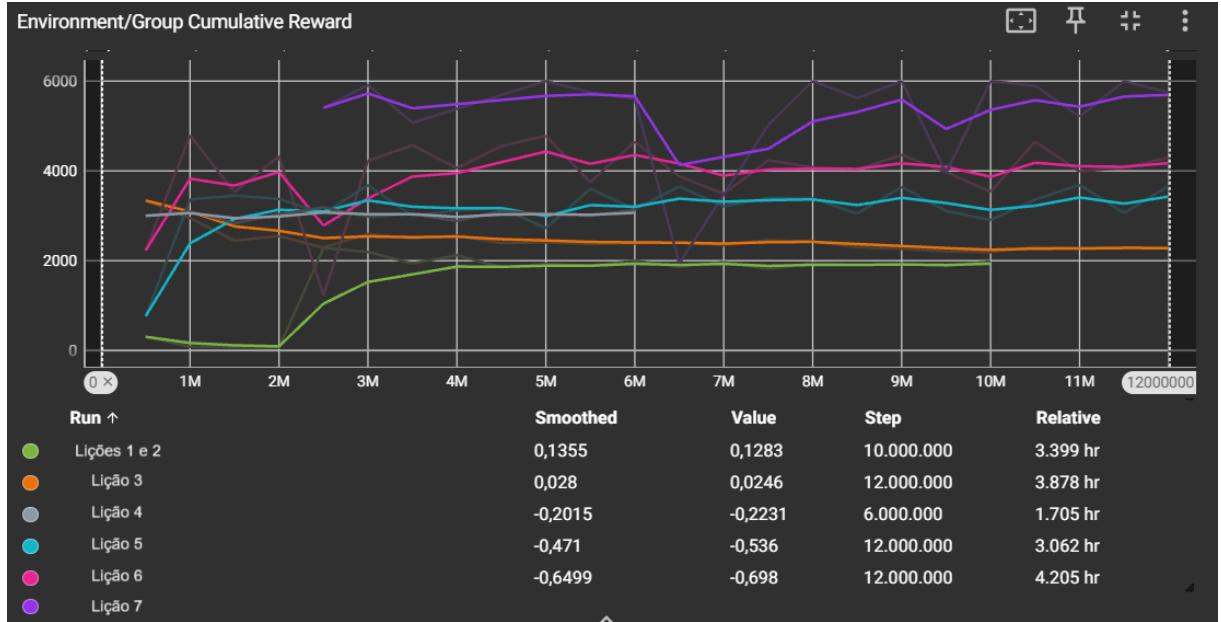
Por fim, as figuras 4.15 e 4.16 unem todas as recompensas e tempos de execução, respectivamente, das lições.

Figura 4.15 – Gráfico de recompensa acumulada para o grupo em todas as lições.



Fonte: Elaborado pelo próprio autor.

Figura 4.16 – Gráfico da quantidade média de passos para se completar cada lição.



Fonte: Elaborado pelo próprio autor.

## 4.2 Observações abandonadas

Durante o desenvolvimento do trabalho, outras observações além das mencionadas na seção 3.4.3 foram utilizadas para treinamento, porém não resultaram em boas políticas para o agente e foram eliminadas.

A primeira delas foi a **velocidade do agente**, que era passada como um vetor de duas dimensões. Sua **rotação** também foi retirada, ela era passada como um número real e se referia a apenas a o eixo z do ambiente. Ambas observações foram retiradas pois não continham o agente como ponto de referência, como no caso da rotação depender da forma que o ambiente é instanciado. Além disso, as informações por elas passadas já podiam ser determinadas no conjunto de observações variáveis.

No conjunto de observações variável, antes era apenas passado o caminho para a próxima sala e a distância do ponto de entrega ou coleta, sendo que o caminho para a próxima sala era codificada em formato one-hot. Alterar o formato dessas distâncias e adicionar o alinhamento aumentou a eficácia de navegação do agente.

Por fim, foi utilizado observação com o componente de *raycast* com uma distância muito maior para os raios, além de ter 2 raios a mais. Ter a distância muito alta para os raios resultou em um comportamento indesejado em que o agente dependia da distância de contato do raio central para saber se havia uma porta na sala ou não. Tal comportamento é melhor explicado na seção 4.3

## 4.3 Comportamentos Emergentes

Durante o treinamento, é esperado que os comportamentos desejados apareçam depois de certo tempo, porém, comportamentos inesperados e indesejados podem surgir em paralelo ou em contraposição aos desejados. Alguns desses comportamentos podem refletir um mal entendimento da política sobre como tratar certas situações, o que normalmente pode ser resolvido com diferentes abordagens de observações e parâmetros de treino. Outras podem surgir com o efeito de *reward hacking*, onde o agente acha e utiliza uma falha no ambiente ou uma forma de facilmente acumular ou deixar de perder recompensas, todos representando comportamentos contrários aos esperados. Alguns desses comportamentos observados são citados a seguir:

- **Infinitamente rotacionar:** em alguns treinamentos, principalmente situados nas primeiras lições, o agente adquiriu o comportamento de rotacionar infinitamente em sentido horário, sem nunca se movimentar do ponto que foi instanciado. A partir desse comportamento, o agente conseguiu evitar que fizesse contato com objetos que gerassem recompensas negativas. A recompensa do agente realmente foi aumentada para esse comportamento, porém representou um *reward hacking*, com um comportamento totalmente contrário ao esperado. A probabilidade de ocorrência desse problema foi reduzido ao diminuir o tamanho da primeira lição e também o valor da recompensa negativa associada ao se chocar com algum objeto;
- **Ficar perdido ao estar entre dois pontos:** este comportamento aconteceu em todos os treinamentos feitos nas primeiras lições e foi reduzido com o treinamento de lições subsequentes. Nele, o agente, ao estar posicionado entre dois pontos com função similar, tanto de entrega como de coleta, fica em um estado de confusão, sem saber qual dos pontos ir em direção. Quando isso acontece, o agente fica rotacionando e realizando alguns poucos movimentos nas direções dos pontos. Em alguns casos, o agente consegue sair desse estado, porém esse tempo varia muito. Com a otimização da política, ao realizar mais lições, o comportamento é reduzido, porém não foi completamente eliminado;
- **Rotação completa para passar de portas:** a partir da terceira lição, a qual introduz portas no ambiente, o agente adquiriu um comportamento indesejado para evitar choques ao passar por portas. Nele, o agente se coloca em uma posição próxima a porta, normalmente sendo perpendicular a ela, e desacelera até parar totalmente sua movimentação. Ao fazer isso, ele começa a rotacionar em sentido horário ou anti-horário até seu alinhamento com a porta seja quase perfeita (próxima de 1). Ao estar alinhado, o agente para de rotacionar e acelera totalmente, passando da porta sem contato com a parede. Esse comportamento permite o agente passar de portas sem receber a recompensa negativa de contato com

outros objetos, mas é uma prática que reduz a velocidade de entregas e que poderia ser otimizada;

- **Dependência de raio para identificação de portas:** este comportamento ocorreu com uma versão antiga de observação de *raycast*, explicada na seção 4.2, que continha uma distância muito maior para cada raio. No comportamento, o agente dependia da distância de contato para determinar se havia um porta ou não na sala. Primeiramente, o agente se locomovia para perto do centro da sala e, logo em seguida, rotacionava para uma direção próxima do ponto de entrega e coleta até achar um ponto específico onde o raio central tinha contato com uma parede a longa distância. Com isso, o agente conseguia entender que havia a passagem para outra sala naquela direção. Esse comportamento gerava muitos problemas em alguns templates de salas, principalmente os que haviam muitos obstáculos no centro, por isso a distância dos raios foi diminuída e a observação de alinhamento foi introduzida.

## 4.4 Discussão

Este trabalho de graduação explora o uso de ambientes simulados, utilizando-se da Unity e ML-Agents, para otimizar a entrega de instrumentos cirúrgicos em hospitais através de Robôs Móveis Autônomos (AMRs) em um ambiente simulado. O estudo aborda a criação de um ambiente hospitalar 3D utilizando a plataforma Unity, onde foram implementadas diversas lições, com aumento de dificuldade de forma incremental, para o treinamento dos agentes. Cada lição apresenta diferentes configurações de salas e complexidades de tarefas, desde a entrega em uma sala única até a navegação em um hospital de grande escala.

Os agentes foram treinados utilizando técnicas de aprendizado por reforço e aprenderam a realizar tarefas para entrega de instrumentos cirúrgicos para equipe médica, minimizando o tempo e evitando obstáculos. Além disso, foram observados comportamentos emergentes e indesejados, como a rotação infinita para evitar penalidades e a hesitação ao escolher entre pontos de coleta e entrega. Para avaliar a performance dos agentes, foram realizadas iterações de inferência, registrando-se o tempo médio das entregas em cada lição.

O método de criação de ambiente se tornou um modelo capaz de gerar salas randomizadas para diferentes tamanhos de hospitais, ainda respeitando uma arquitetura parecida de hospitais. Também permitindo a adição de novas iterações de ambientes em casos de testes paralelos, como para o processo de curriculum. Futuros trabalhos de simulações de ambientes hospitalares podem usufruir do procedimento criado, não só para realização de treinamentos utilizando aprendizado de máquina, mas quaisquer outras finalidades.

Como melhorias para o ambiente, tanto no contexto de local de treinamento com aprendizado de máquina como outras finalidades, cita-se a substituição de obstáculos padronizados com blocos primitivos por modelos 3D de materiais, equipamentos e outros objetos presentes em um hospital. Além disso, existe a possibilidade de criação de personagens humanos que utilizem o algoritmo de busca A\* criado ou métodos de navegação inteligentes que o Unity fornece para navearem pelo hospital, assim adicionando mais um tipo de obstáculo para os agentes. Apesar de não afetar o treinamento, pode-se utilizar modelos mais realista para os agentes e para os obstáculos, além de melhorar a iluminação do ambiente, de forma a se assemelhar a detalhes arquitetônicos de unidades hospitalares fornecidos por normas ou outras pesquisas, como a feita por Rocha (2008).

O uso de curriculum para implementação de desafios progressivos ao agente se mostrou muito valioso para análise de diferentes etapas de aprendizado do agente, principalmente por permitir relacionar dificuldades e comportamentos emergentes com a adição de certa observações ao agente. Por conta da forma de implementação de curriculum no ML-Agents, também foi possível utilizar parâmetros de treinamento específicos para diferentes lições, permitindo melhor

controle sobre a estabilidade e velocidade dos treinos.

Os agentes demonstraram uma ótima capacidade de realizar a navegação durante as primeiras 4 lições, evitando obstáculos, entendendo a relação do ponto de coleta com o pontos de entrega e realizando todo o percurso até as salas de operação ou departamentos de esterilização.

Observando a performance de cada lição, pode-se visualizar uma redução de qualidade ao entrar nas lições que introduziam mais de um agente. Isso pode ser associado a introdução da logística de posicionamento de mais de um agente. O modelo não se mostrou muito adaptativo para essa condição e isso pode ser atribuído a alguns possíveis fatores.

O primeiro deles são as especificações da rede, em que o comportamento desejado de logística adicionou complexidade na formação da política e, por conta disso, o tamanho da rede não era grande o suficiente para se adaptar a essa mudança. A utilização de uma rede maior pode ser uma das opções de se abordar o problema.

Outro possível fator é o tempo de treinamento, que pode ser aumentado nas lições que não convergiram em um valor médio para aumentar a probabilidade de reduzir comportamentos indesejados. O que incluiria principalmente as lições que introduziram ou intensificaram o comportamento do agente se perder entre dois pontos. Também é possível observar nos treinamentos das lições 6 e 7 que começou o tempo total de treinamento começou a ter um aumento exponencial. Caso seja feito o treinamento de plantas hospitalares maiores, o hardware utilizado deverá ser mais robusto para evitar o acréscimo do tempo de treinamento.

As observações também podem ser citadas como possíveis fatores para o desempenho reduzido. Testes com novas observações ou outros métodos de adquirir as mesmas informações mas de diferentes formas podem ser interessantes para se obter as que mais trazem benefícios para a tomada de decisões de agente. Ademais, possíveis novas atualizações do ML-Agents que introduza mais de um conjunto de observações variáveis poderiam ser utilizadas para atualizar as observações fixas de posição e alinhamento de outros agentes.

Por fim, a atribuição de recompensas ao agente pode ser melhor estudada e desenvolvida, pois podem ser mais um motivo do agente não estar conseguindo correlacionar os objetivos de planejamento de rota com sua ação. Isso se relaciona principalmente pelo fato do agente ter reduzido a relação de tempo de entrega da quinta lição em diante. Técnicas de modelagem de recompensa também podem ser estudadas e utilizadas para aumentar a eficiência do modelo (HU et al., 2020).

Tendo todos esses comportamentos dos agentes em vista, seria interessante futuras implementações com MARL que divida o problema na criação de modelos separados para navegação e para planejamento logístico. Tal divisão pode gerar não só otimização de treinamento pelo tamanho da rede não precisar ser aumentado, mas também otimização das observações

utilizadas e possíveis ações que o agente pode tomar. Isso invocaria a criação de uma rede totalmente centralizada, que cuidaria do planejamento logístico e teria como saída pontos de coleta ou entrega específicos associados a cada agente. E outra rede que, tendo apenas um ponto de coleta ou entrega em sua observação, cuidaria apenas da navegação.

## Referências

- AHMADI, E.; MASEL, D. T.; METCALF, A. Y.; SCHULLER, K. Inventory management of surgical supplies and sterile instruments in hospitals: a literature review. *Health Systems*, Taylor & Francis, v. 8, n. 2, p. 134–151, 2019.
- BALACHANDRAN, A.; LAL, A.; SREEDHARAN, P. Autonomous navigation of an amr using deep reinforcement learning in a warehouse environment. In: IEEE. 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon). [S.I.], 2022. p. 1–5.
- BITTAR, O. J.; MENDES, J. D. V. Saúde e inovação. BEPA. *Boletim Epidemiológico Paulista*, v. 16, n. 183, p. 31–36, 2019.
- BRASIL. Art. 196 da constituição da república de 1988. *Constituição da República*, Brasília, DF, 1988. Disponível em: <[https://www.planalto.gov.br/ccivil\\_03/constituicao/constituicao.htm](https://www.planalto.gov.br/ccivil_03/constituicao/constituicao.htm)>.
- BRAZ, L. G.; BRAZ, J. R. C.; CAVALCANTE, G. A. S.; SOUZA, K. M.; LUCIO, L. M.; BRAZ, M. G. [comparison of waste anesthetic gases in operating rooms with or without an scavenging system in a brazilian university hospital]. *Revista brasileira de anestesiologia*, v. 67 5, p. 516–520, 2017. Disponível em: <<https://api.semanticscholar.org/CorpusID:1150489>>.
- BROCKMAN, G.; CHEUNG, V.; PETTERSSON, L.; SCHNEIDER, J.; SCHULMAN, J.; TANG, J.; ZAREMBA, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
- CHOBIN, N.; SWANSON, K. Putting patient safety first: the sterile processing department and healthcare technology management. *Biomedical Instrumentation & Technology*, n. 1, p. 27–31, 2012.
- COHEN, A.; TENG, E.; BERGES, V.-P.; DONG, R.-P.; HENRY, H.; MATTAR, M.; ZOOK, A.; GANGULY, S. On the use and misuse of absorbing states in multi-agent reinforcement learning. *arXiv preprint arXiv:2111.05992*, 2021.
- DEEPMIND. *MuJoCo - Advanced physics simulation*. 2021. Acessado em 4 de junho de 2024. Disponível em: <<https://mujoco.org>>.
- DONG, Y.; ZOU, X. Mobile robot path planning based on improved ddpg reinforcement learning algorithm. In: IEEE. 2020 IEEE 11th International Conference on software engineering and service science (ICSESS). [S.I.], 2020. p. 52–56.
- DUGULEANA, M.; MOGAN, G. Neural networks based reinforcement learning for mobile robots obstacle avoidance. *Expert Systems with Applications*, Elsevier, v. 62, p. 104–115, 2016.
- ESCOBAR-NARANJO, J.; CAIZA, G.; AYALA, P.; JORDAN, E.; GARCIA, C. A.; GARCIA, M. V. Autonomous navigation of robots: Optimization with dqn. *Applied Sciences*, MDPI, v. 13, n. 12, p. 7202, 2023.
- FRAGAPANE, G.; HVOLBY, H.-H.; SGARBOSSA, F.; STRANDHAGEN, J. O. Autonomous mobile robots in hospital logistics. In: SPRINGER. *IFIP International Conference on Advances in Production Management Systems*. [S.I.], 2020. p. 672–679.

- FRAGAPANE, G.; HVOLBY, H.-H.; SGARBOSSA, F.; STRANDHAGEN, J. O. Autonomous mobile robots in sterile instrument logistics: an evaluation of the material handling system for a strategic fit framework. *Production Planning & Control*, Taylor & Francis, v. 34, n. 1, p. 53–67, 2023.
- FRAGAPANE, G.; KOSTER, R. D.; SGARBOSSA, F.; STRANDHAGEN, J. O. Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *European Journal of Operational Research*, Elsevier, v. 294, n. 2, p. 405–426, 2021.
- GOMES, J.; ROMÃO, M.; CARVALHO, H. Successful is/it projects in healthcare: Pretesting a questionnaire. *Procedia Computer Science*, Elsevier, v. 100, p. 375–382, 2016.
- HU, Y.; WANG, W.; JIA, H.; WANG, Y.; CHEN, Y.; HAO, J.; WU, F.; FAN, C. Learning to utilize shaping rewards: A new approach of reward shaping. *Advances in Neural Information Processing Systems*, v. 33, p. 15931–15941, 2020.
- JULIANI, A.; BERGES, V.-P.; TENG, E.; COHEN, A.; HARPER, J.; ELION, C.; GOY, C.; GAO, Y.; HENRY, H.; MATTAR, M. et al. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2018.
- LEIBO, J. Z.; HUGHES, E.; LANCTOT, M.; GRAEPEL, T. Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research. *arXiv preprint arXiv:1903.00742*, 2019.
- LOWE, R.; WU, Y. I.; TAMAR, A.; HARB, J.; ABBEEL, O. P.; MORDATCH, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, v. 30, 2017.
- LYU, X.; XIAO, Y.; DALEY, B.; AMATO, C. Contrasting centralized and decentralized critics in multi-agent reinforcement learning. *arXiv preprint arXiv:2102.04402*, 2021.
- MARCHESINI, E.; FARINELLI, A. Discrete deep reinforcement learning for mapless navigation. In: IEEE. *2020 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2020. p. 10688–10694.
- MULLER, T. Automated guided vehicles. 1983.
- NARVEKAR, S.; PENG, B.; LEONETTI, M.; SINAPOV, J.; TAYLOR, M. E.; STONE, P. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, v. 21, n. 181, p. 1–50, 2020.
- NARVEKAR, S.; STONE, P. Generalizing curricula for reinforcement learning. In: *4th Lifelong Machine Learning Workshop at ICML 2020*. [S.l.: s.n.], 2020.
- NVIDIA. *NVIDIA Isaac Sim*. 2024. Acessado em 24 de junho de 2024. Disponível em: <<https://developer.nvidia.com/isaac/sim>>.
- OROOJLOOY, A.; HAJINEZHAD, D. A review of cooperative multi-agent deep reinforcement learning. *Applied Intelligence*, Springer, p. 1–46, 2022.

- RAIMUNDO, E.; DIAS, C.; GUERRA, M. Logística de medicamentos e materiais em um hospital público do distrito federal. *Revista de Administração Hospitalar e Inovação em Saúde*, v. 12, n. 2, 2015.
- ROBOTICS, O. *Gazebo Sim*. 2024. Acessado em 4 de junho de 2024. Disponível em: <<https://gazebosim.org/home>>.
- ROCHA, M. M. B. Detalhes arquitetônicos em unidades de internação pediátrica. *Monografia apresentada ao curso de Especialização em Arquitetura de Sistemas de Saúde da Universidade Federal da Bahia*, 2008.
- ROMERO-MARTÍ, D. P.; NÚÑEZ-VARELA, J. I.; SOUBERVIELLE-MONTALVO, C.; PAZ, A. Orozco-de-la. Navigation and path planning using reinforcement learning for a roomba robot. In: IEEE. *2016 XVIII Congreso Mexicano de Robotica*. [S.l.], 2016. p. 1–5.
- RYCK, M. D.; VERSTEYHE, M.; DEBROUWERE, F. Automated guided vehicle systems, state-of-the-art control algorithms and techniques. *Journal of Manufacturing Systems*, Elsevier, v. 54, p. 152–173, 2020.
- SARIFF, N.; BUNIYAMIN, N. An overview of autonomous mobile robot path planning algorithms. In: IEEE. *2006 4th student conference on research and development*. [S.l.], 2006. p. 183–188.
- SCHULMAN, J.; WOLSKI, F.; DHARIWAL, P.; RADFORD, A.; KLIMOV, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press, 2018.
- UNITY. *Unity ML-Agents Toolkit*. 2022. Acessado em 4 de junho de 2024. Disponível em: <<https://unity-technologies.github.io/ml-agents/>>.
- UNITY. *Unity Technologies*. 2024. Acessado em 4 de junho de 2024. Disponível em: <<https://unity.com/pt>>.
- VASWANI, A.; SHAZER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. *Advances in neural information processing systems*, v. 30, 2017.
- VONGBUNYONG, S.; TRIPATHI, S. P.; THAMRONGAPHICHARTKUL, K.; WORRASITTICHAI, N.; TAKUTRUEA, A.; PRAYONGRAK, T. Simulation of autonomous mobile robot system for food delivery in in-patient ward with unity. In: IEEE. *2020 15th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*. [S.l.], 2020. p. 1–6.
- WEISS, A.; HOLLANDSWORTH, H. M.; ALSEIDI, A.; SCOVEL, L.; FRENCH, C.; DERRICK, E. L.; KLARISTENFELD, D. Environmentalism in surgical practice. *Current problems in surgery*, Elsevier, v. 53, n. 4, p. 165–205, 2016.

- WUBBEN, I.; MANEN, J. G. van; AKKER, B. Van den; VAARTJES, S.; HARTEN, W. H. van. Equipment-related incidents in the operating room: an analysis of occurrence, underlying causes and consequences for the clinical process. *Quality and Safety in Health Care*, BMJ Publishing Group Ltd, v. 19, n. 6, p. e64–e64, 2010.
- YU, C.; YANG, X.; GAO, J.; CHEN, J.; LI, Y.; LIU, J.; XIANG, Y.; HUANG, R.; YANG, H.; WU, Y.; WANG, Y. Asynchronous multi-agent reinforcement learning for efficient real-time multi-robot cooperative exploration. *ArXiv*, abs/2301.03398, 2023. Disponível em: <<https://api.semanticscholar.org/CorpusID:255546564>>.