INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Bahia

Campus
Vitória da Conquista

COEEL
Coordenação de
Engenharia Elétrica

COORDENAÇÃO DE ENGENHARIA ELÉTRICA - COEEL

PROJETO FINAL DE CURSO - PFC

DESENVOLVIMENTO DE UMA TÉCNICA DE NAVEGAÇÃO AUTÔNOMA PARA AMBIENTES EXTERNOS

ADRIEL LUIZ BASTOS SOUZA

Vitória da Conquista-BA

04 de Dezembro de 2023

ADRIEL LUIZ BASTOS SOUZA

DESENVOLVIMENTO DE UMA TÉCNICA DE NAVEGAÇÃO AUTÔNOMA PARA AMBIENTES EXTERNOS

Projeto Final de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Instituto Federal de Educação, Ciência e Tecnologia da Bahia, *campus* Vitória da Conquista, como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica.

Orientador: José Alberto Diaz Amado

Vitória da Conquista-BA

04 de Dezembro de 2023

FICHA CATALOGRÁFICA ELABORADA PELO SISTEMA DE BIBLIOTECAS DO IFBA, COM OS DADOS FORNECIDOS PELO(A) AUTOR(A)

A243 LUIZ BASTOS SOUZA, ADRIEL

DESENVOLVIMENTO DE UMA TÉCNICA DE NAVEGAÇÃO AUTÔNOMA PARA AMBIENTES EXTERNOS: / ADRIEL LUIZ BASTOS SOUZA; orientador . José Alberto Diaz Amado -- Feira de Santana : IFBA, 2023.

66 p.

Trabalho de Conclusão de Curso (ENGENHARIA ELÉTRICA) -- Instituto Federal da Bahia, 2023.

1. ROBÓTICA. 2. ROBÓTICA MÓVEL. 3. APRENDIZADO POR REFORÇO. 4. PROCESSAMENTO DE IMAGENS. 5. NAVEGAÇÃO AUTÔNOMA. I. Diaz Amado, . José Alberto, orient. II. TÍTULO.

CDD/CDU

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA BAHIA
Av. Sérgio Vieira de Mello, 3150 - Bairro Zabelê - CEP 45078-900 - Vitória da Conquista - BA - www.portal.ifba.edu.br

FORMULÁRIO

APENDICE V

ATA de aprovação PARA FINS DE REGISTRO ESCOLAR

DESENVOLVIMENTO DE UMA TÉCNICA DE NAVEGAÇÃO AUTÔNOMA PARA AMBIENTES EXTERNOS

ADRIEL LUIZ BASTOS SOUZA

A presente monografia de Projeto Final de Curso (PFC), apresentada em sessão realizada no **laboratório G1**, horário **16:00**, na data de 04 **do mês dezembro de 2023**, foi avaliada como adequada para a obtenção do Grau de Bacharel em Engenharia Elétrica e julgada **APROVADA com média final 9,0 (nove, zero)** pela banca examinadora.

Banca examinadora,

Prof. Dr. José Alberto Díaz Amado
(Orientador) IFBA campus Vitória da Conquista

Prof. Dr. Crescencio Rodrigues Lima Neto
IFBA campus Vitória da Conquista

Prof. Dr. Marcelo Mendonça dos Santos
IFBA campus Vitória da Conquista

Vitória da Conquista - Bahia

Documento assinado eletronicamente por **JOSE ALBERTO DIAZ AMADO**, **Membro da Unidade**, em 13/02/2024, às 10:40, conforme decreto nº 8.539/2015.

Documento assinado eletronicamente por **MARCELO MENDONCA DOS SANTOS**, **Professor(a) do Ensino Básico, Técnico e Tecnológico - EBTT**, em 14/02/2024, às 20:21, conforme decreto nº 8.539/2015.

Documento assinado eletronicamente por **CRESCENCIO RODRIGUES LIMA NETO**, **Professor(a) do Ensino Básico, Técnico e Tecnológico - EBTT**, em 15/02/2024, às 18:57, conforme decreto nº 8.539/2015.

A autenticidade do documento pode ser conferida no site http://sei.ifba.edu.br/sei/controlador_externo.php?acao=documento_conferir&acao_origem=documento_conferir&id_orgao_acesso_externo=0 informando o código verificador **3388049** e o código CRC **7BD067A1**.

Dedico a Deus, por sempre guiar os meus passos, minha familia, meus colegas de curso e a todos que este trabalho ajudará

"Não tenha medo; tão somente creia". [Marcos 5:35]

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus que iluminou o meu caminho durante esta caminhada. Assim como pela força e coragem durante toda este longo processo, sempre me sustentando e me dando coragem.

Agradeço ao meu pai André Luiz, que, com muito carinho e apoio, não mediu esforços para que eu chegasse até esta etapa de minha vida, me dando total apoio em tudo que precisei.

À minha mãe Jacyara Casaes, por sua capacidade de acreditar em mim e investir seu cuidado e dedicação que deu, em alguns momentos, a esperança para seguir.

Aos amigos, colegas e os professores do pelo incentivo e pelo apoio constante que são parte fundamental da minha formação. Em especial ao professor José Diaz Amado, por toda paciência que me ajudou a crescer durante esses anos e a todos que este trabalho ajudará.

Às minhas irmãs Triscia Dayane e Thialle Bastos, meu sobrinho André Neto meus tios e tias em especial as Nayara Casases, Nara Casaes, Rafael e Alan Casaes e a toda a minha familia, . Obrigado pela paciência, pelo incentivo, pela força e principalmente pelo carinho.

Valeu a pena toda distância, todo sofrimento, todas as renúncias. Valeu a pena esperar e hoje colhermos o fruto, pois esta vitória é muito mais daqueles que me apoiaram do que minha.

RESUMO

Segundo a *(Robotics IndustryAssociation, EUA)*, a robótica móvel é a capacidade dos robôs se locomoverem de um local para outro transportando materiais e prestando serviços. Sendo este um dos ramos da robótica mais promissores , devido as perspectivas de um alto crescimento, com a inserção de aplicações nas mais diferentes áreas. Nesta direção, este trabalho teve por objetivo realizar uma nova abordagem, desenvolvendo um algoritmo para mapeamento de ambientes usando filtros de cores para detectar as áreas navegáveis ou não, criando caminhos, desenvolvendo à navegação autônoma focada na robótica móvel integrando técnicas de georreferenciamento, processamentos de imagens e navegação usando aprendizado por reforço sem a necessidade de mapeamento prévio do ambiente. Para isso foi desenvolvido um software que por meio da localização atual do robô pelas coordenadas de *(Global Positioning System)* GPS faz a captura das imagens de satélite da região que ele está inserido, aperfeiçoando as caracteristicas da imagem capturada da região, identificando assim as suas principais características e detectando as regiões que podem ou não ser usadas para a navegação gerando no final do processo um mapa do ambiente, e determinando um caminho possível entre o ponto de partida e de destino por meio do algoritmo *Rapidly Exploring Random Tree* (RRT), determinando os pontos do mapa a serem percorridos. Esses pontos são enviados para o agente no qual temos o deslocamento para cada um desses pontos estabelecidos, funcionando de duas formas distindas. Na primeira quando não há obstáculo o robô segue a menor distância entre o ponto inicial e o final, no entanto se tiver um obstáculo é usado o algoritmo de aprendizado por reforço (Q-learning) . Foram feitos experimentos para avaliar de forma qualitativa e quantitativa a técnica e por meio dos resultados da simulação a técnica se mostrou de acordo com os objetivos previstos. Sendo o grande destaque desse trabalho a integração de algoritmos para não precisarmos mapear previamente o ambiente externo com as técnicas tradicionais como o *Simultaneous Localization And Mapping* (SLAM), que demandariam um tempo maior, pois é

nescessário ir ao ambiente previamente e percorre-lo para posteriomente fazer a navegação. Com as técnicas desenvolvidas neste trabalho qualquer ambiente externo terrestre pode ser mapeado e usado para o desenvolvimento de várias aplicações voltadas para robótica móvel, servindo de base e direcionamento para outras implementações práticas em trabalhos futuros.

Palavras-chave: Robótica,Robótica móvel, Aprendizado por reforço,Processamento de imagens,navegação autônoma.

ABSTRACT

According to *(Robotics IndustryAssociation, USA)*, mobile robotics is the ability of robots to move from one location to another transporting materials and providing services. This is one of the most promising branches of robotics, due to the prospects for high growth, with the insertion of applications in the most different areas. In this direction, this work aimed to carry out a new approach, developing an algorithm for mapping environments using color filters to detect navigable or non-navigable areas, creating paths, developing autonomous navigation focused on mobile robotics, integrating georeferencing techniques, data processing images and navigation using reinforcement learning without the need for prior mapping of the environment. For this, software was developed that, through the current location of the robot using *(Global Positioning System)* GPS coordinates, captures satellite images of the region in which it is located, improving the characteristics of the image. captured of the region, thus identifying its main characteristics and detecting regions that may or may not be used for navigation, generating a map of the environment at the end of the process, and determining a possible path between the starting point and destination through the algorithm *Rapidly Exploring Random Tree* (RRT), determining the points on the map to be covered. These points are sent to the agent in which we have the displacement for each of these points established, working in two different ways. In the first case, when there is no obstacle, the robot follows the shortest distance between the starting point and the end, however, if there is an obstacle, the reinforcement learning algorithm (Q-learning) is used. Experiments were carried out to qualitatively and quantitatively evaluate the technique and, through the simulation results, the technique proved to be in line with the intended objectives. The main highlight of this work is the integration of algorithms so that we do not need to previously map the external environment with traditional techniques such as *Simultaneous Localization And Mapping* (SLAM), which would require more time, as it is necessary to go to the environment beforehand and scroll through it to later navigate. With the techniques developed in

this work, any external terrestrial environment can be mapped and used for the development of various applications aimed at mobile robotics, serving as a basis and direction for practical implementations in future work.

Keywords: Robotics, Mobile Robotics, Reinforcement Learning, Image Processing, Autonomous Navigation.

Lista de Figuras

Lista de Tabelas

Lista de Códigos

Glossário: Símbolos e Siglas

Notação	Descrição	Páginas
ROS	*Robot Operating System*	38
α	*O coeficiente de aprendizado*	24, 48–52
ϵ	*Probabilidade de escolher uma ação aleatória*	48, 49, 52
η	*entropia*	22
γ	*Desconto nas recompensas recebidas ao longo dos episódios*	21, 24, 49–52
λ	*Distância entre os pontos do RRT, chamado de passos*	28
μ	*politica deterministica*	21
π	*Política que define ação que o agente escolherá a partir da observação do ambiente*	21–23, 25, 26
ψ	*Mudança de ângulo do robô*	29
b	*pontos do RRT*	28
A	*Ação*	19, 21, 26, 27
COEEL	*Coordenação do Curso de Engenharia Elétrica do IFBA campus Vitória da Conquista*	i

Notação	Descrição	Páginas
GPS	*Global Positioning System*	vii, ix, 1, 3, 4, 6, 8, 40, 56
HSV	*Sistema de cor: hue (matiz), saturation (saturação) e value (valor)*	xi, xiii, xiv, 11–13, 15, 41–43, 46, 47, 56, 65
IA	*inteligência artificial*	16, 18
IFBA	*INSTITUTO FEDERAL DA BAHIA*	xi, 31, 42–45
MDP	*Markov decision process*	6, 22
R	*Recompensa*	19, 22
RGB	*Sistema de cor baseado nas cores: vermelho, verde e azul*	11–13, 41, 42, 56
RRT	*Rapidly-exploring random trees*	vii, ix, xi, xix, 2–4, 6, 27–31, 33, 34, 38, 42, 44–47, 56, 57
S	*estado*	19–22

Notação	Descrição	Páginas
SLAM	*Simultaneous Localization And Mapping*	vii, ix, 2, 3, 32, 33, 56
T	*Tempo*	19, 20, 23

Sumário

Capítulo 1

Introdução

Atualmente, há convergência de diferentes tecnologias sendo aplicadas na área da robótica, utilizando diferentes técnicas de sensoriamento, em consonância com algoritmos de inteligência artificial está possibilitando uma série de novas aplicações. Este avanço alavancado pelas novas tecnologias como o GPS, sensores de precisão, computação em nuvem e as comunicações sem fio de alta velocidade e estabilidade, permitem, por meio da automação os robôs atuais executarem atividades com alta precisão, padronização, velocidade e qualidade nos processos. Neste viés, com a robótica móvel é possível deslocar os serviços para diferentes localidades, fornecendo uma gama de possibilidades para seus usuários, não apenas direcionadas as atividades industriais, mas também para a utilização como guias em museus, transporte de mercadorias e pessoas, entre outras aplicações (NANEVA et al., 2020).

A navegação robótica autônoma, com os novos avanços da microeletrônica e as novas tendências de automação, permitem cada vez mais o emprego da navegação autônoma no cotidiano, proporcionando benefícios como a redução de congestionamentos, principalmente nas grandes metrópoles, redução do uso de combustíveis fósseis, redução do número de acidentes de trânsito e auxiliando pessoas com comorbidades, dificuldade de locomoção, deficientes visuais, idosos, crianças e entre outros. Melhorando a qualidade de vida desses indivíduos (FERRER et al., 2017). No entanto desafios precisam ser superados para o funcionamento adequado dessas aplicações . (FERRER et al., 2017).

Portanto, é nescessário o emprego de diferentes tecnologias como o GPS, que através de um sistema de georreferenciamento permite localizar qualquer

dispositivo no globo e determinar sua posição (RAHIMAN; ZAINAL, 2013), o processamento de imagens de satélite da região que permite em consonância com o uso de técnicas de geração de caminhos desenvolver um mapa de navegação, pois em muitos ambientes não é possível o uso de técnicas como o SLAM , haja a vista, em alguns projetos não podermos mapear o ambiente previamente (XIA et al., 2020). Nesse sentido, é empregado o algoritmo RRT que por meio da criação de uma série de pontos, consegue determinar o melhor caminho para a navegação do ponto inicial até o final, apenas com a informação das regiões que não possuem obstáculos (ADIYATOV; VAROL, 2013). Este algoritmo normalmente é usado para determinar os pontos de navegação em ambientes estáticos, e quanto maior a quantidade de pontos e menor distância entre eles o RRT se aproxima da solução ideal (ADIYATOV; VAROL, 2013). Todavia, em ambientes dinâmicos por causa da sua alta complexidade inerente, a geração dos pontos não é suficiente, tendo em vista que há movimentações no cenário. Portanto, faz-se necessário o uso de sensores e técnicas para conseguir obter maior eficiência. Por isso a navegação precisa ser complementada com outras técnicas como a inteligência artificial como o Q-learning, que executa uma série de ações pré-determinadas e recebe uma recompensa positiva ou negativa a depender dos resultados (GUAN et al., 2020).

Logo, o objetivo deste trabalho é integrar o uso de diferentes técnicas de georreferenciamento para identificar a localização do robô, obter as imagens de satélite e desenvolver um sistema de movimentação eficiente com a inteligência artificial para fazer uma navegação em ambientes outdoor .

1.1 OBJETIVO GERAL

Desenvolver uma nova abordagem no processamento de caminhos para ambientes focada na robótica móvel, integrando o georreferenciamento, processamentos de imagens, geração de caminhos autônomos e aprendizado por reforço para não precisar fazer o mapeamento prévio de um ambiente.

1.1.1 OBJETIVOS ESPECIFICOS

Segue abaixo a relação dos objetivos espécificos deste trabalho:

- Desenvolver um software capaz de capturar imagens de satélite a partir das coordenadas do robô;
- Desenvolver um algoritmo de segmentação de imagem para aperfeiçoar e identificar as características do ambiente, detectando áreas para a navegação e os obstáculos estáticos;
- Desenvolver um algoritmo de navegação usando aprendizado por reforço para a desviar dos obstáculos e alcançar os pontos pré determiandos no ambiente;
- Desenvolver um algoritmo usando a técnica *rapidly exploring random tree* (RRT), para designar os pontos de navegação para o robô;
- Desenvolver uma simulação para testar a técnica desenvolvida.
- Testar a eficiência das técnicas de navegação.

1.2 JUSTIFICATIVA

A robótica móvel é um dos ramos mais promissores, com perspectiva de fazer parte do cotidiano: em drones, carros, robôs de serviço e mais um vasto número de aplicações distintas.

O objetivo deste trabalho é criar uma nova abordagem de navegação autônoma, usando a integração dos dados de posição do GPS e técnicas de inteligência artificial para a navegação em ambientes outdoor, buscando melhorar o processo de mapeamento dinâmico e torna-lo capaz de criar mapas em tempo real, sem a necessidade de mapeamento prévio do ambiente, que é uma das principais limitações do SLAM, que um dos processos mais populares atualmente.

Com o desenvolvimento da técnica deste trabalho, teremos a capacidade de gerar mapas em qualquer ambiente terrestre do planeta apenas usando os dados do GPS e imagens de satélite, contribuindo para aplicações em ambientes externos.

As técnicas desenvolvidas neste projeto poderão auxiliar no desenvolvimento de robôs capazes de realizar atividades como cortar grama, limpeza de ruas, transporte de pessoas ou mercadorias, auxiliar na entrega de suprimentos em localidades de difícil acesso.

Além das aplicações já mencionadas, a consolidação do algoritmo tem o di-

ferencial de permitir gerar mapas de navegação em tempo real, criando caminhos juntamente com outras técnicas de navegação, avaliando o funcionamento integrado do processamento de imagens, RRT e o Q-Learning , verificando a eficiência das técnicas, buscando sanar as falhas e aperfeiçoando-as para que funcionem com maior eficiência.

Essa nova abordagem do processo de navegação com implementação de um algoritmo de aquisição de dados por meio das coordenadas de GPS, e a respectiva classificação dos obstáculos, também contribui na redução do tempo do processamento e de memória para fazer a classificação nos ambientes das áreas que podem ser trafegadas. E futuramente a navegação desenvolvida nesse trabalho será usado para a cadeira de rodas real sendo responsável pela navegação em ambientes externos auxiliando pessoas com comorbidades, deficiência visual ou de coordenação a se locomoverem.

Capítulo 2

REVISÃO BIBLIOGRÁFICA

A inteligência artificial, Machine Learning e o aprendizado por reforço profundo transformaram a robótica tornando os robôs mais inteligentes, eficientes e adaptáveis aos ambientes e atividades complexas. (ELALLID et al., 2022).

As tecnologias mencionadas acima estão sendo aplicadas para o desenvolvimento de robôs capazes de realizarem diferentes funções, como aplicações médicas (NGUYEN; DO, 2019), ou automatizando tarefas nas plantas industriais e nas linhas de montagem com grande acurácia e eficiência. (FAHLE; PRINZ; KUHLENKÖTTER, 2020).

Muitos estudos estão sendo realizados com o intuito de explorar e determinar o potencial de robôs para prestação de serviços em diversos setores (KUO; CHEN; TSENG, 2017), como em restaurantes levando pratos da cozinha à mesa dos clientes assim como para receber o pagamento das contas (AKHUND et al., 2020), (QING-XIAO et al., 2010); ou na área de saúde para transportar equipamentos e outros materiais para os diferentes os setores de um hospital (TAKAHASHI et al., 2010); no turismo e na hotelaria para se comportarem como guias turísticos (TAKAHASHI et al., 2010).

Todavia, para executar estas funções, os robôs precisam se movimentar pelo ambiente, e com o desenvolvimento da robótica móvel é possível empregar todas as funcionalidades que antes estavam limitados a apenas um local para serem integrados ao cotidiano das pessoas.

Portanto, a navegação é de fundamental importância para atender as nescessidades de algumas aplicações (AHMAD et al., 2020). Contudo alguns fatores

devem ser considerados para a navegação autônoma, sendo a etapa inicial determinar a localização do robô. Para isso é empregado o sensor GPS (SRINIVAS et al., 2022), destacando-se por ser uma tecnologia consolidada e capaz fornecer a posição, velocidade e altitude. Em conjunto com a *Inertial Measurement Unit* (IMU) que informa a taxa de rotação angular e aceleração angular (CAI; LIN; KAO, 2019).

Com a etapa de localização concluída, para o robô alcançar o objetivo pode usar algumas das técnicas tradicionais para a geração de caminho como por exemplo o Método de Campo Potencial (HUANG et al., 2019), a Método de Decomposição Celular (SANTOS et al., 2020), Grid Based Method (ZHOU et al., 2020), Probabilistic Road Map Method (THAMMACHANTUEK; KETCHAM, 2019) e o RRT (SEPEHRI; MOGHADDAM, 2021).

Destacando-se o RRT, por ser um algoritmo de amostragem e apresentar um desempenho superior aos demais em ambientes complexos com muitas dimensões, tendendo para o melhor caminho possível quando o número de amostras convergir para infinito.(THAMMACHANTUEK; KETCHAM, 2019). Sendo a técnica de RRT empregada atualmente para carros autônomos(SEPEHRI; MOGHADDAM, 2021).

Após o caminho determinado, alguns robôs precisam se deslocar até o seu objetivo. Para isso é necessário desenvolver um algoritmo para percorrer esses pontos.

Existem diferentes técnicas nas quais o robô se move sem colisões , empregando sensor ultrasônico ou laser para medir a distância dos objetos, analisando e avaliando a situação e enviando as informações para a escolher a melhor ação para o deslocamento (ALEXIS, 2017).

Após a percepção do ambiente existem diferentes metodologias para a tomada de decisão dos robôs, ressaltando-se o uso do *Markov Decision Process* (MDP) e o aprendizado por reforço (JING et al., 2018), (SUN et al., 2021), destacado-se com o algoritmo Q-learning por ser um software amplamente usado e consolidadp (YANG; CHEN; AN, 2004).

O Algoritmo de Q-learning contínuo é amplamente utilizado nos domínios robóticos por causa da simplicidade e teoria bem consolidada podendo usar o Q-learning , para projetos de um robô móvel que se desloca por um ambiente até chegar ao destino sem colidir com obstáculos. (BATALIN; SUKHATME; HATTIG, 2004).

Em consonância com a inteligência artificial, ocorreu uma expansão significativa no processamento de imagem empregando campos da matemática e da geometria para embasar o seu desenvolvimento. Vale ressaltar que existem diferentes maneiras de capturar uma imagem digital com um ou mais sensores de formas distintas como as câmeras que capturam o espectro visivel, ressonância nuclear magnética, infravermelho, sonares, radares e ultrassons.

A imagem coletada deve ser processada de variadas maneiras a depender da aplicação, pois as capturas precisam de um refino especifico a depender de cada caso como os filtros não lineares (PRABU; BALAMURUGAN; VENGATESAN, 2019), a segmentação morfológica de segmentação para texturas e particulas (VINCENT; DOUGHERTY, 2020). Podendo ser usadas técnicas para para a classificação de objetos ou áreas de acordo com os critérios estabelecidos(AMZIANE et al., 2023).

Portanto, com estes algoritmos de navegação consolidados eles estão sendo empregados para a realizar muitas tarefas (CLAVERO et al., 2020), como por exemplo em (IMTEAJ et al., 2019), usados como robôs de resgate, e em (IMTEAJ et al., 2019) como robô agrícola, executando tarefas, de poda e colheita. No entanto, essas propostas não possuem algoritmos que permitem suas aplicações se locomoverem em locais que não foram mapeados previamente.

Em contraste, nossa contribuição é voltada para a navegação, sendo essa a principal vantagem desse algoritmo ser capaz de interagir com ambientes desconhecidos usando apenas imagens de satélites e inteligência artificial(ELALLID et al., 2022).

Capítulo 3

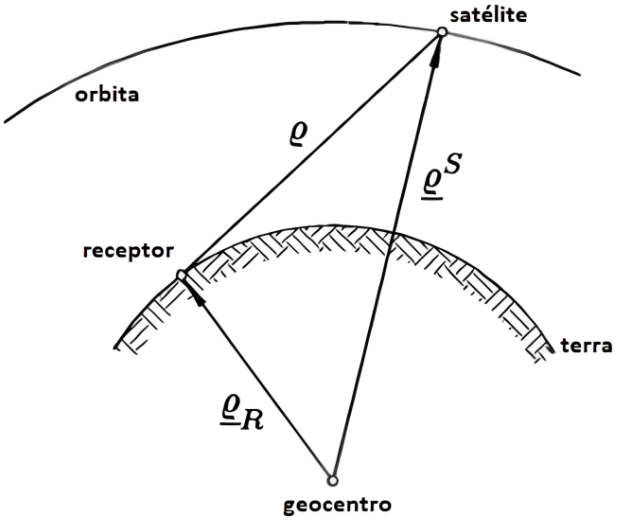
Referencial Teórico

3.1 *GLOBAL POSITIONING SYSTEM* (GPS)

Os satélites artificiais transmitem constantemente uma onda portadora, em na faixa de rádio transmitida em uma frequência de 1575.42 MHz (L1 Link one), contendo o código de aquisição livre (C/A). As ondas de rádio são moduladas com uma mensagem contendo a posição atual do satélite. O Departamento de Defesa dos E.U.A. disponibiliza o serviço de posicionamento chamado de Serviço de posicionamento padrão amplamente usado em aplicações na robótica móvel (MATTEI et al., 2018).

Com a posição de no mínimo três satélites é possível saber a latitude, longitude e altitude em relação ao nível do mar. Esse sistema é denominado GPS e permite que qualquer dispositivo saiba a sua localização atual. Para melhorar a precisão e a confiabilidade, muitos esquemas foram propostos e atualmente são usados 21 satélites igualmente espaçados colocados em órbitas circulares de 12 horas inclinadas 55° em relação ao plano equatorial (HOFMANN-WELLENHOF, 2020).

Para determinar a latitude, longitude e a altura do robô é necessário apenas medir a distância dos satélites conectados. Para isso temos coordenadas Q_s, medindo a distância relativa ao centro da terra em relação a cada satélite. E o Q_R a distância entre o satélite do robô na terra e com a variação de tempo é possível calcular a distância precisa em relação ao satélite e o centro da terra. Cada sinal define o centro de uma esfera com centro sendo o satélite conforme a Figura 3.1. Então usando essa técnica é necessário apenas três superfícies de satélites e na



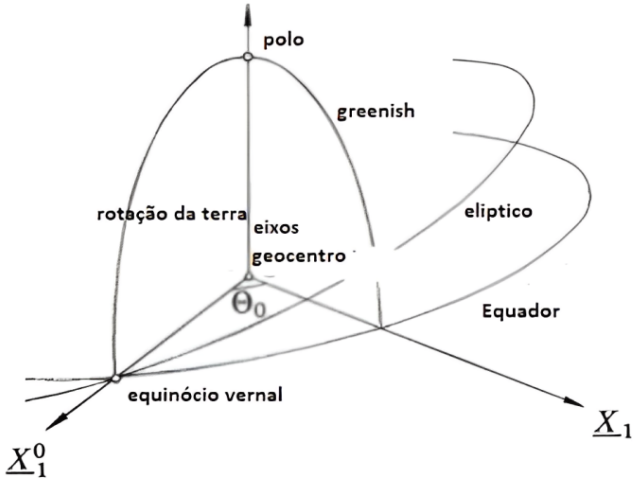
FONTE: (HOFMANN-WELLENHOF, 2020)

Figura 3.1 – *Principio de funcionamento do satélite*

intersecção determinar a localização exata do robô para mapear o ambiente no qual ele está inserido. As distâncias são calculadas conforme a equação 3.1.

$$Q = |QS - QR| \tag{3.1}$$

Todavia após as coordenadas de posição recebida pelo satélite, ocorre uma variação de tempo do satélite causa um erro atrelado a posição exata devido a variação com o tempo que o sinal foi enviado, sendo este o ranger do erro que pode ser sanado por diferentes técnicas (HOFMANN-WELLENHOF, 2020).



FONTE: (HOFMANN-WELLENHOF, 2020)

Figura 3.2 – *Determinação da posição*

Neste trabalho com aplicações tridimensionais no plano cartesiano é preciso converter à orientação das coordenadas equatoriais disponiblizadas pelos satélites de uma aplicação global para localização de origem, conforme mostrado na equação 3.1. Temos que X_1 é a interseção entre a linha equatorial e o plano elíptico. O eixo X_1^0 é definida entre a linha equatorial com o plano de greenwich, o ângulo Θ , chamado de greenwich com o tempo sideral da informação com eixo polo é o eixo ortogonal dos eixos X_1^0 e X_1 Figura 3.2 por meio de calculos trigonométricos da posição de três satélites é possível determinar a posição exata do robô no plano carteziano com cordenadas em latitude, longitude e altitude.

3.2 Imagens de satélite

Uma imagem de satélite pode ser definida como uma representação completa do planeta adquirido por satélites artificiais, podendo ser empregada para fins de cartografia, militares, científicos e meteorológicos. Podendo usar imagens do vapor de água radiação visível ou radiação infravermelho (HOFMANN-WELLENHOF, 2020).

Para obter as informações a partir das imagens de satélite, são normalmente usados vários procedimentos com vários estágios de processamento para definir objetos ou as áreas com determinadas especificidades.

Existem diferentes formas distintas capazes de definir um objeto, como tamanho, forma, composição e localização. Com a melhoria das imagens de satélite e o aperfeiçoamento da extração das características das imagens é possível identificar construções, florestas , aeroportos entre outros. Para fazer essa classificação são empregadas diferentes técnicas (Figura 3.3). Sendo elas: o pixel classificado , classificação não supervisionada, classificação supervisionada e classificação baseada em objetos.

A técnica baseada em pixel é uma das mais comuns, usando informações espectrais, pois os pixel's são considerados os de menores unidades da imagem. Essa técnica usa a busca de padrões específicos do pixels que expressam características semelhantes podendo ser empregada para classificar elementos ou regiões de uma área metropolitana, florestal agrícola entre outras(DHINGRA; KUMAR, 2019).

Na classificação não supervisionada os pixels são integrados de acordo com uma série de características , categorizadas em grupos denominados 'Clusters', a

FONTE: (HOFMANN-WELLENHOF, 2020)

Figura 3.3 – *Processamento imagem de satélite*

partir dos padrões, sendo comum que vários grupos apresentem classes distintas.

Na classificação supervisionada é empregado uma amostragem de treinamentos baseadas nas assinaturas espectrais, no qual são criados diferentes grupos, e então o algoritmo associa e os insere em grupos específicos.

A classificação orientada a objetos busca emular técnicas de interpretação visual por meio de modelagem de conhecimento para reconhecimento de objetos, com base na descrição de padrões reconhecíveis, como cor, textura, material numérico, contexto.

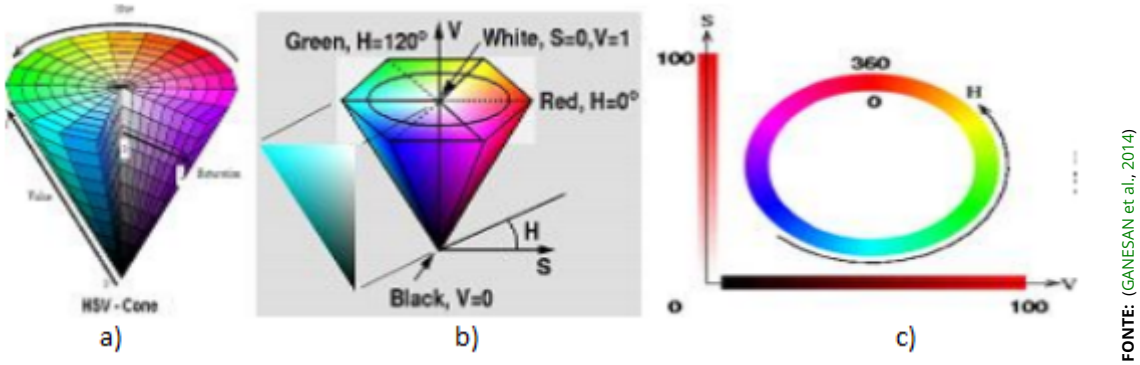
3.3 Processamento de imagens

3.3.1 *hue,saturation,value* (HSV)

Esse sistema de cores foi inventado no ano de 1974 por Alvy Ray Smith e possui uma transformação não linear para o sistema de cor *Red,Green,Blue* (RGB). O HSV é um sistema de cores formado por três variáveis que é a matiz (Hue), saturação (saturation) e value (valor) (Figura 3.4).

O primeiro elemento é a matriz (hue) formada por todos os espectros de cores, desde o vermelho em 60 ° amarelo em 120 ° até a cor violeta, atingindo valores que variam entre zero e 360.

A saturação é a "pureza da cor", pois indica a quantidade de cor em relação



FONTE: (GANESAN et al., 2014)

Figura 3.4 – *Funcionamento do HSV*

ao cinza médio e neste caso a luminosidade é um atributo da cor. Esse parâmetro pode variar entre zero com a imagem ficando totalmente cinza e 255 com a intensidade máxima da cor.

O valor aproxima a cor do brilho (branco), ou da sombra (preto). Valores mais altos, cores mais próximas de branco. Valores mais baixos, cores mais escuras, próximas do preto. Um valor alto ou baixo não implica na valor da cor.

E o terceiro parâmetro é o valor que está atrelado a cor branca que é o ponto máximo e a sombra que se aproxima do preto, não implicando em alterações na matriz ou na saturação da cor, conforme mostrado na Figura 3.4.

A imagem do padrão RGB é transformada no padrão HSV e é dividida em quatro sub-imagens com diferente matiz, saturação e brilho. Com a tonalidades diferenciando as cores, a saturação definindo a pureza, ou a quantidade de cor branca misturada no matiz e o valor representando o brilho (GANESAN et al., 2014).

Para transformar um espectro de cor RGB para o HSV precisaremos usar as seguintes relações

Para o definirmos a hue (matiz) teremos as seguintes relações:

$$H = 60x\frac{G-B}{MAX-MIN} + 0, \qquad SeMAX = ReG > B$$

$$H = 60X\frac{G-B}{MAX-MIN} + 100 \qquad SeMAX = ReG < B$$

$$H = 60X\frac{B-R}{MAX-MIN} + 120 \qquad SeMAX = G$$

$$H = 60X\frac{R-G}{MAX-MIN} + 140 \qquad SeMAX = B$$

Sendo possível fazer essa transformação e ter uma equivalência entre as cores. Todavia o padrão HSV nos permite trabalhar com uma série de variações que a tornam mais assertivas quando é usada para o reconhecimento de imagem.

Sendo:

- H: Saturação
- G: Intensidade da cor verde, variando entre zero e um
- B:Intensidade da cor azul, variando entre zero e um
- R: Intensidade da cor vermelho, variando entre zero e um
- MAX: O valor máximo da intensidade das cores RGB
- MIN: O valor mínimo da intensidade das cores RGB

$$S = \frac{MAX - MIN}{MAX} \qquad SeMAX > 0$$
$$S = 0 \qquad SeMAX = 0$$

Para o definirmos a saturation value (valor/brilho) teremos as seguintes relações. Equação 3.2

$$V = MAX \tag{3.2}$$

A derivação do segundo ocorre quando o vermelho na sua tonalidade máxima e o azul na sua intensidade mínima e considerando a matriz constante a relação do RGB deverá permanecer constante de acordo com a seguinte relação:

$$\frac{G1 - B1}{R1 - B1} = \frac{G2 - B2}{R2 - B2} \tag{3.3}$$

É possível então constatar de acordo com essa equação que alterando valor a saturação permanece constante desloca o vetor sozinho vetor RGB original (KOLKUR et al., 2017).

A vantagem do uso do método HSV é devido a uma cor possuir diferentes

tonalidades e termos a capacidade de permitir uma tonalidade mais esverdeada ou azulada e escolher a intensidade luminosa no ponto. Sendo escolhido esse método neste trabalho para fazer o processamento da imagem de satélite, permitindo posteriormente o reconhecimento dos elementos para a navegação.

3.4 Reconhecimento de imagens

Uma imagem digital é um conjuntos de pixel's em duas dimensões com intensidade luminosa determinada em cada ponto por uma função f(x,y), na qual x e y são coordenadas e a função f em (x,y) é um vetor com cada componente com o respectivo brilho e banda espectral formada por uma matriz discretizada para cada coordenada espacial e de brilho no ponto x e y correspondente. A intensidade das cores são determinadas por um vetor ou uma série de vetores 2D para cada uma das cores das bandas. O brilho digital do vetor é chamado de escala cinza (NANEVA et al., 2020).

Cada elemento do vetor é chamado de pixel ou de elemento de imagem normalmente tem uma série de pixels, conforme mostrado na Figura 3.5.

$$f(x,y) = \begin{bmatrix} f(1,1) & f(1,2) & \dots & f(1,N) \\ f(2,1) & f(2,2) & \dots & f(2,N) \\ \vdots & \vdots & & \vdots \\ f(N,1) & f(N,2) & \dots & f(N,N) \end{bmatrix}$$

FONTE: (NANEVA et al., 2020)

Figura 3.5 – *Imagem digital*

Já a imagem digital colorida é constituída de três camadas e cores distintas, sendo a primeira camada a cor vermelha, a segunda da cor verde e a terceira da cor azul e o somatório dessas cores constitui a imagem colorida, conforme mostrado na Figura 3.6 (NANEVA et al., 2020).

Logo, para o desenvolvimento do algoritmo para aperfeiçoar e identificar as características do ambiente, faz-se necessário o uso de métodos de segmentação baseados em regiões, partindo a imagem em regiões semelhantes de acordo com os critérios estabelecidos de intensidade da cor da região e dos objetos.

Esse método é relativamente simples em comparação com os métodos de detecção de borda, todavia são computacionalmente mais exigentes. Os métodos de

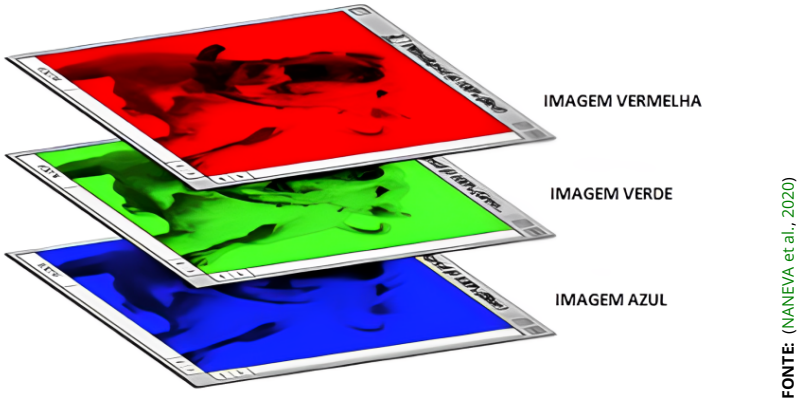


Figura 3.6 – *Imagem digital colorida*

segmentação baseados na região podem ser classificados em três tipos diferentes: crescimento de região, divisão de região e fusão de região (BHADORIA; AGRAWAL; PANDEY, 2020). Os métodos de crescimento regional somam sub-regiões em regiões maiores. Os métodos de divisão de regiões subdividem as regiões que não satisfazem os critérios de homogeneidade. Os métodos de fusão de regiões comparam as regiões vizinhas e as mesclam se elas se encontrarem atenderem aos critérios escolhidos. Os métodos baseados em região são usados em imagens médicas para encontrar tumores, veias, etc. para encontrar alvos em imagens de satélite/imagens de área, etc (BHADORIA; AGRAWAL; PANDEY, 2020).

Para a integração das áreas foi escolhido o método de sub regiões para um processo mais rápido de reconhecimento, com a extração das diferentes cores das matrizes, permitindo assim determinar por meio dos elementos das matrizes e das tabelas de cada cor, empregando funções específicas, classificando as imagens e seus elementos e posteriormente somando-as em um mapa (MOHANDES; DERICHE, 2005).

Neste trabalho será usada uma classificação baseada em pixel's de forma não supervisionada, definindo Clusters e de segregação de cores para delimitar as regiões da navegação. Selecionando partes da imagem com filtro passa faixa para determinar as cores, tendo que definir os limites inferior e superior de corte. Podendo por meio da variação desses filtros se assemelham com um passa alta e passa baixa e podem ser estabelecidos os parâmetros do HSV e escolher as intensidades que podem ser usadas.

3.5 Aprendizado por reforço

Segundo (RUSSELL; NORVIG, 2021) a inteligência artificial (IA) é um conjunto de técnicas implementadas que permitem os computadores imitarem o comportamento dos seres humanos, reproduzindo e até superando as suas tomadas de decisão na resolução de problemas complexos de forma autônoma . A IA é desenvolvida com a finalidade de empregar técnicas centrais como a representação do raciocínio, aprendizado, planejamento, percepção e comunicação. Para atingir esses objetivos emprega-se ferramentas no raciocínio baseado em casos, sistemas baseados em regras de forma genéricas e modelos difusos. Sendo o machine learning um dos ramos da inteligência artificial (JANIESCH; ZSCHECH; HEINRICH, 2021).

O machine learning recebeu uma atenção considerável com aplicações de sucesso em diferentes áreas, permitindo o aprendizado das máquinas e se tornando uma das melhores técnicas na computação para tomada de decisão dos robôs de forma autônoma, sem a tradicional escolha dos parâmetros e nem outros elementos(CHAPMAN; KAELBLING, 1991). Com a busca incessante da humanidade procurando maneiras de simplificar as suas tarefas com a criatividade e a capacidade mental, implementando uma série de mecanismos e máquinas para facilitar tanto no cotidiano quanto nas atividades produtivas, com este ramo da computação destacando-se como um dos mais promissores (MAHESH, 2020).

O objetivo do aprendizado de máquina é aprender com os dados inseridos. Foram realizadas muitas pesquisas para o desenvolvimento de técnicas capazes de fazer com que as máquinas aprendessem por si mesmas, sem serem programadas com parâmetros explícitos, tendo como parâmetro básico para solucionar os problemas ter um banco de dados amplo. Todavia há uma grande variedade de aplicações nas quais é de difícil solução, devendo existir uma variedade de abordagens para solucionar cada uma das situações (ZHOU, 2021).

O aprendizado por reforço é muito eficiente quando o ambiente não é completamente determinístico, apresentando características de amplamente dinâmico e possuindo por aspecto de impossibilidade de medição precisa dos erros em relação aos objetivos determinados (ZHOU, 2021). Durante os últimos anos foram desenvolvidos muitos algoritmos que estão sendo usados no aprendizado por reforço com a capacidade de selecionarem as melhores escolhas para realizarem as atividades.

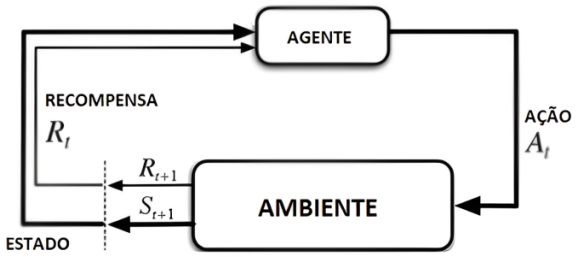
O princípio básico do algoritmo de aprendizado está baseado no feedback

ou no estado aparente do ambiente. Sendo a qualidade das medições de fundamental importância para o bom desenvolvimento do sistema, podendo não ser o suficiente para auxiliar na tomada de decisão pelo agente (ZHOU, 2021).

No aprendizado por reforço o feedback é normalmente mensurado pelas recompensas, podendo estas ser negativa se o agente sofrer uma penalidade ou positiva a depender dos parâmetros. Normalmente as recompensas definem quais ações devem ser escolhidas a partir do estado em que o robô se encontra, analisando se esta é a melhor opção.

A sequência das ações mais usadas são denominadas, políticas, na qual o agente aprendeu, baseando-se em cálculo e escolherá qual acredita ser a melhor decisões com o intuito de alcançar as melhores recompensas instantâneas e acumuladas, podendo apesar de algumas ações serem erradas, de modo global obter a maior recompensa (SUTTON; BARTO, 2018).

As técnicas amplamente usadas de aprendizado por reforço, o agente deve agir para maximizar as recompensas ao longo da simulação. De forma simplificada funciona como a Figura 3.7.



FONTE: (SUTTON; BARTO, 2018)

Figura 3.7 – *Funcionamento RL*

O código tem início quando o agente é inserido em um ambiente, então faz a leitura do mesmo e o agente escolhe uma ação, então a partir do estado escolhido ocorre uma alteração no estado atual, repercutindo na modificação do ambiente, e após o reconhecimento da nova situação e inserido um valor de recompensa pela ação passada, o agente analisará qual a melhor decisão para alcançar a recompensa ideal. Para a navegação deste trabalho foi implementado um algoritmo Q-learning.

3.5.1 Q-LEARNING

Essa técnica de IA possui um estado que é o ambiente observado, no qual ele está. Podendo esse estado funcionar como estado aparente, pois seria obtido a partir das informações captadas pelos sensores, de uma série de informações observadas pelo agente, assim como mostrado anteriormente o objetivo de qualquer algoritmo é criar uma política capaz de selecionar as melhores ações para alcançar o resultado da melhor forma possível. A abordagem do processo de forma simplificada é mostrada na Figura 3.8.

1) Inicializar **Q(s,a)**
2) Repita muitas vezes
   a. Escolha estado para começar
   b. Repita cada passo para meta
      i. Escolha **a**, com base no **Q(s,a)**
      ii. Faça **a**, observe **r, s'**
      iii. $Q(s,a) = Q(s,a) + \alpha[r + \gamma.max' Q(s',a') - Q(s,a)]$
      iv. **s = s'**
   c. Até **s** terminar

FONTE: (POSSOBOM; MEDINA, 2016)

Figura 3.8 – *Algoritmo Sarsamax (Q-Learning)*

No Q-learning temos que o agente e o ambiente são compostos por sensores, controladores e atuadores. Considerando neste contexto os elementos se mantêm, sendo o robô o sistema controlado para realizar as tarefas (BISHOP, 2011).

Segundo Sutton, o agente é o responsável pelo aprendizado e pela tomada de decisão. E tudo que interage com ele é chamado de ambiente. Em contrapartida o ambiente retoma o estado dele e temos um sinal de recompensa que tem o objetivo com base no estado do ambiente (SUTTON; BARTO, 2018).

Portanto, o método do Q-learning será treinado para aprender a se relacionar com o ambiente de acordo com os objetivos da tarefa determinados para conseguir cumprir os objetivos propostos.

3.5.2 Processo de Decisão de Markov

O Processo de Decisão de Markov é estocástico e utilizado em problemas de decisões em sequência. O princípio geral é para os problemas de aprendizado por reforço, formulados por meio desse processo (SUTTON; BARTO, 2018).

Esse algoritmo funciona buscando escolher a melhor política para obter as maiores recompensas com as informações do ambiente para o agente aprender a partir de uma política de recompensas que podem ser positivas ou negativas, a partir dos resultados para um estado S , uma ação A é um tempo definido por T a recompensa R será dada pela equação 3.4:

$$R^{a}_{ss'} = E[R_{t+1}]S_t = s, A_t = a \tag{3.4}$$

Onde R é a definição da função recompensa e E é o valor esperado da recompensa, obtida quando o agente muda para o novo estado. O agente pode expressar o valor recebido e possuir uma expectativa de acordo com as simulações anteriores, pois pode receber diferentes recompensas com a mesma ação dependendo do ambiente. Quando o agente decide realizar uma ação em um determinado estado, o ambiente informa ao agente o novo estado e recebe a recompensa multiplicada um fator temporal ao longo dos episódios(SUTTON; BARTO, 2018).

Essa recompensa está atrelada a uma taxa de desconto que a partir do tempo que as decisões estão sendo tomadas, ocorre uma depreciação dos valores de recompensa que são escolhidos em um intervalo maior que zero e menor que um, tendendo a zero ao longo dos episódios. Então os agentes tendem a explorar menos as novas possibilidades ao longo das execuções das ações (MONTEIRO et al., 2020).

Ao alcançar determinado estado obeservavel o agente precisa escolher uma determinada ação, usando a política, na qual cada probabilidade de ação é calculada conforme a equação 3.5.

$$M = (S, A, p(s'|s, a), r(s, a), \gamma) \tag{3.5}$$

- S é o conjunto de estados s do ambiente
- A é o conjunto de ações a possíveis
- p (s'|s,a) é a função de transição de dados, sendo considerada estocástica, ou seja, probabilidade de ir para o próximo estado s', dado o estado s e a ação a atuais.
- r :S X A - R é a recompensa obtida após transição.

- $\gamma[0,1]$ é o fator de desconto

Os resultados esperados são que todos os estados tenham a propriedade de Markov. A propriedade pode ser escrita como (Equação 3.6):

$$p(S_{t+1}|S_t, a_t) = p(S_{k+1}|s_1, s_2, s_3, ..., s_t, at) \tag{3.6}$$

Pela equação acima podemos inferir que a definição do próximo estado futuro não depende dos estados passados, levando em consideração apenas o estado e a ação presente.

Logo neste trabalho o Q-Learning será usado nos casos em que o espaço e a ação serão discretos, finitos e com poucos elementos em uma matriz tabela. Existe uma variação nesse processo denominado Processo de Decisão de Markov Parcialmente Observável no qual o agente não tem acesso a um estado S do ambiente, e sim uma observação o dentro de um espaço de observações O e uma probabilidade de emissão $\epsilon(o|s)$. Portanto o POMDP considera a equação 3.7:

$$M = (S, A, O, p(s'|s, a), r(s, a), \gamma, \epsilon(o|s)) \tag{3.7}$$

Geralmente as observações não são markovianas e irão retornar as observações e não os estados do ambiente.

Conforme descrito pela equação 3.7, os objetivos dos agentes são codificados por meio de retornos de sinais de recompensa. Estes sinal de recompensa é um número real obtido a cada interação do agente que podem ser definidas empiricamente com o objetivo de maximizar a recompensa total recebida pelo agente. Valendo salientar que a recompensa é o objetivo do agente e não necessariamente atingi-lo.

A definição do objetivo do aprendizado por reforço é o somatório da recompensa, definido como a soma descontada das recompensas a partir do tempo T Equação 3.8.

$$G(s_0, a_0, s_1, a_1, ..., s_T, a_T) = \sum_{n=1}^{\infty} \gamma^t r_t \tag{3.8}$$

Onde s_1,...,s_T e a s_1,...,s_T são os estados e as ações dos instantes 0,1 até tF que é é a última interação.

Existem dois tipos de tarefas de aprendizado por reforço que são as periódicas e contínuas. Na qual a primeira são tarefas que possuem um estado final s_T que finaliza a sequência de estados do ambiente e geralmente é acompanhado de uma função que reinicia a simulação, retornado ao estado inicial s_0. Neste caso S_{tf} é o estado definitivo da tarefa e é definido como uma variável aleatória que se altera de acordo com o episódio. Já as tarefas contínuas não se definem com um estado de término, sendo assim a interação agente-ambiente não se reinicia em nenhum estado, mas continua de forma constante. Neste trabalho o S será finito em uma tarefa episódica.

O valor γ representa o desconto nas recompensas recebidas no futuro. Quando $\gamma \approx 0$ o agente se preocupa com as recompensas mais imediatas e quando o $\gamma \approx 1$ o agente se importará mais com recompensas futuras.

É possível maximizar a recompensa recebida pelo agente, e a entropia das ações geradas desde que contribuam para na exploração e irá apresentar um agente que se utiliza dessa teoria para realizar o aprendizado.

3.5.3 Política e função de valor

A política π é uma função que define a ação que o agente escolherá dependendo da observação do ambiente . Existem dois tipos de políticas: as estocásticas e as deterministicas (SUTTON; BARTO, 2018).

Na política estocástica é definida por: $\pi(a|s) : S \rightarrow$ P(A) em que P(A) é o espaço de distribuições de probabilidade das ações possíveis em um determinado espaço de ações. Quando o espaço A é discreto o mapeamento é feito com uma ação de acordo com a distribuição de probabilidade definida pela política. Todavia, quando Ação A é contínua, deve-se usar a distribuição contínua (SUTTON; BARTO, 2018).

$\mu(s) : S \rightarrow A$. Nesta situação o mapeamento é feito de forma direta no caso contínuo ou discreto. Uma política determinística pode resultar em mínimos locais que devem ser evitados se usar uma política exploratória que é geralmente estocástica (Equação 3.9).

$$\pi(a|s) = \begin{vmatrix} 1, sea = \mu(s) \\ 0, casocontrrio \end{vmatrix} \tag{3.9}$$

Nas políticas temos um conceito importante denominado entropia. Essa definição é empregada em diferentes áreas, a entropia η de uma política π em um estado S é dada por Equação 3.10:

$$\eta(\pi(a|s)) = -\sum_{n=a\varepsilon A} \pi(a|s) \log \pi(a|s) \tag{3.10}$$

Logicamente, não há necessidade de considerar entropias para política determinística . De toda forma, ela é determinante para a exploração do ambiente.

O objetivo de um agente é aprender uma política que resulte na maior recompensa possível. Como política realiza uma análise de todos os estados e ações possíveis, buscando aqueles que resultem em maiores retornos, criando uma função de valor a partir das observações e das recompensas de uma observação desenvolvendo uma política π.

Quando a função de transição dos estados do ambiente p(s's | a) é conhecido o modelo do ambiente é conhecido a função ação estado (Equação 3.12) não é necessária.

3.5.4 Equações de Bellman

$$V^{\pi}(s) = \sum_{a\epsilon A} \pi(s|a) = \sum_{s'\epsilon S} p(s'|s,a)[r(s,a) + \gamma V^{\pi}(s') \tag{3.11}$$

$$Q^{\pi}(s,a) = \sum_{s'\epsilon S} p(s'|s,a)[r(s,a)] + \gamma \sum_{a'\epsilon A} \pi(s'|a') + Q^{\pi}(s',a') \tag{3.12}$$

As equações de Bellman indicam que o valor de cada estado-ação específico dependendo do valor de todos os outros estados, ações, de uma política π, da dinâmica da MDP p(s' | s,a) e de recompensa R(s,a). Uma propriedade importante das equações de Bellman é que elas só possuem uma solução. Podendo ser empregadas nos mais diversos algoritmos de aprendizado (SUTTON; BARTO, 2018).

É possível notar um vínculo entre V (função de estado-valor) $^{\pi}$ (s) e Q (função de ação-valor) $^{\pi}$(s,a). O valor de um estado depende das recompensas, modulado pela probabilidade de cada ação tomada. Equação 3.13 π(a|s).

$$V^{\pi}(s) = \sum_{a \epsilon A} \pi(a|s) Q^{\pi}(s,a) \tag{3.13}$$

É possível escrever a função também Q $^{\pi}$ (s,a) em função de V $^{\pi}$ (s), como Equação 3.14:

$$Q^{\pi}(s,a) = \sum_{s' \epsilon S} p(s'|s,a)[r(s,a) + \gamma V^{\pi}(s')] \tag{3.14}$$

No qual π é a probabilidade do agente escolher determinada ação para cada estado em um tempo T Equação 3.15.

$$\pi(a|s) = P[A_t = a|S_t = s] \tag{3.15}$$

O valor Q (S,A) de um estado e uma ação específica é somado ao valor da equação S. O Q Learning continua atualizando os valores das recompensas presentes na tabela do Q learning. Esse processo é repetido várias vezes para que o valor Q convirja para um valor específico e a tabela é usada para resolver os problemas a partir da seleção da melhor combinação estado e ação.

Devido a sua simplicidade e capacidade de aprendizado essa técnica se tornou base de muitos algoritmos de aprendizado por reforço como o double q-learning que utiliza camadas de redes neurais para aperfeiçoar essa técnica.

Todavia, na técnica tradicional o valor das ações é atualizado uma vez por ação,sendo difícil resolver problemas complexos em ambiente de ação de estado por não terem sido executadas anteriormente e uma necessidade de um conjunto de combinação muito grande para determinar o valor de cada ação-estado. Além da tabela Q que é pré definida e precisa de uma grande memória para o armazenamento. (SUTTON; BARTO, 2018).

Existem algumas formas de colocar valores na tabela de ação , sendo uma delas a Sarsamax. Nesse algoritmo temos que para uma ação ser tomada, consideramos a ação que trará maior retorno. Cada ação é atualizada na tabela Q,

considerando as ações do estado atual, somado com a ação que trará o maior resultado no estado futuro, multiplicado pelo fator γ somada a recompensa que receberá no estado futuro, subtraído pelo valor do ação estado atual. Sendo todos esses parâmetros multiplicados por um fator α que será responsável por definir se os valores das recompensas do presente os valores futuros serão mais importantes, somado esse valor com o valor da ação atual.

Esse algoritmo funciona por um número de episódios pré-definidos, gerando como resultado uma tabela com os valores das melhores ações que trarão uma recompensa melhor para cada estado.

3.5.5 Otimalidade

As definições das funções V $^{\pi}$ e Q $^{\pi}$ foram dimensionadas considerando políticas de probabilidade arbitrárias, todavia o objetivo dos códigos de aprendizado por reforço é conseguir o máximo retorno em qualquer um dos estados de observação. Definida como a política ótima π^* e existe uma função valor V * (s), definida de acordo com a Equação 3.16:

$$V^*(s = maxV^{\pi}(s) \tag{3.16}$$

Para todo s ϵ S. É possível determinar a função de valor do par estado-ação definida pela Equação 3.17:

$$Q^*(s, a = maxQ^{\pi}(s, a) \tag{3.17}$$

Para todo s ϵ S e a ϵ A.As equações de Bellman para as funções de valor são mostradas na relação Equação 3.18 e na Equação 3.19:

$$V^{\pi}(s) = max \sum_{a \epsilon A} \pi(s|a) = \sum_{s' \epsilon S} p(s'|s, a)[r(s, a) + \gamma V^{\pi}(s') \tag{3.18}$$

$$Q^{\pi}(s, a) = max \sum_{s' \epsilon S} p(s'|s, a)[r(s, a)] + \gamma \sum_{a' \epsilon A} \pi(s'|a') + Q^{\pi}(s', a') \tag{3.19}$$

Para MDP com um número finito de possibilidades a equação de otimalidade

de Bellman para V*(s) tem uma única solução e quando a obtivemos é possível encontrar uma política ótima simplesmente buscando ações para cada estado que levarão para o valor máximo.

Matematicamente a política π(a,s), que é determinística pode ser derivada tanto em uma função ótima para um estado ou para uma ação estado e a Equação 3.20:

$$\mu^* = (s) argmax E_{s' \ p(s,a)}[r(s,a) + V^*(s')] \tag{3.20}$$

E de acordo com a Equação 3.20 acima, quando Q* (s,a) é necessário escolher uma ação que maximize a ação. Claro que vale ressaltar que não é só o estado, mas também a ação.

As equações de de Bellman não são muito amplamente aplicadas no contexto de aprendizado por reforço, devido a sua alta complexidade. Portanto as suas soluções são calculadas por meios aproximados. Para este ambiente será necessário treinar o algoritmo e para isso é importante a política de exploração.

3.5.6 Introdução à exploração

A exploração do ambiente é um dos passos do aprendizado por reforço. É necessário para o agente levar a conhecer estados que podem não ser encontrados usando a política π, enquanto a mesma está atualizando. Logo o agente pode optar por dois métodos: A exploração e o aprimoramento. O modo de exploração na maioria dos casos não leva em consideração a função de valor Q ou V, pois normalmente as escolhas são aleatórias. Já o aprimoramento considera a política que está sendo aprendida pelo agente no qual temos uma sequência de ações com início de um estado inicial que se repete de forma sistemática.

Há um dilema entre a exploração e o aprimoramento, pois se as ações sempre forem repetidas o agente não explora novos estados e as estimativas vão convergir para mínimos locais. Se a política sempre fizer a exploração, a política avaliada nunca é a política atual π, mas sim aleatória. Logo é necessário estimular percentuais das duas abordagens é importante para o aprendizado por reforço.

Há várias técnicas para exploração, mas a principal estratégia usada é a ϵ-greedy gerando a seguinte regra:

$$a = \begin{vmatrix} a\ \pi(s,a), comaprobabilidade(1-\epsilon) \\ 0, Qualqueroutraaocomprobabilidade\epsilon \end{vmatrix}$$

.

Com a estratégia estabelecida, neste trabalho o algoritmo escolhido é o sarsamax com as características discutidas abaixo.

3.5.7 Q-Learning e SARSA

No aprendizado por reforço da função Q pela amostragem, podemos usar diferentes ações A': ação que será escolhida na próxima transição, definida pela política π(a',s') ou outra ação que maximixa a estimativa da observação atual da função definida por $a^* = argmax_{a'}Q^{\pi}(s',a')$. Conforme já visto na secção anterior temos dois algoritmos o on-policy e off-policy para o aprendizado por reforço:

1) *On - policy*;

2) *Off - policy*

No algoritmo *On-policy* temos a criação dos caminhos com a política π, em conjunto com as estratégias de exploração. Já a Off-policy as trajetórias são criadas pela política de comportamento alvo para o aprimoramento.

No *On-policy* ou também chamado de SARSA *(state-action-reward-state-action)*. A ação escolhida nesse estado é feita usando a próxima ação a partir da política π (a',s') para atualizar a transição. É necessário que a próxima ação seja escolhida na próxima transição . A Equação da função Q^{π}(s,a) para esse método é Equação 3.21 :

$$Q^{\pi}(s,a) \leftarrow Q^{\pi}(s,a) + \alpha(r(s,a)) + \gamma Q^{pi}(s',a') - Q^{pi}(s,a) \tag{3.21}$$

Já o Off-policy é chamado de Q-Learning. A ação no próximo estado (aquela com mais valor Q) será utilizada para atualizar a transição atual sem considerar a próxima ação e será calculada conforme a Equação 3.22:

$$Q^{\pi}(s,a) \leftarrow Q^{\pi}(s,a) + \alpha(r(s,a)) + \gamma max Q^{pi}(s,a) - Q^{pi}(s,a) \tag{3.22}$$

A política π resultante desse método poderá ser determinada pela máxima estimativa da função Q em cada estado. Para a exploração esse algoritmo deve estar atrelado às estratégias de exploração.

No algoritmo Sarsa max é inserida uma política π e um determinado número de episódios, tendo como resultado uma tabela Q com as ações para cada estado.

O algoritmo inicia coletando as informações da observação ϵ para o espaço em um determinado estado. Logo é escolhido uma ação "A"a partir da análise da recompensa atual e das recompensas futuras somando cada um dos valores, conforme a Equação 3.22 . Repetindo esse processo para o número de episódios determinado inicialmente.

Portanto, para o desenvolvimento de um algoritmo de aprendizado por reforço para a desviar dos obstáculos e alcançar os pontos pré determinados no ambiente foi escolhido o Q-learning sarsamax por se apresentar uma alternativa viável, eficiente e de forma simples para a navegação, desde que seja implementado um algoritmo para designar os pontos de navegação para o robô.

3.5.8 *RAPIDLY EXPLORING RANDOM TREE* (RRT)

Para criar os pontos no qual o robô deve desenvolver sua navegação, temos que o padrão de movimentos é um dos problemas mais estudados na robótica, existindo diferentes metodologias, e muitas técnicas de movimentos que foram introduzidas e empregadas diversas aplicações além da robótica. Como a manipulação de objetos 3D, computação biológica, gráficos computacionais (PRABOWO et al., 2022).

Destaca-se o algoritmo RRT, tornando-se amplamente utilizado para resolver problemas de navegação robótica. Essa técnica utiliza uma amostragem aleatória de um espaço.

Os caminhos são conectados em uma estrutura em árvore, na qual o caminho que liga o ponto inicial ao final pode ser encontrado. O RRT pode ser dividido em três partes principais: seleção de um vértice para expansão, expansão e condição de término. O algoritmo RRT original é descrito na (Figura 3.9) (VONÁSEK et al., 2009).

Uma das desvantagens do desempenho do algoritmo RRT está em ambientes

com passagens estreitas, pois as mesmas prejudicam o processo de crescimento da árvore retardando a busca do caminho do resultado (VONÁSEK et al., 2009).

O RRT foi projetado para pesquisar caminhos com eficiência para muitas dimensões, construindo uma árvore aleatoriamente para encontrar caminhos. A árvore é construída de forma gradual a partir das amostras extraídas de forma aleatória do espaço e possui tendência de crescer em direção às grandes áreas que não foram exploradas empregado para solucionar problemas com obstáculos e restrições para o movimento robótico autônomo.

O básico desse algoritmo para reduzir a exploração aleatória dos locais e o tempo no planejamento dos caminhos. Na seleção das árvores de nós b_{rand}, os pontos alvos são introduzidos e o alcance da seleção aleatória do nó da árvore é controlado dentro do alcance retangular da posição entre o ponto atual e do alvo em caminhos diagonais, a fim de evitar que o alcance do obstáculo entre os dois exceda o alcance regular, tornando impossível executar uma trajetória adequada e uma margem expansivel de r (alcance regular) é definida para o intervalo. Equação 3.23.

$$b_{rand} = rand(b_{alvo} - bprximo + r) \tag{3.23}$$

Para obter a melhor trajetória para se aproximar do destino estão de acordo com a seleção dos novos pontos b_{new} conforme a Equação 3.24:

$$b_{new} = b_{nearest} + \lambda_1 \frac{b_{brand} - b_{nearest}}{|b_{brand} - b_{nearest}|} + \lambda_2 \frac{b_{goal} - b_{nearest}}{|b_{goal} - b_{nearest}|} \tag{3.24}$$

Onde λ_1 é um passo de expansão na árvore na direção em qualquer ponto e λ_2 é um passo de expansão na direção do ponto final $|b_{goal} - b_{nearest}|$ É a distância Euclidiana entre o ponto final e o ponto mais recente na árvore. O ponto de fechamento na árvore será quando $b_{goal} = b_{init}$.

Após o alvo do algoritmo do RRT a seleção do novo ponto estará próxima do alvo, quando λ_1 é maior que a amostragem aleatória esse ponto é possivelmente positivo e quando λ_2 é grande significa que o desvio é maior.

Quando a distância entre b_{new} e b'_{new} é menor do que o seu passo médio $\min\lambda_1, \lambda_2$ a exploração do novo nó termina e o caminho é gravado com a trajetória desejada.

Desde que o robô tenha a velocidade máxima v_{max} constante então corresponde a um passo constante da árvore de expansão. Equação 3.25.

$$max\lambda_1, \lambda_2 < v_{max}t \tag{3.25}$$

No qual t é o tempo do nó corrente para o próximo nó. Valendo ressaltar na criação de caminhos as limitações na trajetória do robô, como a rota do robô que contém um ângulo máximo. ψ_{max} de 30°/m e velocidade angular e de 1m/s de velocidade linear(CAO et al., 2022).

Logo teremos as seguintes limitações para a geração de caminhos. Equação 3.26 e a Equação 3.27.

$$v_t \leq v_{max}\ [\psi_t - \psi_{t-1}] \leq \psi_{max} \tag{3.26}$$

$$[\psi_t - \psi_{t-1}] \leq \psi_{max} \tag{3.27}$$

O RRT cresce usando uma configuração inicial com amostras aleatórias do espaço para busca. O algoritmo começa gerando pontos aleatórios em todas as direções, então analisa qual o último feito em uma determinada direção, sempre os pontos nas extremidades, então gera uma nova cordenda e analisa se é possível um caminho entre o antigo ponto ao novo, se for possível conectar esse ponto vira um vértice servindo de local inicial para as novas expansões.

A medida que cada ponto é alcançado uma nova conexão é tentada entre ela e o estado mais próximo da árvore, escolhendo aquele que fica mais próximo do objetivo final. Se for possível ligar os dois pontos a reta passa por eles sem obstáculos ou restrição e isso resulta em um novo estado da árvore. Com amostragem uniforme do espaço de busca, a probabilidade de expandir um estado existente é proporcional ao tamanho da região e expande preferencialmente áreas grandes e não pesquisadas. Até que a distância do próximo ponto seja menor do que a do objetivo final, como mostrado na Figura 3.9. Se for possível conectar o vértice ao ponto final o algoritmo conecta os pontos e encerra, se não for possível ele repete o processo (CAO et al., 2022).

Para a construção do algoritmo de RRT é estabelecido um ponto inicial, posteriormente é escolhido um ponto aleatório e ligados os pontos e retornando esse

Original RRT

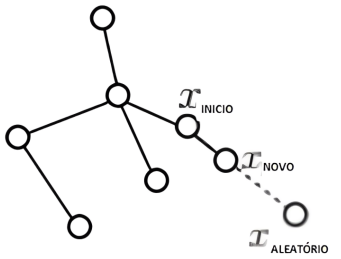
```
1  T.add(q_start)
2  Enquanto interação < K faça
3      q_rand = configuração aleatória
4      q_near = vizinho mais próximo na árvore T para q_rand
5      q_new = estender q_near em direção a q_rand
6      se q_new pode conectar ao q_near então:
7          T .Adicionavertice(q_new )
8          T.Adcionaborda(q_near,q_new):
9      fim
10     se (q_new, q_goal) < distâciadoalvo então:
11         interrompa
12     fim
13 fim
```

FONTE: (CAO et al., 2022)

Figura 3.9 – *Algoritmo RRT*

nó.

As principais vantagens deste algoritmo é a capacidade de implementar, por meio da probabilidade, um caminho em ambientes com muitos obstáculos de forma eficiente Figura 3.10 (VONÁSEK et al., 2009).



FONTE: (VONÁSEK et al., 2009)

Figura 3.10 – *Esquema RRT*

Outra vantagens deste algoritmo é a capacidade fazer a geração, se aproximando do solução ideal, desde que tenhamos o número suficiente de amostras(HUANG et al., 2019).

Contudo, esse método tem dificuldade em explorar ambientes em espaços uniformes, não encontrando os melhores caminhos para se mover do ponto inicial ao final tendo um desvio médio variando entre 5 % a 10% do que se fosse ligado em linha.(HUANG et al., 2019).

Capítulo 4

Metodologia

Neste capítulo serão abordadas todas as etapas para o desenvolvimento da simulação de um robô de navegação autônoma em ambientes externos . Primeiro será abordado sobre a arquitetura já desenvolvida do projeto e depois o desenvolvimento da navegação autônoma em ambiente simulado. Vale ressaltar, que este trabalho visa ser aplicado à implementação da navegação em uma cadeira de rodas real, porém foi empregada na simulação.

4.1 Captura, processamento e classificação de imagens

Para a captura de imagens foi primeiramente desenvolvido um sistema mapeamento simulando imagens de satelite como se as as capturas das fossem obtidas a uma escala de 1cm para 5 m código 1.1. Analisando o funcionamento do processamento de imagens e o RRT, após de obtidos os dados da navegação, isso faz com que reduza possíveis erros de hardware quando empregadas na cadeira real, peemitindo fazer variados testes de navegação, e verificá-los quando submetido em diferentes tipos de mapas.

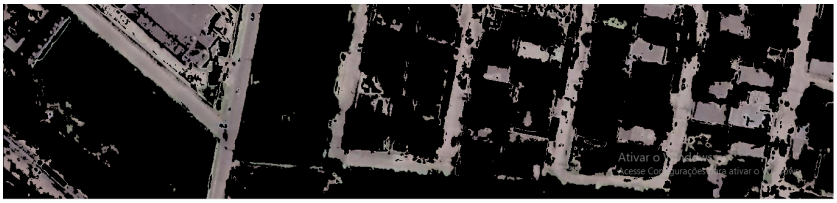
No primeiro momento foram feitas imagens de satélite mostrando o Intituto Federal da Bahia (IFBA), campus Vitória da Conquista para verificar qualidade da imagem e da integridade dos elementos da mesma e se algoritmo teria a capacidade de realizar a tarefa de forma autônoma. Sendo feitas capturas em diferentes pontos do mapa, avaliando quais as melhores estratégias para a captura das ima-

gens Figura 4.1.

FONTE: Próprio Autor

Figura 4.1 – *Captura de imagem de satélite*

Após o funcionamento autônomo da aquisição de imagens, foi feita a etapa de escolha de parâmetros para cada um dos elementos das imagens de pontos aleatórios do mapa, afim de verificação do funcionamento dos filtros de imagem para cada um dos elementos, conforme exemplo abaixo que são escolhidos os parâmetros para as ruas do ambiente no mapa, avaliando se o projeto conseguia realizar as etapas de captura, processamento e classificação de forma qualitativa Figura 4.2.



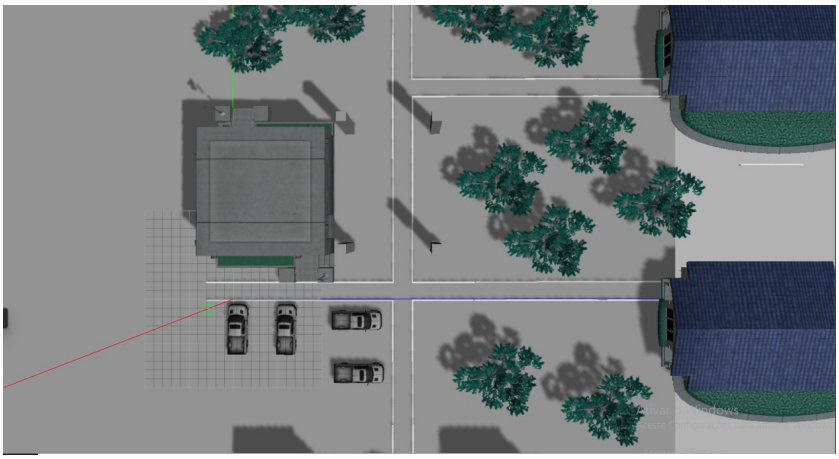
FONTE: Próprio Autor

Figura 4.2 – *Filtro de cor do ambiente*

4.2 Simulação

Após o processo em mapas de satélite reais foram feitas simulações em ambientes simulados para salvar as fotos aéreas do ambiente para a captura da imagem, processamento e classificação dos elementos do cenário baseado na cor usando o software GAZEBO. Gerando um mapa que foi modelado representando uma instituição de ensino com todos os prédios e salas de aula em ambiente externo, conforme a Figura 4.3.

Então, foi usado um processo de mapeamento por processamento de imagens com a definição do filtro usando a biblioteca em python cv2 para selecionar as cores que determinavam áreas nas quais os robôs não poderiam trafegar. Substituindo a técnica do SLAM baseado em laser, foi criado um mapa de grade de



FONTE: Próprio Autor

Figura 4.3 – *Simulação do ambiente*

ocupação 2D (semelhante a uma planta baixa) baseado na imagem gerada, substituindo as técnicas tradicionais que precisavam integrar dados do laser e posição do robô móvel.

Neste novo método apenas com a imagem será criado um mapa para o robô de serviços. Sendo a viabilidade deste ambiente validada de acordo com o tempo de processamento, neste processo em relação ao mapeamento do SLAM e se o mapa consegue atender de forma qualitativa a necessidade da navegação autônoma naquele ambiente.

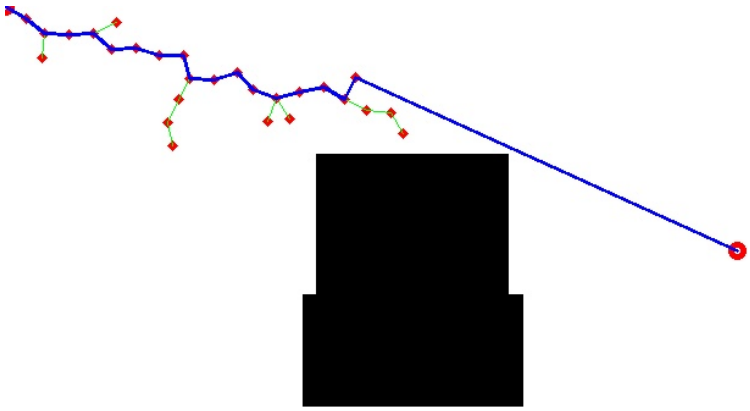
Com a implementação do robô de serviços, com o plugin que simula o sensor LiDAR para analisar os obstáculos e com outro para controlar o robô e determinar as velocidades e as direções,sendo possível deslocar o robô de serviços.

O odometry source é o nó que o robô vai publicar dados de odometria para calcular a distância percorrida, e nesta simulação foi utilizado o plugin da base-drivecontroller que também serve para o base controller. O plugin do LiDAR foi integrado no sensor source com o intuito de utilizar o laser para evitar obstáculos. O componente sensor transforms possui os dados referentes à árvore de transformação do robô, informando os locais onde estão todas as partes atreladas aos nós presentes no arquivo tipo URDF desenvolvido.

4.3 *Rapidly-exploring random trees*

Para realizar a navegação era necessário estabelecer quais pontos o robô irá percorrer, portanto foi escolhido o RRT um algoritmo que permite criar os pontos

em uma trajetória a partir dos pixels do processamento de imagens selecionado os nós e expandindo as ramificações dos pontos até atingir o objetivo final no mapa da simulação. Um exemplo de funcionamento desse algoritmo é mostrado na Figura 4.4:



FONTE: Próprio Autor

Figura 4.4 – *Funcionamento do RRT*

Portanto, nessa simulação o algoritmo consegue gerar pontos até alcançar o destino final, com as marcações na cor vermelha e com as trajetórias entre os pontos em verde e a trajetória ideal em azul. Os pontos que devem ser percorridos para atingir o menor caminho calculado. Os resultados do algoritmo RRT são mensurados em relação a menor distância em linha reta e levando em consideração o menor caminho possível para atingir o destino final em comparação ao registrado na odometria.

Posteriormente, após gerar os pontos temos o desenvolvimento de um algoritmo para realizar a navegação, buscando ao menor distância euclidiana e estabelecendo o ângulo que o robô deve prosseguir para atingir esses valores, desde que não sejam detectados obstáculos, enviando ao nó cmd vel a velocidade linear e angular para percorrer os pontos dimensionados no RRT.

Todavia, se for detectado pelo sensor um objeto em uma distância menor do que 0,5 m a navegação será feita usando o algoritmo de aprendizado por reforço, sendo neste caso empregado o Q-learning treinado previamente para desviar dos obstáculos, através dos dados do sensor Lidar e da odometria.

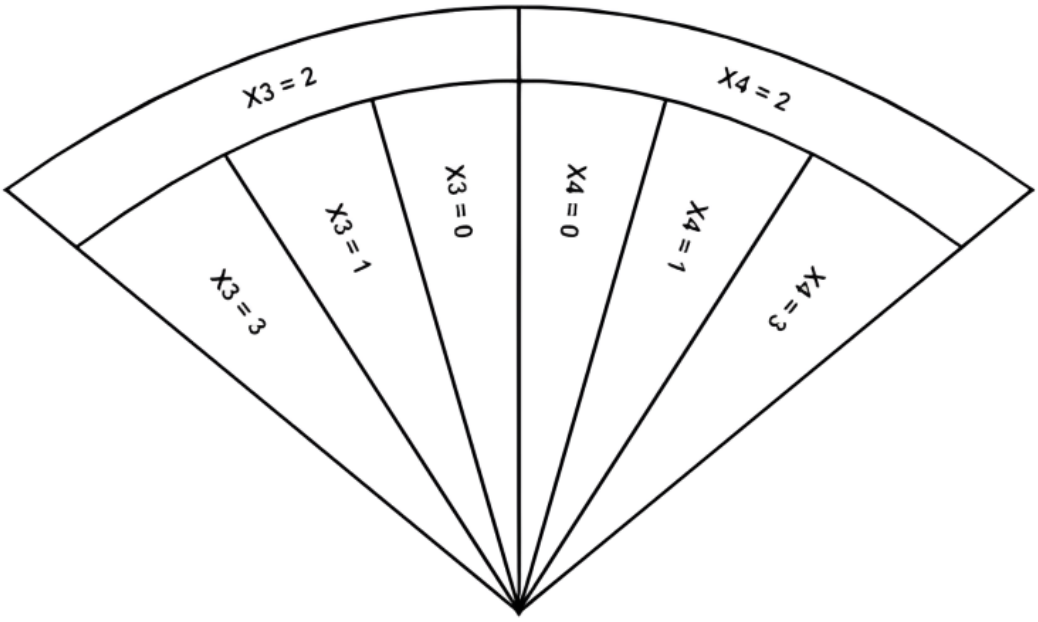
Com a navegação usando aprendizado por reforço, avaliada de acordo com sua capacidade de percorrer todos os pontos determinados sem colisão com nenhum obstáculo e calculada a menor distância entre os pontos e a distância registrada na odometria do robô.

4.4 Aprendizado por reforço

Como o sensor laser usado mede distâncias na faixa de 1 cm a 3,5 m em um círculo de 360° ao redor do robô, no entanto para reduzir o processamente de dados, optou-se por discretizar o algoritmo de inteligência artificial para que ele consiga convergir.

Para sanar este problema, a medição da distância do sensor foi limitada a um máximo de 1 m e no processo da discretização em estados seriam incluídas apenas medições na faixa de [–75°, 75°] em relação à direção e o sentido na qual o robô está se deslocando, representam uma suposição válida para mitigar a quantidade de dados, pois os obstáculos localizados a uma distância maior 1 m de ou atrás do robô não são perigosos e não devem ser considerados.

O espaço de estados consiste em 4 estados variáveis na faixa de medição (x_1, x_2, x_3, x_4), determinados com base na varredura da distância baseada em diferentes regiões e quando algum obstáculo é detectado em uma das três áreas possíveis há uma mudança de estado. As variáveis de estado $x_i = 0$ e x_i = 2 são determinadas com base na distância do obstáculo mais próximo do robô. A distância mencionada é denotada por i, variável os estados x_1 e x_2 são determinados na Figura 4.5:



FONTE: Próprio Autor

Figura 4.5 – *Funcionamento da discretização do espaço*

A posição p (posição do robô) é calculada separadamente para o lado esquerdo e especialmente para o lado direito do robô, onde X3 corresponde ao lado

esquerdo e X4 para a direita. Também deve ser observado que os valores são atribuídos às variáveis de estado em ordem decrescente de prioridade, o que significa que se a condição para x_i = 1 for satisfeita, terá uma prioridade mais alta que a condição para x_i = 0, que nesse caso nem será verificado. Isso faz sentido porque será a prioridade mais alta obstáculos localizados diretamente na frente do robô. Uma ilustração deste particionamento de subespaço de estado das variáveis X3 e X4 é mostrado na Figura 4.5.

Desta forma, o espaço de estados infinito é reduzido a um espaço discretizado e finito, no qual há convergência do algoritmo é possível. Claro que, com um número maior de estados discretos seria o mais adequado para a representação do estado contínuo, mas isso aumentaria a complexidade do algoritmo e tornaria mais difícil convergência e treinamento.

O próximo passo é discretizar o espaço de ações que o agente pode realizar. Para fazer o algoritmo da forma menos complexa possível, optou-se por que o agente tenha 3 ações à sua disposição: avançar, curva à esquerda e curva à direita. Cada uma dessas ações é definida por um dado linear com a velocidade do robô em 0,8 m/s e angular com a velocidade linear e 0,4 $\frac{rad}{s}$.

As velocidades são escolhidas para que o robô gire em um arco ângulo adequado para contornar o obstáculo rápido o suficiente. Isso dá mais flexibilidade ao algoritmo, pois assim poderá aprender e evitar até os obstáculos que estão nas imediações. Vale ressaltar que maior o número de ações também permite um possível melhor desempenho do algoritmo, todavia dificulta significativamente a possibilidade de sua convergência no treinamento para sua aplicação. É por isso que o número de ações definidas dessa maneira é ideal para o problema considerado.

Após definir as ações, finalmente é possível definir a tabela Q abaixo que será dimensão 144 × 3, ou seja, $N_{Espaos} \times N_{Aes}$. Essa tabela Q terá 432 posições sendo de dimensões razoáveis e espera-se que após o processo de treinamento, seus valores convirjam e garantam o desempenho desejado do algoritmo. Sendo dimensionada, conforme abaixo e a próxima etapa no projeto do algoritmo é definir a função de recompensa, que é necessária para determinar o valor Q e preencher a tabela Q dada.

A função de recompensa determina imediatamente o que o agente recebe por realizar uma determinada ação em um determinado estado sob certas circunstâncias desempenhando um papel muito importante no processo de formação,

Tabela - Q	$a_1 = f(v, w)$	$a_2 = f(v, w)$	$a_3 = f(v, w)$
$S1 = f(x_1, x_2x_3x_4)$	$Q(s1, a1)$	$Q(s1, a2)$	$Q(s1, a3)$
$S2 = f(x_1, x_2x_3x_4)$	$Q(s2, a1)$	$Q(s2, a2)$	5415
$S3 = f(x_1, x_2x_3x_4)$	$Q(s3, a1)$	$Q(s3, a2)$	$Q(s3, a3)$
...	...	...	...
$S144 = f(x_1, x_2x_3x_4)$	$Q(s144, a1)$	$Q(s144, a2)$	$Q(s144, a3)$

visando demarcar para o agente quais ações são mais favoráveis, e quais não são.

Essa metodologia ocorrerá por meio da escolha de boas ações nas quais a recompensa será positiva, enquanto para as desfavoráveis a recompensa será negativa. Além disso, as recompensas são essenciais para se adaptar às características do ambiente em que o agente se encontra, para que ele possa através do processo de treinamento aprender a responder às exigências do ambiente e da tarefa que está sendo realizada.

Para atingir todos os itens acima, a função de recompensa é definida como uma combinação de três diferentes funções para calcular a recompensa total do episódio.

Sendo calculada com o somatório da recompensa da ação, da recompensa da distância em relação ao obstáculo e a recompensa pela mudança de direção.

O primeiro caso é quando a ação escolhida é seguir em apenas com a velocidade angular temos uma recompensa de 0,2 e será negativa de 0,1 se a velocidade angular altera a direção do robô.

O segundo termo da recompensa é a análise da distância em relação ao obstáculo mais próximo, na qual temos uma recompensa positiva se os obstáculos estiverem a uma distância maior que 1m e, negativa quando o obstáculo é menor do que 1 metro.

O terceiro elemento que compõe o somatório das recompensas é a alteração da direção em relação à direção anterior, e neste caso temos somado valor nulo se a distância persistir e uma recompensa negativa de 0,8 caso ocorra uma alteração.

O quarto fator da recompensa é caso ele fique a uma distância menor que 0,1 m do obstáculo detectado, no qual ele receberá uma recompensa negativa de -100 e o episódio irá reiniciar. Sendo a recompensa em cada episódio da simulação:

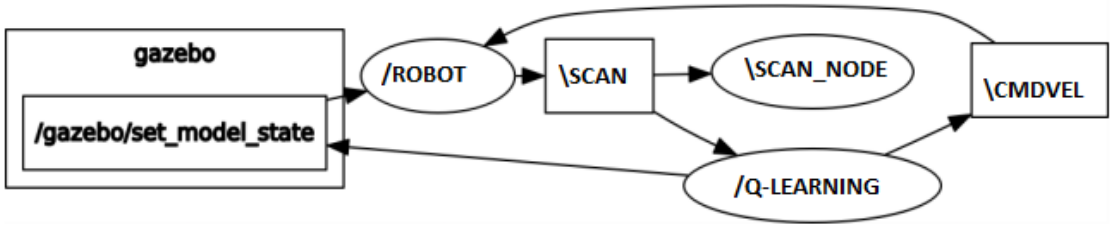
$$reward = r_{ao} + r_{obstculo} + r_{mudana} + r_{choque}$$

Depois de determinar o espaço discreto dos estados e das ações, bem como definir a função de recompensa, o processo de treinamento do agente é abordado, com o objetivo principal do algoritmo evitar obstáculos usando o Q-learning, sendo esperado que no final do processo de treinamento que o agente trafegue continuamente no ambiente em que se encontra sem colisões com obstáculos. Apenas em combinação com um controlador interior previamente concebido o sistema de direção autônoma estará completo, o que significa que o robô móvel chegará à posição desejada enquanto evita todos os obstáculos em seu caminho.

O processo de treinamento do agente foi realizado no ambiente de simulação Gazebo do *Robot Operating System* (*ROS*), que possui os benefícios significam principalmente a abstração de componentes de hardware de robôs reais e a disponibilidade de sinais de sensores e atuadores.

Para o ambiente de treinamento o agente foi usado o mapa turtlebot3_world, do *ROS* que possui a forma reconhecível da tartaruga. Este mapa representa uma escolha frequente dos alunos e engenheiros no projeto de localização paralela e algoritmos de mapeamento, por isso é considerada uma escolha válida também nesta situação e como teste um mapa desenvolvido semelhante a um instituto de educação.

Para ter uma visão melhor do diagrama de fluxo do programa por trás do processo de treinamento, será mostrado seu gráfico rqt, que é uma estrutura do *ROS* que implementa as diversas ferramentas por meio das *Graphical User Interface* (GUI) na forma de plugins. Conforme mostrado na Figura 4.6:



FONTE: Próprio Autor

Figura 4.6 – *Funcionamento do RRT e do Q-learning*

O gráfico rqt tem a estrutura do software projetado, bem como os principais nós tópicos que participam da formação de uma determinada estrutura. Ou seja, o nó principal no qual o completo ocorre o algoritmo de treinamento é um "learning_node"que obtém informações LIDAR dos tópicos "scan"e envia seu controle para as rodas do robô usando o tópico "cmd_vel".

Desta forma, o algoritmo o treinamento se comunica com o simulador Gazebo, na forma de nós e tópicos no lado esquerdo do gráfico rqt, que possibilita uma simulação fiel do treinamento de um robô móvel real na simulação do ambiente ROS, na qual será calculada a capacidade do robô se deslocar pelo ambiente sem choques e seguir os pontos determinados.

Capítulo 5

Desenvolvimento

5.1 Mapeamento da área externa

A nova técnica de navegação desenvolvida neste trabalho para realizar o mapeamento das área externa funciona como com diferentes etapas. Sendo o funcionamento do princípio geral é mostrado no esquema da Figura 5.1.

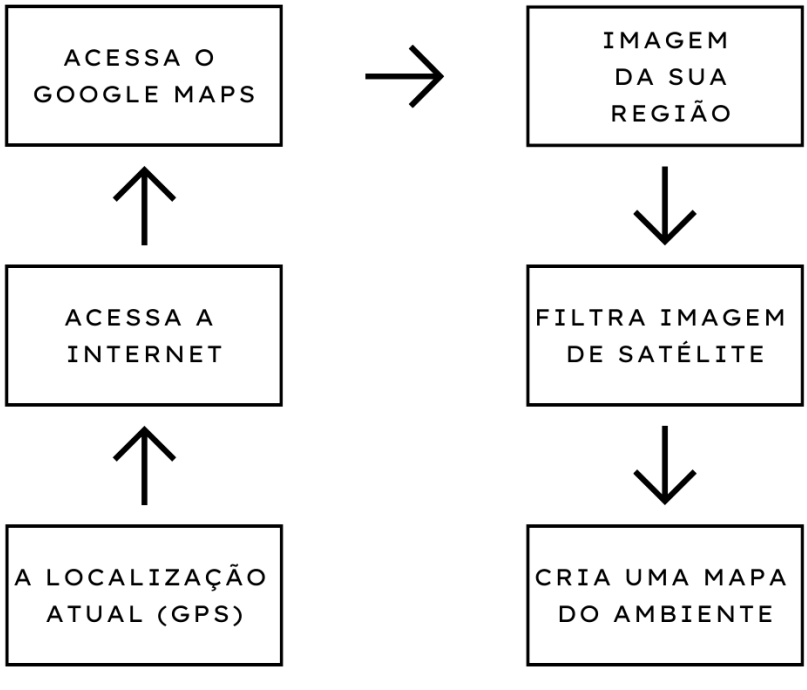
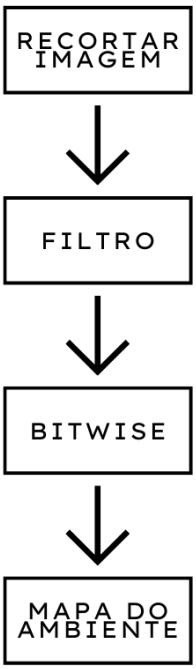


Figura 5.1 – *Esquema do funcionamento geral do mapeamento do ambiente*

O processo tem início quando por meio do GPS, o robô recebe a sua localização atual, então acessa o navegador, buscando a imagem de satélite no banco de

dados do Google Maps, concatenando as informações, capturando a imagem da vista superior do ambiente.

Logo após ocorre o processamento da imagem, para possibilitar o emprego dos filtros com a finalidade de classificar as regiões do ambiente. Com a inserção dos parâmetros da imagem são determinadas as áreas nas quais o robô pode trafegar. Então é criado um mapa do ambiente determinando também os pontos que não podem ser trafegados. Para determinar esses pontos que podem ser navegáveis o processo abaixo é desenvolvido conforme a imagem na Figura 5.2.



FONTE: Próprio Autor

Figura 5.2 – *Processo de criação do mapa do ambiente*

O processo tem início ocorre à aquisição de dados da imagem de satélite do local, usando o com o sync_playwright e o pyautogui capturando a imagem com escala de 1cm para 5 m e cada pixel´s equivalente a 0,7 m , então a imagem é transformada em um padrão np.array e há uma redução das dimensões de 1366 x 768 pixels para 789 x 530 pixel´s para aplicar os filtros apenas na área útil para a criação do mapa. Para obtermos um maior detalhamento do ambiente, conforme mostrado em na Figura 5.3 logo após a imagem é transformada do padrão RGB para o padrão HSV, conforme desenvolvido nas equações da Subseção 3.14 para diferentes cores e dos distintos elementos a partir do espectro de cores por meio do HSV ou matriz, saturação e brilho.

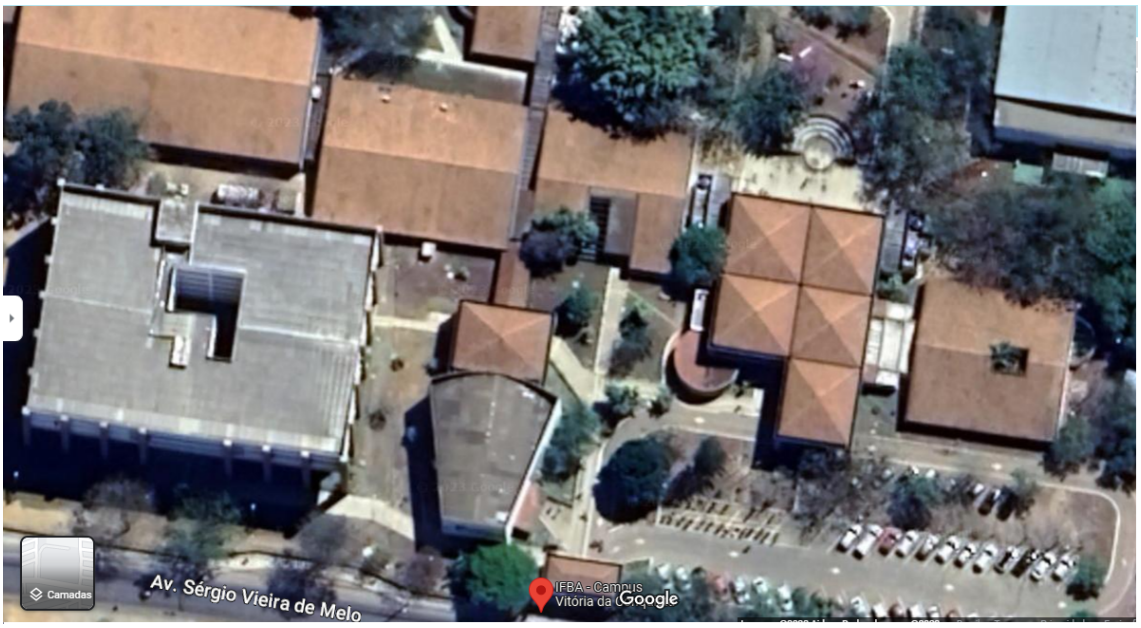
Existem diferentes modos de fazer essa seleção de objetos como o RGB e BGR, mas foi neste caso usado o HSV com base nos intervalos de matiz e saturação,

com variação de valor da matiz da cor, saturação e intensidade.

A imagem coletada é feita no padrão RGB, todavia esse não é o melhor método de seleção, pois as cores não podem ser divididas uniformemente e os componentes quantizados com a mesma qualidade que o padrão escolhido.

Por meio da cvtColor temos a conversão da imagem do tipo RGB para HSV com intervalo H variando no intervalo entre 0 e 180 e uma imagem de 8 bits, sendo escolhidas os limites dos valores que serão estabelecidos, criando uma máscara que é a região navegável.

O histograma para todos os três componentes (matiz, saturação e valor) são calculados e plotados e escolhidos os limites superiores e inferiores para cada componente. O resultado é mostrado na Figura 5.3, no qual é possível verificar o resultado final das operações mencionadas acima e a criação do mapa que posteriormente será usado na navegação no método RRT.



FONTE: Próprio Autor

Figura 5.3 – *Imagem de satélite IFBA*

A imagem da Figura 5.3 acima mostrada na imagem original de satélite extraída do google maps no formato RGB. Posteriormente essa imagem é transformada no padrão numpy e HSV, resultando na imagem abaixo (Figura 5.4).

Especificando os valores para identificar as cores das calçadas, no qual a operação é realizada bitwise or integrando as diferentes áreas que estão dentro do intervalo estabelecido com os bit's classificados em verdadeiro (branco) ou falso (preto) conforme imagem Figura 5.4.

FONTE: Próprio Autor

Figura 5.4 – *Imagem de satélite IFBA transformação HSV*

5.2 Processamento de imagens

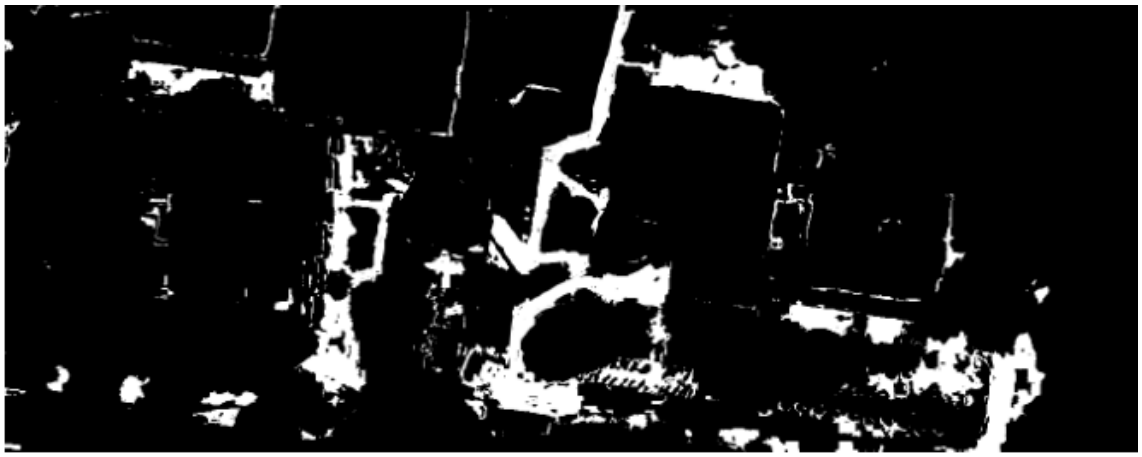
Após essa transformação é iniciado o processo para identificar por meio das cores os locais dos quais é possível fazer a navegação autônoma . No primeiro momento é feita a identificação dos edifícios, conforme a Figura 5.5.

FONTE: Próprio Autor

Figura 5.5 – *Filtro HSV edificios*

Em um segundo momento são identificadas as cores das calçadas nas quais o robô de serviço é capaz de se locomover Figura 5.6.

Posteriormente as imagens de cada um dos filtros de imagem são unidas e formam a imagem resultante na Figura 5.7 com o produto bit a bit e se em uma das imagens o bit for verdadeiro na imagem resultante também será verdadeiro e se em todas as imagens o bit naquele ponto for falso o resultado daquele bit também será. O resultado final é mostrado Figura 5.7.



FONTE: Próprio Autor

Figura 5.6 – *Caminhos filtrados*

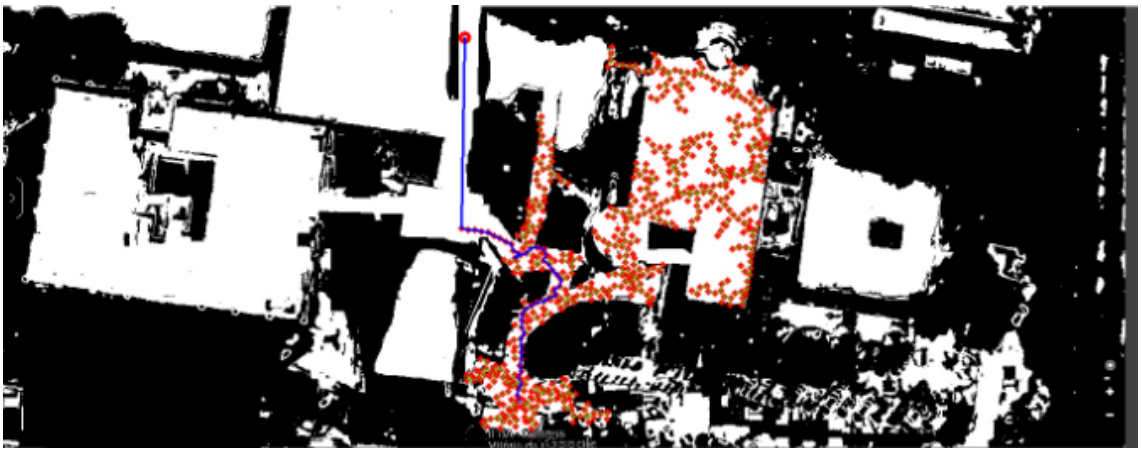
FONTE: Próprio Autor

Figura 5.7 – *Mapa completo*

5.3 Geração de caminhos

Na Figura 5.8 é mostrado o resultado da RRT e cada passo possui uma distância média de 76 cm com a área total do mapa 300 m x 100 m.

A Figura 5.8 mostra a geração de caminho usando a técnica do RRT, no qual foi gerada na simulação saindo do auditório até a coordenação do IFBA campus Vitória da Conquista Figura 5.8.



FONTE: Próprio Autor

Figura 5.8 – *mapa do RRT do IFBA*

5.4 Simulação

5.4.1 Cenário da Simulação

Para realizarmos a simulação do ambiente foi usado o Ubuntu 16.04 com o software gazebo instalado e feita uma simulação buscando parecer um instituto federal de educação, e neste ambiente há obstáculos . O mapa possui batentes para o tráfego dos robôs com cerca de 1 m de distância e um mapa com um total de 100 metros por 300 metros, conforme mostrado na Figura 5.9.

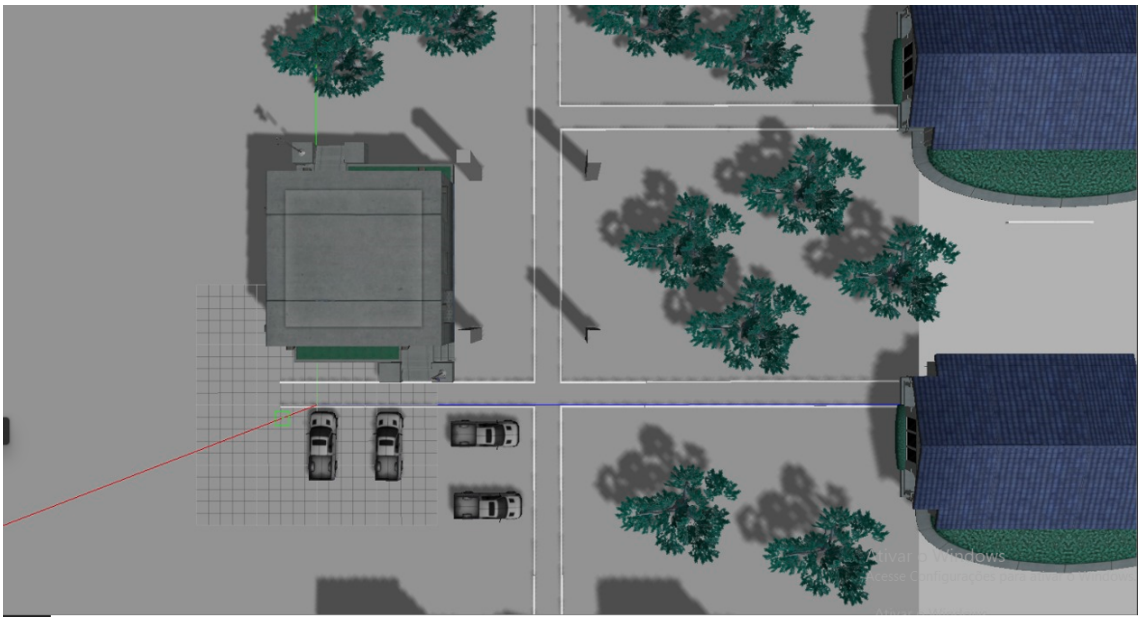
FONTE: Próprio Autor

Figura 5.9 – *Mapa da simulação parte*

Nesta visão superior do mapa é possível constatar todos os ambientes.

5.4.2 Corte da imagem

Nessa primeira etapa do processamento da imagem, será feita uma transformação na imagem tornando-a uma matriz de vetores pela biblioteca *numpy*, para permitir o processamento da imagem para a posterior geração de caminhos, fazendo a transformação no qual cada pixel equivale a um número específico de intensidade e feita a transformação da imagem original no padrão HSV, resultando na Figura 5.10.



FONTE: Próprio Autor

Figura 5.10 – *Mapa da simulação filtro HSV*

5.4.3 Filtro de cor

Para filtrar o mapa de geração de caminho e posteriormente usarmos o algoritmo RRT. Random Trees são usados alguns padrões de filtros dos parâmetros aparentes de cor para filtrarmos quais as regiões que o robô poderá trafegar. Portanto usamos um algoritmo para filtrarmos as intensidades HSV, Código 1.2 nos Anexos classificando os parâmetros para a matiz, intensidade e saturação Como por exemplo para filtrarmos a vegetação foram escolhidos os valores conforme a Tabela 5.1.

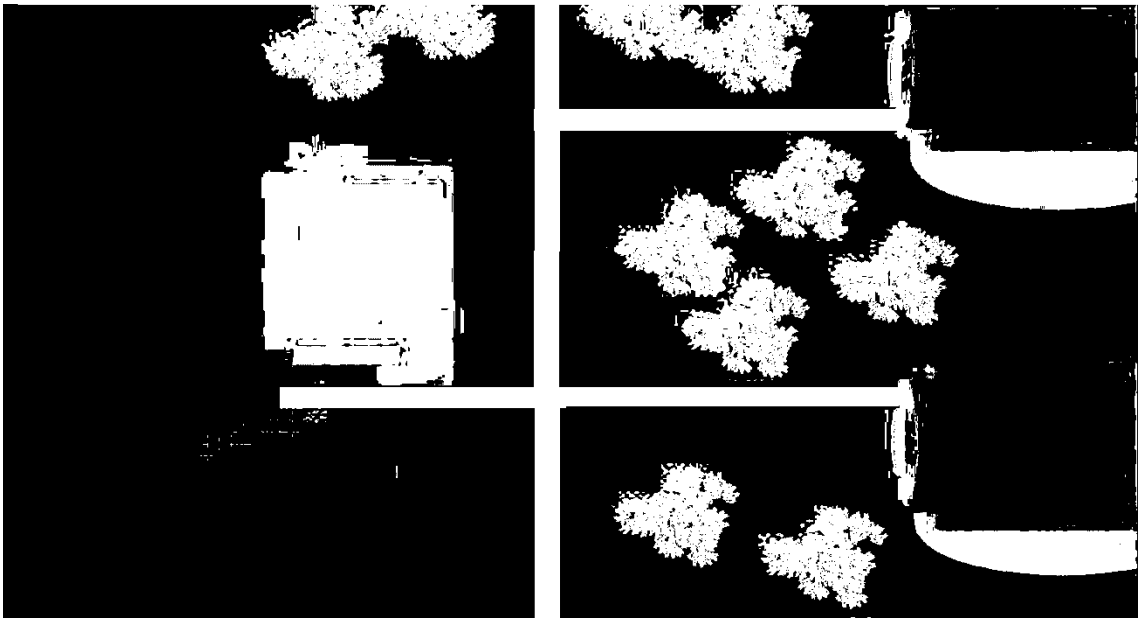
Após todos os filtros para as diferentes de HSV possíveis das áreas que a navegação poderia ser realizada, temos as áreas que podem ser feitos os pontos da navegação, conforme a Figura 5.11 abaixo, sendo possível avaliar as áreas dis-

Tabela 5.1 – *Parâmetros escolhidos do HSV para filtrar a vegetação*

Filtro	valores mínimos	Valores máximos
Matiz	12	65
Saturação	3	150
Valor	26	131

FONTE: Própria

poníveis escolhidas como os edifícios de sala de aula, as áreas de vegetação e os batentes do passeio (Código 1.3) . Mostrando no mapa na cor branca as áreas que os pontos da RRT podem ser inseridos e em preto as áreas que o robô não pode navegar.



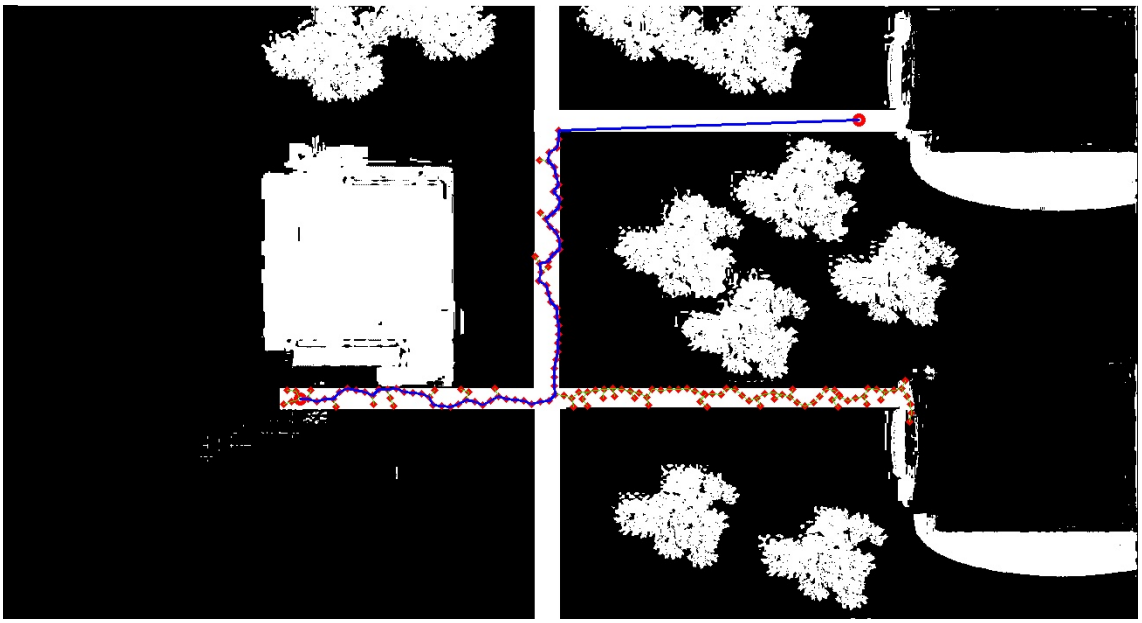
FONTE: Próprio Autor

Figura 5.11 – *Mapeamento da simulação*

5.4.4 Simulação da geração de caminhos

Para realizar a geração dos caminhos da simulação foi escolhido o algoritmo de RRT, quadrados vermelhos para simbolizar os pontos dos caminhos gerados, em azul a ligação do caminho mais próximo entre os pontos gerados pelo algoritmo com ponto. O inicial e final simbolizados com um círculo vermelho e em azul o caminho gerado pelo algoritmo para o melhor caminho Figura 5.12.

Na Figura 5.12 temos o ponto inicial no local mais a esquerda com a possibilidade de diferentes caminhos para atingir o ponto final e é mostrado o caminho



FONTE: Próprio Autor

Figura 5.12 – *Funcionamento RRT na simulação*

escolhido e cada um possui uma distância de 0,5 m e do ponto inicial e final da simulação temos um total de 60 pontos. Os dados comparativos entre a simulação e o melhor caminho são mostrados na tabela abaixo.

Tabela 5.2 – *Parâmetros mapeamento*

Filtro	Medida	Algoritmo
Distância	43	52
tempo	108	130
percentual distância	-	20 %

FONTE: Própria

5.4.5 Treinamento Q-learning

O treinamento foi desenvolvido para que o robô tivesse a capacidade de desviar dos objetos estáticos como paredes e dos cilindros de forma satisfatória, sem choque, com o objetivo de apenas trafegar pelo ambiente sem nenhum dano usando o Q-learning.

Em primeiro lugar, considera-se a seleção dos hiperparâmetros do algoritmo, Para o parâmetro α da pesquisa softmax, o inicial é selecionado o valor α_0 = 0,95, enquanto para o parâmetro em ϵ busca o valor inicial selecionado α = 0,95 ao final de cada época até o mínimo valores de α_{min} = 0,05. O gradiente de decaimento

dos parâmetros foi escolhido para ser aproximadamente o mesmo para comparar algoritmos de forma justa, um valor mínimo de 0,05 em ϵ representa uma probabilidade de 5 % de escolher uma ação aleatória. Valores dos parâmetros α e ϵ se relacionam ao longo dos episódios para consistência eles são dimensionados para no intervalo [0.05,0.95].

Para os parâmetros do próprio algoritmo Q-learning, α e γ foram escolhidos os valores α = 0,75 e γ = 0,98 para garantir um treinamento mais lento dos agentes, além de levar em consideração o efeito de longo prazo de cada ação. O processo de treinamento foi realizado com 500 episódios e o número máximo de ações do agente durante uma época é igual a 500. Resultados de simulação para comparação da quantidade de passos em cada simulação é mostrada no gráfico da Figura 5.13 a seguir:

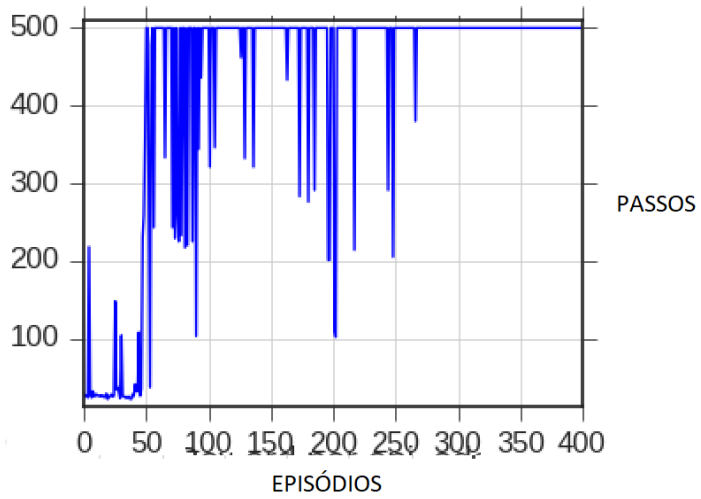


Figura 5.13 – *quantidade de passos x episódios*

A recompensa acumulada é o número médio de etapas do agente calculados em 10 épocas cada, a fim de tornar os gráficos o mais suave possível, o que destaca sua tendência em relação ao valor em cada ponto, atenuando as variações momentâneas de menor importância.

O algoritmo de softmax foi escolhido por convergir para o número máximo de passos por épocas, na qual o agente foi capaz de aprender a evitar obstáculos por causa da distribuição da sua probabilidade de escolha de ações. A mesma conclusão também pode ser tirada olhando para o gráfico da recompensa acumulada ao longo das simulações, a partir da qual fica claro que o agente já desenvolveu um modelo adequado nas primeiras 100 épocas usando a pesquisa softmax ações que lhe trarão as maiores recompensas como pode ser visto na Figura 5.14.

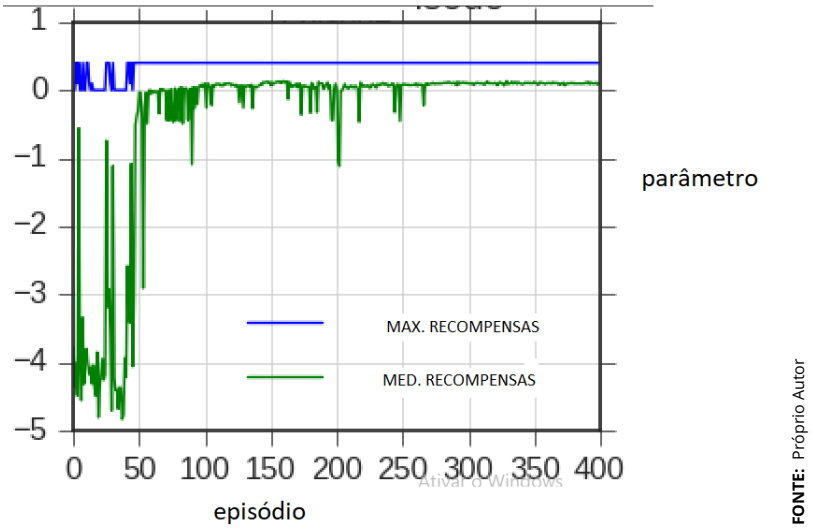


Figura 5.14 – *Recompensa máxima e média por episódio*

O softmax será usado para a estratégia de busca, enquanto o parâmetro γ será fixado no valor γ = 0,9 para levar em consideração o efeito de longo prazo das ações. Número de épocas, como o número máximo de passos por época permanece o mesmo das simulações anteriores.

As escolhas empíricas dos parâmetros no processo de treinamento, o valor na tabela Q é atualizado mais rapidamente, ou seja, aumenta o impacto do conhecimento adquirido em relação ao previamente existente. A conclusão acima apenas confirma a influência do parâmetro para atualizar o valor Q. Embora um valor maior do parâmetro α proporcione uma convergência mais rápida do algoritmo, valores muito grandes podem resultar na instabilidade do processo de treinamento.

Uma vez que o processo de formação final implica um maior número de episódios a um maior número de passos por época, o valor ótimo do parâmetro α será escolhido α = 0,75 para garantir uma convergência mais segura do algoritmo no longo prazo.

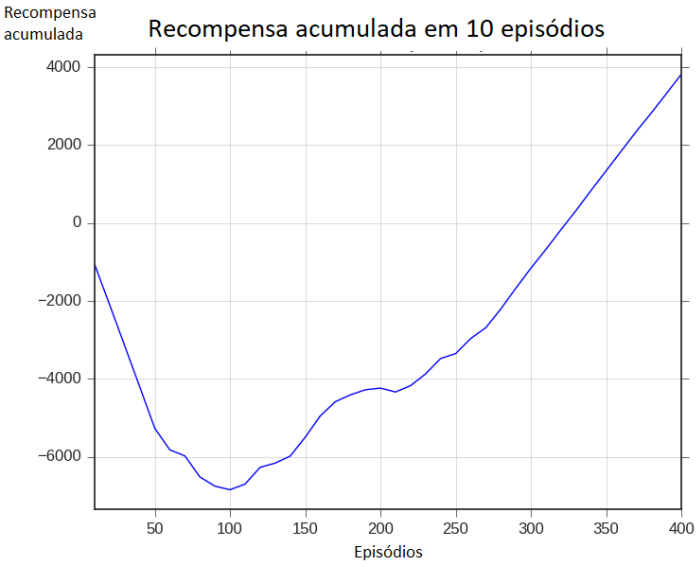
O último parâmetro a discutir é o parâmetro γ. foram realizadas novamente simulações com valores diferentes para γ, onde são ajustados outros parâmetros e hiperparâmetros do algoritmo com base em conclusões anteriores.

A partir das simulações foi possível constatar que apenas valores maiores do parâmetro γ fornecerão convergência do algoritmo, o que significa que o agente deve levar em consideração as consequências de longo prazo de seus atuais , não contando apenas com a recompensa imediata que trazem, mas o pensamento das

melhores estratégias no longo prazo.

A influência do parâmetro γ no caminho com base nos experimentos realizados, concluiu-se que o valor ideal para o parâmetro γ é 0,98, portanto será utilizado durante o treinamento final do agente.

A fim de fornecer uma revisão e monitoramento mais fáceis do processo , a gravação também fornece arquivos de parâmetros de treinamento, bem como o próprio fluxo do processo (Figura 5.15).



FONTE: Próprio Autor

Figura 5.15 – *Cenário projeto*

Deve-se notar também que os parâmetros de treinamento α e γ não são completamente independentes e não podem ser completamente alterados ao observar separadamente. No entanto, as simulações realizadas mostraram que algumas conclusões ainda podem ser tiradas e que os valores ideais podem ser determinados. Quanto aos resultados de simulação apresentados, deve-se apontar o fato de que algumas tiveram que ser executadas mais de uma vez, devido à natureza estocástica dos algoritmos, a convergência não é certa.

Ao final, foi realizado o processo final de treinamento do agente em ambiente de simulação com parâmetros e hiperparâmetros escolhidos com base na análise anterior, sendo que desta vez o treinamento foi estendido para um maior número de épocas e um maior número de passos por época. Visão geral dos hiperparâmetros do processo de treinamento final é dado na Tabela 5.3:

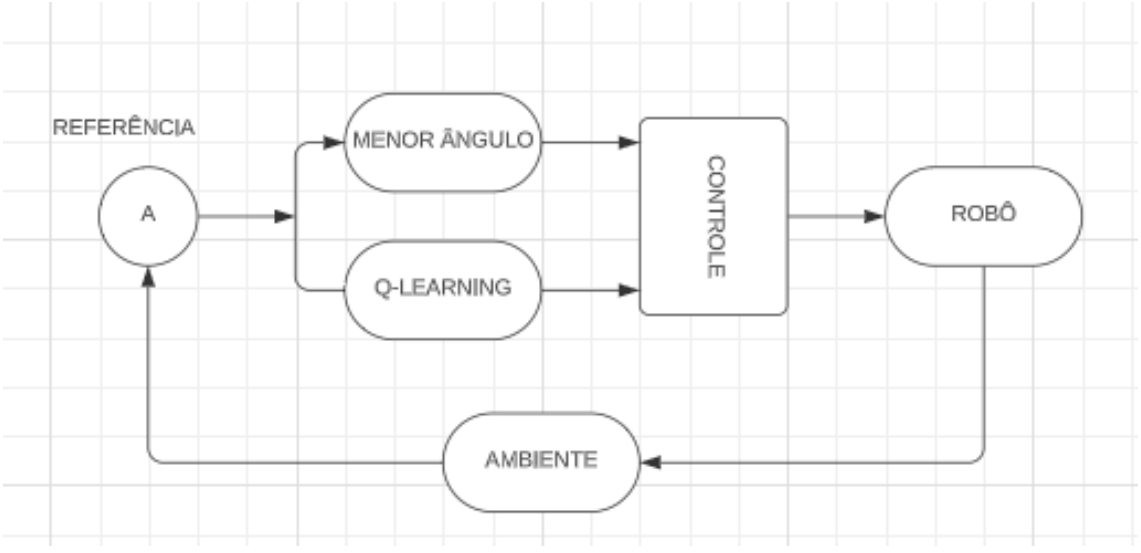
Tabela 5.3 – *Parâmetros Q-learning*

N° Episódios	1000
Número máximo de ações	500
ϵ	0,05
α	0,75
γ	0,98
Algoritmo	SARSAMAX

FONTE: Própria

5.4.6 Navegação

Depois de concluir o design do algoritmo Q-learning, bem como o treinamento do agente na simulação , resta apenas combinar o algoritmo de inteligência artificial com o menor ângulo, projetado, resultando em um algoritmo híbrido de condução autónoma que será implementada num robô móvel. Diagrama de blocos simplificado do algoritmo de controle do robô móvel híbrido mostrado na Figura 5.16:



FONTE: Próprio Autor

Figura 5.16 – *Cenário projeto*

A ideia básica do algoritmo combinado projetado dessa maneira é que o robô móvel se mova em direção ao objetivo sob o controle do algoritmo de controle em um loop de feedback fechado, desde que esteja a caminho não encontre obstáculos. Quando o robô móvel se aproxima de um obstáculo que ameaça seu movimento, o controle é assumido pelo algoritmo Q-learning que lhe permitirá evitá-lo.

Apenas quando o obstáculo está a uma distância maior que 1 m da direção

de leitura de dados do robô, o controle é novamente assumido pelo algoritmo de controle por direção guiando-o até o alvo. Quando a estrutura do algoritmo híbrido é comparada ao básico pelo ciclo de ação de robôs móveis autônomos.

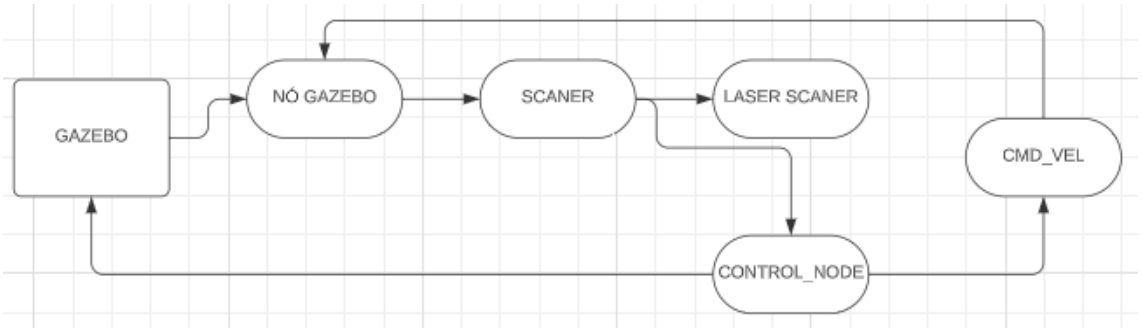
Pode-se concluir que o algoritmo híbrido projetado contém todas as subestruturas necessárias para o funcionando. Com a necessidade de implementar e testar o algoritmo projetado na simulação.

5.4.7 Simulação

Implementação e teste do algoritmo projetado foram realizados na simulação do Gazebo mapa da Figura 5.17, Além de usarmos o ambiente ROS, no mapa específico "turtlebot3_world"para o processo de treinamento.

Foram realizadas diversas simulações nas quais foram definidas as posições inicial e final para o robô alcançar o objetivo desejado. Depois de realizar várias simulações, verificou-se que o algoritmo funciona perfeitamente e que em cada uma das situações dadas garantindo que o robô móvel alcance a posição alvo, conforme as capturas feitas durante as simulações que são mostradas na Figura 5.18.

Para ver o diagrama de fluxo do programa por trás de toda a simulação, será mostrado o gráfico rqt na Figura 5.17:



FONTE: Próprio Autor

Figura 5.17 – *Gráfico rqt navegação final*

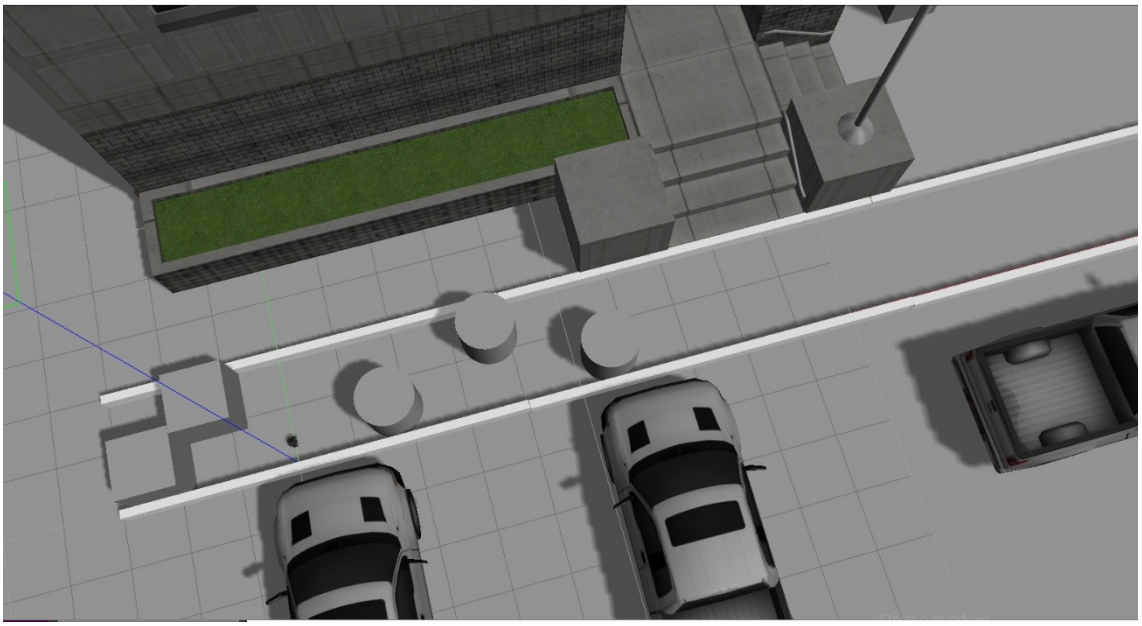
Como os gráficos rqt dos processos anteriores são muito semelhantes a este gráfico . O principal aspecto a ressaltar é a existência de um nó de controle principal que apenas representa um algoritmo híbrido projetado, combinando algoritmos projetados anteriormente Q - aprendizagem e controle em loop de feedback com o controle do robô móvel.

Quando não há obstáculos a uma distância menor que 1 m temos o cálculo de direção do robô feito da seguinte maneira (Equação 5.1).

$$angulo_{desejado} = math.atan2(posicao_{desejada}.y - y, posicao_{desejada}.x - x) \quad (5.1)$$

Se o ângulo for maior que 0,4 rad teremos escolhida a ação 2 no sentido horário e se $-0,4$ a ação 1 para anti-horário.

A simulação ocorreu no mapa mostrado na Figura 5.18 mostrando os pontos iniciais e finais, sendo acrescentados obstáculos para que o robô pudesse desviar dos mesmos e a imagem do algoritmo como pode ser visto na Figura 5.12 que demonstra o cenário do projeto. com os dados coletados na Tabela 5.4.



FONTE: Próprio Autor

Figura 5.18 – *Simulação Final*

Tabela 5.4 – *Parâmetros Navegação*

Tempo calculado	120 s
Tempo real	170 s
Distância calculada	100 m
Distância real	120 m

FONTE: Própria

Nas simulações testadas os dados foram apresentados na tabela acima e não ocorreram choques entre o robô e os obstáculos que foram inseridos no cenário

do ambiente e usando a navegação baseada no Q-learning quando o robô tinha obstáculos próximos e usando apenas o ângulo na menor distância quando não era detectado objetos pelos sensores do robô.

5.4.8 Ameaças à validação do trabalho

Este projeto por usar diferentes tecnologias para ser empregado para sanar a dificuldade em não poder mapear todos os ambientes externos, fez com que tivessemos alguns problemas atrelados a validação da técnica desenvolvida, pois não foram encontrados trabalhos que utilizassem os mesmos algoritmos. Então há dificuldades para fazer estudos e encontrar o estado atual do trabalho, validar a técnica ou analisar sua eficiência comparando com os resultados desenvolvido por outros autores ou metodologias diferentes.

Por não encontrarmos estudos desenvolvidos usando os mesmos algoritmos, há uma dificudade em fazer fazer estudos comparativos de resultados e comparar se o trabalho está de acordo com a com outras situações que não foram exploradas e se será no futuro capaz de ser implementado em um robô real de acordo com os objetivos propostos.

Portanto, alguns resultados tiveram que ser analisados de modo qualitativo, não podendo mensurar sua eficiência em compraração a outras técnicas ou trabalhos existentes, e em alguns casos retringindo-se a avaliar se os objetivos foram alcançados conforme porposto.

Futuramente este trabalho será empregados na cadeira de rodas autônoma para a navegação em ambiente externos, facilitando sua validação, haja vista, podermos avaliar se consegue atender as demandas requeridas.

Capítulo 6

Considerações Finais

Neste trabalho foi desenvolvido um software que por meio da localização de GPS com posição atual do robô e usando a biblioteca playwright.syncapi, foi usado o banco de dados do Google Maps para fotografar as imagens da vista superior.

A fotografia da imagem aérea passa por um processamento de dados no qual temos a transformação da imagem do padrão RGB para o HSV e numpy , com o intuito de aperfeiçoar a imagem coletada. Seguidamente foi feito um filtro usando como critério as diferentes cores dos objetos para conseguir determinar quais os locais possuem vegetação, área de tráfego de pedestre e os edifícios.

Logo, após a escolha dos filtros para cada elemento como os tons de verde para a vegetação, para os telhados dos edifícios, determinando os limites estabelecidos. Então são feitas operações do tipo **or** para cada um dos Bits e modelada a imagem final com o resultado estabelecido e em branco são as áreas com bit´s do tipo verdadeiro onde pode ser feita a navegação e bits falsos são áreas não trafegáveis, com essa técnica substituindo o antigo processamento por SLAM cumprindo com os objetivos propostos.

Para o mapeamento temos o emprego do algoritmo RRT para criar os pontos que o robô percorreria para navegar, desviando de todos os obstáculos estáticos visíveis na vista superior e gerando o melhor caminho para alcançar o ponto alvo.

Empregamos um algoritmo de Q-Learning para trafegar pelo mapa criado anteriormente, conseguindo fazer todo o percurso solicitado, desviando dos objetos que estavam na área e que não fazem parte do mapa estático como pessoas, objetos, animais e outros que não são mapeados na imagem por satélite. Os pro-

cessamento e as tomadas de decisão dados foram unidas em uma arquitetura integrando os pontos gerados pelo algoritmo de RRT que são informados ao Q-leaning responsável pela navegação.

Para testar todas as aplicações foi feita a simulação semelhante a uma instituição de ensino, usando o software gazebo para gerar todos os ambientes disponíveis para a navegação com o mapa gerado a aplicação foi testada de forma simulada os filtros de imagem, o processamento, mapeamento, emprego do RRT para gerar o caminho e o uso do aprendizado por reforço para percorrer o mapa. Sendo que essa técnica é válida conseguindo alcançar os objetivos estabelecidos.

Todavia, o emprego dessa técnica de navegação ainda precisam de algumas melhorias como o aperfeiçoamento do processamento e detecção por imagem para conseguir mapear o ambiente e determinar os obstáculos de forma mais satisfatória e adicionar ao entendimento do algoritmos as regiões proxêmicas para aprimorar a interação com as pessoas.

6.1 SUGESTÕES DE TRABALHOS FUTUROS

Conforme mencionado nas considerações finais, o processamento de imagens foi uma difícil etapa de realização deste trabalho, haja vista a dificuldade na implementação e a indisponibilidade de tempo. A técnica empregada apesar de apresentar resultados satisfatórios em uma ambiente simulado, em locais mais complexos a quantidade de cores e os filtros se tornaram muito maiores, necessitando de uma enorme quantidade de filtros para a criação de um mapa, ocasionando um tempo longo tempo de processamento prejudicando a navegação e a aplicação do algoritmo em ambientes reais. Para aperfeiçoar a imagem podem ser usados outros processos para a melhoria de imagem, pois ainda há a presença de distoções no mapa e usar técnicas de inteligência artificial para classificar os elementos pertinentes à navegação.

O segundo ponto de melhoria é a implementação câmeras acopladas a cadeira de rodas possibilitando a extração das características visuais, para que as mesmas possam detectar obstáculos não detectados pelo sensor LIDAR, consonância com algoritmos de tomadas de decisão autônomo como o convolutional neural network e a partir de uma série de ações pré-estabelecidas desvia desses objetos de forma mais apropriada em ambientes tridimensionais.

A terceira melhoria seria o desenvolvimento da análise das melhores técnicas de geração de caminhos para a navegação com algoritmos de inteligência artificial ou de programação tradicional, determinar os caminhos identificando aquelas que conseguem alcançar o objetivo da melhor forma, analisando as limitações de cada um deles indicando quais os melhores para serem aplicadas no robô em cada ambiente.

Também sugere-se posteriormente devido às suas aplicações em ambientes dinâmicos com a presença de pessoas acrescentar no algoritmo as regiões proxêmicas para melhorar a interação com os seres humanos facilitando sua integração.

REFERÊNCIAS

ADIYATOV, O.; VAROL, H. A. Rapidly-exploring random tree based memory efficient motion planning. In: IEEE. *2013 IEEE international conference on mechatronics and automation*. [S.l.], 2013. p. 354–359. (Citado na página 2.)

AHMAD, M. A. et al. Features of the construction and control of the navigation system of a mobile robot. WARSE, 2020. (Citado na página 5.)

AKHUND, T. M. N. U. et al. Iot waiter bot: A low cost iot based multi functioned robot for restaurants. In: IEEE. *Internat. Conf. on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)*. [S.l.], 2020. p. 1174–1178. (Citado na página 5.)

ALEXIS, P. R-tourism: Introducing the potential impact of robotics and service automation in tourism. *Ovidius University Annals, Series Economic Sciences*, v. 17, n. 1, 2017. (Citado na página 6.)

AMZIANE, A. et al. Msfa-net: A convolutional neural network based on multispectral filter arrays for texture feature extraction. *Pattern Recognition Letters*, Elsevier, v. 168, p. 93–99, 2023. (Citado na página 7.)

BATALIN, M. A.; SUKHATME, G. S.; HATTIG, M. Mobile robot navigation using a sensor network. In: IEEE. *IEEE Internat. Conf. on Robotics and Automation, 2004. Proceedings*. [S.l.], 2004. v. 1, p. 636–641. (Citado na página 6.)

BHADORIA, P.; AGRAWAL, S.; PANDEY, R. Image segmentation techniques for remote sensing satellite images. In: IOP PUBLISHING. *IOP Conference Series: Materials Science and Engineering*. [S.l.], 2020. v. 993, n. 1, p. 012050. (Citado na página 15.)

BISHOP, R. C. D. R. H. *Modern control systems*. [S.l.: s.n.], 2011. (Citado na página 18.)

CAI, G.-S.; LIN, H.-Y.; KAO, S.-F. Mobile robot localization using gps, imu and visual odometry. In: IEEE. *2019 International Automatic Control Conference (CACS)*. [S.l.], 2019. p. 1–6. (Citado na página 6.)

CAO, L. et al. 3d trajectory planning based on the rapidly-exploring random tree–connect and artificial potential fields method for unmanned aerial vehicles. *International Journal of Advanced Robotic Systems*, SAGE Publications Sage UK:

London, England, v. 19, n. 5, p. 17298806221118867, 2022. (Citado 2 vezes nas páginas 29 and 30.)

CHAPMAN, D.; KAELBLING, L. P. Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In: *Ijcai*. [S.l.: s.n.], 1991. v. 91, p. 726–731. (Citado na página 16.)

CLAVERO, J. G. et al. Defining adaptive proxemic zones for activity-aware navigation. *arXiv e-prints*, p. arXiv–2009, 2020. (Citado na página 7.)

DHINGRA, S.; KUMAR, D. A review of remotely sensed satellite image classification. *International Journal of Electrical and Computer Engineering*, IAES Institute of Advanced Engineering and Science, v. 9, n. 3, p. 1720, 2019. (Citado na página 10.)

ELALLID, B. B. et al. A comprehensive survey on the application of deep and reinforcement learning approaches in autonomous driving. *Journal of King Saud University-Computer and Information Sciences*, Elsevier, 2022. (Citado 2 vezes nas páginas 5 and 7.)

FAHLE, S.; PRINZ, C.; KUHLENKÖTTER, B. Systematic review on machine learning (ml) methods for manufacturing processes–identifying artificial intelligence (ai) methods for field application. *Procedia CIRP*, Elsevier, v. 93, p. 413–418, 2020. (Citado na página 5.)

FERRER, G. et al. Robot social-aware navigation framework to accompany people walking side-by-side. *Autonomous robots*, Springer, v. 41, n. 4, p. 775–793, 2017. (Citado na página 1.)

GANESAN, P. et al. Hsv color space based segmentation of region of interest in satellite images. In: IEEE. *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. [S.l.], 2014. p. 101–105. (Citado na página 12.)

GUAN, L. et al. Real-time vehicle detection framework based on the fusion of lidar and camera. *Electronics*, MDPI, v. 9, n. 3, p. 451, 2020. (Citado na página 2.)

HOFMANN-WELLENHOF, B. Elementary mathematical models for gnss positioning. *Mathematische Geodäsie/Mathematical Geodesy: Handbuch der Geodäsie, herausgegeben von Willi Freeden und Reiner Rummel*, Springer, p. 1315–1448, 2020. (Citado 4 vezes nas páginas 8, 9, 10, and 11.)

HUANG, Y. et al. A motion planning and tracking framework for autonomous vehicles based on artificial potential field elaborated resistance network approach. *IEEE Transactions on Industrial Electronics*, IEEE, v. 67, n. 2, p. 1376–1386, 2019. (Citado 2 vezes nas páginas 6 and 30.)

IMTEAJ, A. et al. Robofi: autonomous path follower robot for human body detection and geolocalization for search and rescue missions using computer vision and iot. In: IEEE. *Internat. Conf. on Advances in Science, Engineering and Robotics Technology*. [S.l.], 2019. p. 1–6. (Citado na página 7.)

JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. Machine learning and deep learning. *Electronic Markets*, Springer, v. 31, n. 3, p. 685–695, 2021. (Citado na página 16.)

JING, W. et al. A computational framework for automatic online path generation of robotic inspection tasks via coverage planning and reinforcement learning. *IEEE Access*, v. 6, p. 54854–54864, 2018. (Citado na página 6.)

KOLKUR, S. et al. Human skin detection using rgb, hsv and ycbcr color models. *arXiv preprint arXiv:1708.02694*, 2017. (Citado na página 13.)

KUO, C.-M.; CHEN, L.-C.; TSENG, C.-Y. Investigating an innovative service with hospitality robots. *International Journal of Contemporary Hospitality Management*, Emerald Publishing Limited, 2017. (Citado na página 5.)

MAHESH, B. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, v. 9, p. 381–386, 2020. (Citado na página 16.)

MATTEI, G. et al. An advanced itar-free ins/gps designed and developed in italy. In: IEEE. *2018 DGON Inertial Sensors and Systems (ISS)*. [S.l.], 2018. p. 1–17. (Citado na página 8.)

MOHANDES, M.; DERICHE, M. Image based arabic sign language recognition. In: *8th International Symposium on Signal Processing and its Applications, ISSPA 2005*. [S.l.: s.n.], 2005. p. 86–89. (Citado na página 15.)

MONTEIRO, N. S. et al. Localização e planejamento de movimento de robôs móveis em ambientes internos utilizando processos de decisão de markov. Universidade Federal de Minas Gerais, 2020. (Citado na página 19.)

NANEVA, S. et al. A systematic review of attitudes, anxiety, acceptance, and trust towards social robots. *International Journal of Social Robotics*, Springer, v. 12, n. 6, p. 1179–1201, 2020. (Citado 3 vezes nas páginas 1, 14, and 15.)

NGUYEN, T. L.; DO, T. T. H. Artificial intelligence in healthcare: A new technology benefit for both patients and doctors. In: IEEE. *2019 Portland International Conference on Management of Engineering and Technology (PICMET)*. [S.l.], 2019. p. 1–15. (Citado na página 5.)

POSSOBOM, C.; MEDINA, R. *Dissertação CAMILA*. 2016. (Citado na página 18.)

PRABOWO, Y. A. et al. Utilizing a rapidly exploring random tree for hazardous gas exploration in a large unknown area. *IEEE Access*, IEEE, v. 10, p. 15336–15347, 2022. (Citado na página 27.)

PRABU, S.; BALAMURUGAN, V.; VENGATESAN, K. Design of cognitive image filters for suppression of noise level in medical images. *Measurement*, Elsevier, v. 141, p. 296–301, 2019. (Citado na página 7.)

QING-XIAO, Y. et al. Research of the localization of restaurant service robot. *International Journal of Advanced Robotic Systems*, SAGE Publications Sage UK: London, England, v. 7, n. 3, p. 18, 2010. (Citado na página 5.)

RAHIMAN, W.; ZAINAL, Z. An overview of development gps navigation for autonomous car. In: IEEE. *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*. [S.l.], 2013. p. 1112–1118. (Citado na página 2.)

RUSSELL, S.; NORVIG, P. Artificial intelligence: a modern approach, global edition 4th. *Foundations*, v. 19, p. 23, 2021. (Citado na página 16.)

SANTOS, L. C. et al. Path planning for ground robots in agriculture: A short review. In: IEEE. *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. [S.l.], 2020. p. 61–66. (Citado na página 6.)

SEPEHRI, A.; MOGHADDAM, A. M. A motion planning algorithm for redundant manipulators using rapidly exploring randomized trees and artificial potential fields. *IEEE Access*, IEEE, v. 9, p. 26059–26070, 2021. (Citado na página 6.)

SRINIVAS, T. et al. A comprehensive survey of techniques, applications, and challenges in deep learning: A revolution in machine learning. *International Journal of Mechanical Engineering*, v. 7, n. 5, p. 286–296, 2022. (Citado na página 6.)

SUN, H. et al. Motion planning for mobile robots—focusing on deep reinforcement learning: A systematic review. *IEEE Access*, IEEE, v. 9, p. 69061–69081, 2021. (Citado na página 6.)

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press, 2018. (Citado 6 vezes nas páginas 17, 18, 19, 21, 22, and 23.)

TAKAHASHI, M. et al. Developing a mobile robot for transport applications in the hospital domain. *Robotics and Autonomous Systems*, Elsevier, v. 58, n. 7, p. 889–899, 2010. (Citado na página 5.)

THAMMACHANTUEK, I.; KETCHAM, M. Path planning for autonomous mobile robot navigation with evolutionary particle swarm optimization. In: *2019 Research, Invention, and Innovation Congress (RI2C)*. [S.l.: s.n.], 2019. p. 1–5. (Citado na página 6.)

VINCENT, L.; DOUGHERTY, E. R. Morphological segmentation for textures and particles. In: *Digital image processing methods*. [S.l.]: CRC Press, 2020. p. 43–102. (Citado na página 7.)

VONÁSEK, V. et al. Rrt-path–a guided rapidly exploring random tree. In: SPRINGER. *Robot motion and control 2009*. [S.l.], 2009. p. 307–316. (Citado 3 vezes nas páginas 27, 28, and 30.)

XIA, L. et al. A survey of image semantics-based visual simultaneous localization and mapping: Application-oriented solutions to autonomous navigation of mobile robots. *International Journal of Advanced Robotic Systems*, SAGE Publications Sage UK: London, England, v. 17, n. 3, p. 1729881420919185, 2020. (Citado na página 2.)

YANG, G.-S.; CHEN, E.-K.; AN, C.-W. Mobile robot navigation using neural q-learning. In: *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*. [S.l.: s.n.], 2004. v. 1, p. 48–52 vol.1. (Citado na página 6.)

ZHOU, T. et al. Mmw radar-based technologies in autonomous driving: A review. *Sensors*, MDPI, v. 20, n. 24, p. 7283, 2020. (Citado na página 6.)

ZHOU, Z.-H. *Machine learning*. [S.l.]: Springer Nature, 2021. (Citado 2 vezes nas páginas 16 and 17.)

ANEXO I

ANEXOS

Código I.1 – *Programação imagens WEB*

```
import time
import os
from crop import*
from color3 import*
from color5 import*
from screenshot import*
import pyautogui
from selenium import webdriver

navegador = webdriver.Chrome()

latitude  = -14.8415204
longitude = -40.8771423

/url = "https://www.google.com.br/maps/@"
myTuple1 = (url,"/@",str(latitude),",",str(longitude)
,",90m/data=!3m1!1e3")

tes = f"{url}{latitude},{longitude},47m/data=!3m1!1e3"
navegador.get(tes)
time.sleep(5)
print(tes)
time.sleep(5)
```

```
pyautogui.hotkey('win', 'up')
time.sleep(3)

pyautogui.click(180, 195)
time.sleep(3)
print("tirar print")
screenshot ()
crop()
color()
join()
time.sleep(10)

/navegador.find$_$element("xpath",'//*
/[@id="QA0Szd"]/div/div/div[2]/button').click()
/time.sleep(3)
\ldots
```

Código I.2 – *Parâmetros do HSV*

```


# The vegetation color
lower_vegetation = np.array([12, 3, 26])
upper_vegetation = np.array([65, 252, 131])

# The sand color
lower_sand = np.array([4, 32, 134])
upper_sand = np.array([22, 62, 227])

# The street color

lower_street = np.array([0, 0, 242])
upper_street = np.array([0, 0, 255])
                    .
                    .
                    .
                    .
```

I.0.1

Código I.3 – *Junção Imagem*

```
1
2
3
4
5
6  import numpy as np
7  import cv2
8
9
10 def join():
11
12
13
14     cap1 = cv2.imread('1.png')
15     cap2 = cv2.imread('2.png')
16     mask = cv2.bitwise_or(cap1,cap2)
17
18     status = cv2.imwrite('3.png',mask)
19     cv2.destroyAllWindows()
20     #if cv2.waitKey(1) == ord('k'):
21         #break
22
23 cap.release()
```