

MARIA LETÍCIA NEGREIROS VENTURA ROSA

**NAVEGAÇÃO AUTÔNOMA DE ROBÔS MÓVEIS
PARA AGRICULTURA DE PEQUENA ESCALA**

São Paulo
2023

MARIA LETÍCIA NEGREIROS VENTURA ROSA

NAVEGAÇÃO AUTÔNOMA DE ROBÔS MÓVEIS
PARA AGRICULTURA DE PEQUENA ESCALA

Versão Corrigida

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Ciências.

São Paulo
2023

MARIA LETÍCIA NEGREIROS VENTURA ROSA

NAVEGAÇÃO AUTÔNOMA DE ROBÔS MÓVEIS
PARA AGRICULTURA DE PEQUENA ESCALA

Versão Corrigida

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Ciências.

Área de Concentração:

Engenharia de Sistemas

Orientador:

Prof. Dr. Diego Colón


São Paulo
2023

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

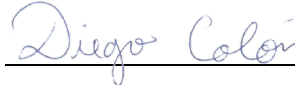
Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 19 de dezembro de 2023

Assinatura do autor:



Assinatura do orientador:



Catálogo-na-publicação

Rosa, Maria Letícia Negreiros Ventura
Navegação Autônoma de Robôs Móveis para Agricultura de Pequena Escala /
M. L. N. V. Rosa -- versão corr. -- São Paulo, 2023.
90 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Telecomunicações e Controle.

1.Robôs Móveis Inteligentes (Aplicações; Uso) 2.Simulação de Sistemas
3.Agricultura I.Universidade de São Paulo. Escola Politécnica. Departamento
de Engenharia de Telecomunicações e Controle II.t.

AGRADECIMENTOS

Agradeço a Deus pela oportunidade de novos conhecimentos através deste trabalho. Agradeço aos meus pais, Maria Christina e José Airton por sempre me apoiarem, pela confiança e incentivo dados. Agradeço ao meu querido esposo Rafael pela paciência e apoio durante os tempos de estudo e ao Prof. Dr. Alysson pelo incentivo e apoio. Agradeço à Universidade de São Paulo pelo ensino de qualidade e suporte sempre oferecidos e a todo corpo docente, membros e assistentes das secretarias e bibliotecas, principalmente Maria Cristina Bonesio, que me ajudou muito na etapa final deste trabalho. Um agradecimento especial ao Prof. Dr. Diego Cólón por sua orientação, ensinamentos, confiança e apoio durante todo o tempo para a realização deste trabalho.

Lembre-se que as pessoas podem tirar tudo de você, menos o seu conhecimento.

-Albert Einstein-

RESUMO

Este trabalho estuda a navegação autônoma de robôs para tarefas de posicionamento em uma sequência de pontos de trajetória. A aplicação imaginada é para agricultura de média e pequena escala pois nestas situações as coordenadas de GPS não são precisas o suficiente. Neste trabalho portanto trata-se do rastreamento de trajetória integrado com fusão de sensores inerciais e o algoritmo de controle. Inclui considerações de energia no controlador e permite, utilizando o algoritmo do morcego, integração de vários robôs de forma colaborativa sem aumentar a complexidade do algoritmo. A geração de trajetória é automática em conjunto com o controle. Os resultados são obtidos em simulação em um ambiente de software específico para robótica (Webots) e testado em dois robôs simultaneamente. As respostas indicam que o resultado é robusto para variações de posição e consideração de energia além da facilidade de uso de mais robôs.

Palavras-Chave – Navegação autônoma de robôs móveis, Sistemas de controle para robótica, Controle de trajetória, Autoencoder, Fusão de sensores.

ABSTRACT

This work studies the autonomous navigation of robots for positioning tasks and following a sequence of trajectory points. The envisioned application is for medium and small scale agriculture, as in these situations the GPS coordinates are not accurate enough. This work therefore involves trajectory tracking integrated with fusion of inertial sensors. The control algorithm includes energy considerations and allows, using the bat algorithm, integration of several robots collaboratively working without increasing the complexity of the algorithm. Trajectory generation is combined with control. The results are simulated in a specific software environment for robotics (Webots) and tested on two robots simultaneously. The results indicate that the methodology is robust for position variations and consideration of energy besides the simplicity of adding more robots.

Keywords – Mobile robots navigation, Robotics control systems, Trajectory control, Autoencoder, Sensor fusion.

LISTA DE FIGURAS

1	Representação do sistema de coordenadas inercial do robô móvel de quatro rodas com direção deslizante para a modelagem cinemática.	36
2	Representação do centro de massa do robô.	38
3	Modelo que calcula uma estimativa para os ângulos de rotação e duas estimativas independentes para a velocidade angular.	44
4	Representação do autoencoder e a estimativa da velocidade angular de saída é feita pela média das duas. $\Omega(t) = \Omega_1(t) + \Omega_2(t)$	45
5	Arquitetura NHC	48
6	Relacionamento de Herança entre um nó e um nó derivado.	58
7	Representação da <i>Scene Tree</i> e Visualização 3D.	60
8	Configuração Inicial do ambiente <i>Rectangle Arena</i>	62
9	Configuração do ambiente <i>Rectangle Arena</i> para semeadura.	62
10	Representação lateral e frontal do Terreno irregular com $xDimension = 4$ e $yDimension = 10$	63
11	Representação lateral e frontal do Terreno irregular com $xDimension = 3$ e $yDimension = 20$	63
12	Representação em <i>Scene Tree</i> do nó <i>Shape</i> e seus nós derivados <i>geometry</i> e <i>appearance</i>	64
13	Representação em <i>Scene Tree</i> do nó <i>Joint</i> e todos seus nós derivados. . . .	65
14	Representação em <i>Scene Tree</i> do nó <i>Robot</i> e todos seus nós derivados. . . .	66
15	Representação em <i>Scene Tree</i> do nó <i>Robot</i> e todos seus nós derivados. . . .	66
16	Conexões conceituais para montagem do robô no ambiente de simulação. . .	67
17	Representação de como configurar o controle no Robô através da <i>Scene Tree</i> . .	67
18	Representação do controle já selecionado no nó <i>Robot</i>	68
19	Pontos da trajetória a ser seguida em terreno regular.	70

20	Representação do ambiente em terreno regular com um robô.	70
21	Trajetória de referência e trajetória real executada pelo robô em terreno regular.	71
22	Esforço de controle de posição para um robô em terreno regular quando comparado com o perfil velocidade.	71
23	Esforço de controle do ângulo θ para um robô em terreno regular quando comparado com o perfil velocidade.	72
24	Erro de posição na trajetória de um robô em terreno regular.	72
25	Erro do ângulo θ na trajetória de um robô em terreno regular.	73
26	Representação do ambiente em terreno regular com dois Robôs.	73
27	Trajetórias de referência e trajetórias reais para dois robôs em terreno regular.	74
28	Esforço de controle de posição para cada um dos dois robôs em terreno regular quando comparado com o perfil velocidade.	74
29	Esforço de controle do ângulo θ para cada um dos dois robôs em terreno regular quando comparado com o perfil velocidade.	75
30	Erro de posição na trajetória de dois robôs em terreno regular.	75
31	Erro de ângulo θ na trajetória de dois robôs em terreno regular.	76
32	Pontos da trajetória a ser seguida em terreno irregular.	77
33	Trajetórias de referência e trajetórias reais para dois robôs em terreno irregular.	77
34	Esforço de controle de posição para cada um dos dois robôs em terreno irregular quando comparado com o perfil velocidade.	78
35	Esforço de controle do ângulo θ para cada um dos dois robôs em terreno irregular quando comparado com o perfil velocidade.	78
36	Erro de posição na trajetória de dois robôs.	79
37	Erro de ângulo na trajetória de dois robôs em terreno irregular.	79
38	Representação da simulação em terreno irregular com obstáculo cerca.	80
39	Controle para dois robôs em terreno irregular com obstáculo cerca.	81

40	Esforço de controle de posição para cada um dos dois robôs em terreno irregular com obstáculo quando comparado com o perfil velocidade.	81
41	Esforço de controle do ângulo θ para cada um dos dois robôs em terreno irregular com obstáculo quando comparado com o perfil velocidade.	82
42	Erro de posição na trajetória de dois robôs em terreno irregular com obstáculo cerca.	82
43	Erro de ângulo θ na trajetória de dois robôs em terreno irregular com obstáculo cerca.	83

LISTA DE SÍMBOLOS

CoM Centro de massa

X_l, Y_l, Z_l Sistema de coordenadas local

v velocidade linear

v_x componente da velocidade linear na direção x

v_y componente da velocidade linear na direção y

w velocidade de rotação angular

q vetor de estados

X coordenada do vetor de estados na direção x

Y coordenada do vetor de estados na direção y

θ ângulo de orientação do vetor de estados

\dot{q} vetor de velocidades generalizadas

\dot{X} vetor de velocidades locais na direção x

\dot{Y} vetor de velocidades locais na direção y

$\dot{\theta}$ orientação do sistema de coordenadas em relação ao sistema de coordenadas inercial.

v_i vetor de velocidade da i -ésima roda representado no sistema de coordenadas local

r_i raio efetivo de rolagem

v_{ix} componente de velocidade longitudinal do vetor de velocidade

v_t velocidade tangencial

d_x variação da posição do ponto no eixo x

d_y variação da posição do ponto no eixo y

a, b, c parâmetros cinemáticos positivos do robô

v_{xR} coordenadas longitudinais das velocidades das rodas do lado direito

v_{xL} coordenadas longitudinais das velocidades das rodas do lado esquerdo

v_{yF} coordenadas de velocidade laterais das velocidades da roda da frente

v_{yB} coordenadas de velocidade laterais das velocidades da roda de trás

w_w velocidade angular

w_L velocidade angular da roda esquerda

w_R velocidade angular da roda direita

η nova entrada de comando introduzida a nível cinemático

t tempo fixo

LISTA DE ACRÔNIMOS

LARS Latin American Robotics Symposium

SBR Brazilian Symposium on Robotics

WRE Workshop on Robotics Education

PID Proporcional Integral Derivativo

TVP Perfil de velocidade trapezoidal

ICR Centro instantâneo de Rotação

GPS Global Positioning System

WMRs Robôs móveis com rodas

IoT Internet das coisas

IT2FIS Algoritmo de inferência fuzzy tipo 2 de intervalo

LoRaWAN Rede ampla de longo alcance

NHC Controle Hierárquico Aninhado

API Interface de programa de aplicação

M2M Máquina para máquina

R1 Robô 1 utilizado na simulação

R2 Robô 2 utilizado na simulação

SUMÁRIO

1	Introdução	23
1.1	Motivação	23
1.2	Objetivos	25
1.3	Trabalhos publicados	25
1.4	Revisão da Literatura	25
1.4.1	Robôs autônomos terrestres	25
1.4.2	Aplicações em agricultura	28
1.4.3	Geração de trajetória	30
1.5	Estrutura da tese	34
2	Robótica móvel e fusão de sensores	35
2.1	Robótica móvel	35
2.1.1	Definições	35
2.1.2	Modelagem Cinemática	35
2.2	Sensores	40
2.2.1	Sensores instalados	40
2.2.2	Fusão de dados	41
3	Geração e controle de trajetória	47
3.1	Controle de trajetória	47
3.1.1	Método de geração de trajetória por otimização	47
3.2	Controle de movimento e posição	48
3.2.1	Descrição do controlador	49
3.2.2	Atualização da trajetória conforme espaço percorrido	50

3.2.3	Pseudo código do Algoritmo do Morcego	52
4	Simulação para robótica móvel	55
4.1	Panorama	55
4.1.1	Gazebo™	55
4.1.2	CoppeliaSim™	56
4.1.3	Webots™	56
4.2	Plataforma de Simulação	57
4.2.1	Gráfico de Nós	58
4.2.2	Tipos de Nós	59
4.2.2.1	Nós VRML97	59
4.2.2.2	Nós específicos do Webots	59
4.2.3	<i>World</i> e <i>Scene Tree</i>	60
4.3	Ambiente	61
4.3.1	Terreno regular - <i>Rectangle Arena</i>	61
4.3.2	Terreno irregular - <i>Uneven Terrain</i>	62
4.4	Robô	63
4.5	Programa	68
5	Resultados	69
5.1	Condições do problema	69
5.2	Terreno Regular	69
5.2.1	Situação 1 - Terreno regular com um robô	70
5.2.1.1	Controle e trajetória	71
5.2.2	Situação 2 - Terreno regular com dois robôs	73
5.2.2.1	Controle e trajetória	74
5.3	Terreno irregular	76
5.3.1	Situação 3 - Terreno irregular com dois robôs	77

5.3.1.1	Controle e trajetória	78
5.3.2	Situação 4 - Terreno irregular com obstáculo cerca	80
5.3.2.1	Controle e trajetória	80
6	Conclusão	85
	Referências	87

1 INTRODUÇÃO

1.1 Motivação

A evolução da robótica móvel tem contribuído para a agricultura nos últimos anos (HACKENHAAR; HACKENHAAR; ABREU, 2015). Os principais desafios encontrados na agricultura relacionados à robótica móvel são tarefas de implementos agrícolas para semeadura, preparo da terra para plantio, ou aplicação de insumos. Os robôs móveis autônomos potencialmente podem contribuir para a solução destes problemas. Tais aplicações se justificam em um contexto de terrenos irregulares com grande variedade de cultivos. Além disso, a agroecologia (agricultura com preocupação ecológica) é incompatível com grandes áreas planas e lavouras uniformes passíveis de automação baseada em grandes máquinas como colheitadeiras (Fendt, 2020) ou tratores.

Apresenta-se aqui uma alternativa para substituir um trator convencional por um enxame de robôs tratores elétricos menores e de menor potência, que juntos possuem a mesma capacidade operacional, porém com menor custo energético para arar em determinada lavoura. Conclui-se que o enxame é capaz de realizar tal atividade e é proposto um projeto preliminar para construção de um trator elétrico autônomo dentro das metodologias de operação do enxame. No presente trabalho, é estudado um caso particular de controle de trajetória em dois robôs móveis, que devem realizar uma determinada atividade de forma otimizada. O robô percorre uma trajetória dentro de uma área pré-definida em um intervalo de tempo fixo t . Em seguida, avalia-se o gasto energético, a área coberta e o esforço de controle. O rastreamento de trajetória utiliza sensores inerciais amplamente disponíveis para robótica móvel e não requer medições de GPS, que são imprecisas para escalas de plantio menores e pequenas culturas.

Atualmente, nas operações agrícolas utilizando robôs móveis, uma das dificuldades encontradas é determinar um caminho ótimo, que minimize os gastos energéticos da navegação e garanta a conclusão da tarefa por completo, (MOYSIADIS et al., 2020; DURMUS et al., 2015). Em (MAHMUD et al., 2019) um algoritmo multi-objetivo é

implementado para resolver o problema de planejamento de trajetória para pulverização de pesticidas em estufas. Neste problema específico, adaptou-se uma solução para um problema de roteamento de veículos encontrado na pesquisa operacional, as rotas entre as plantas são geradas usando um planejador de caminho probabilístico com base no ambiente virtual. Este é projetado com base no ambiente real da estufa, de forma a visualizar a operação agrícola dentro da mesma.

Outra questão frequentemente discutida é a navegação do robô em um ambiente com obstáculos, quando vivenciada em um ambiente agrícola, esta situação pode ser mais complexa. Em (GAO et al., 2018), são utilizados robôs móveis com rodas (WMRs) e são estudados seus respectivos problemas e mecanismos de navegação, analisando-se detalhadamente os métodos de resolução de subproblemas como mapeamento, localização e planejamento de caminho. O trabalho visa a especial complexidade do ambiente agrícola, prospecta a solução para o problema de navegação de WMRs na engenharia agrícola e propõe o direcionamento de pesquisas para solucionar os problemas de navegação precisa.

Em (WANG et al., 2022) é abordada a agricultura de precisão, destacando-se a produção em larga escala. Os autores se concentraram nas aplicações de técnicas de aprendizado profundo baseadas em imagens, como mapeamento de terras, classificação de culturas, monitoramento de estresse biótico/abiótico e previsão de rendimento. Também é realizada uma investigação abrangente com ênfase especial em aprendizado profundo e sensoriamento remoto. Deste último, são avaliados métodos não destrutivos e não invasivos, como a integração de sensores de imagem de alto desempenho, plataformas móveis não tripuladas e técnicas de aprendizado profundo são usadas para modelar o extenso volume de dados coletados por esses sensores como forma de obter resultados rapidamente e eficientemente e, em seguida, realizar o gerenciamento da cultura com base na tomada de decisões informadas.

Semelhante ao que foi feito em (WANG et al., 2022), em (GAO et al., 2022) um algoritmo de estimativa de posição aprimorado baseado em fusão multisensor e rede neural de autoencoder é usado para modelar a equação de estados com ruído e a otimização é realizada por meio de filtro de Kalman estendido. O objetivo deste último trabalho, juntamente com a fusão destes sensores, é melhorar o desempenho da previsão de posição.

1.2 Objetivos

Neste trabalho, estuda-se um caso particular de controle de trajetórias em robôs móveis, em que os mesmos devem realizar determinada atividade de forma otimizada e colaborativa.

Os dois robôs simulados percorrem uma trajetória dentro de uma área pré-definida em um intervalo de tempo combinado. Após o intervalo de tempo, avalia-se o gasto de energia, a área coberta e o esforço de controle. As estimativas para o acompanhamento da trajetória utilizam sensores inerciais amplamente disponíveis para robótica móvel e não necessitam de medidas de GPS, que são imprecisas para escalas de plantio de vegetais de menor porte e lavouras pequenas.

1.3 Trabalhos publicados

Anteriormente a esta dissertação, um trabalho foi desenvolvido e apresentado por VENTURA M. L. N.; MAZONI A. F; COLÓN D (2022) no *Latin American Robotics Symposium (LARS) 2022*, *Brazilian Symposium on Robotics (SBR) 2022*, e *Workshop on Robotics in Education (WRE) 2022*.

Este trabalho resume parte do desenvolvimento desta dissertação, ou seja, o controle de um robô móvel seguindo uma trajetória de semeadura utilizando sensores inerciais. A dificuldade de estimativa usando sensores inerciais é resolvida usando uma estrutura de autoencoder, treinada por meio da simulação. O controlador é testado no simulador Webots™ e possui uma estrutura com dois PIDs, um para a posição e outro para a direção do robô. Os parâmetros de ambos os controladores são ajustados usando-se o algoritmo de otimização do morcego.

1.4 Revisão da Literatura

1.4.1 Robôs autônomos terrestres

Robôs móveis autônomos se diferenciam dos demais pela sua capacidade de conseguir se mover de forma independente, devido a capacidade de tomada de decisão a partir de estímulos e percepções obtidos do ambiente através de alguma fonte de dados como informações obtidas através de sensores. Em (RUBIO; VALERO; LLOPIS-ALBERT, 2019) é feita uma revisão de literatura a respeito de robôs móveis, desde seus tipos de locomoção,

que conseguem correr, pular, andar até os equivalentes biológicos, ainda é discutido os campos da robótica, tais como robôs móveis com rodas, que é o caso deste trabalho, robôs com pernas, voadores, visão robótica, inteligência artificial. Por último é discutido as pro-pensões futuras de tematicas como inteligência artificial, direção autônoma, comunicação em rede, nanorobótica, aplicadas em distintos campos tais como medicina, indústria, serviços, agricultura.

Robôs autônomo realizam interações para terem a capacidade de executarem as ati-vidades e cumprir seus objetivos. Em (INGRAND; GHALLAB, 2017) discutem-se sobre sistemas robóticos e suas funcionalidades estendidas, robustas e mais adaptáveis através de funções de deliberação. Essas funções se resumem em planejar, agir, monitorar, obser-var e aprender, definidos da seguinte maneira (INGRAND; GHALLAB, 2017):

- *Planejamento*: é a combinação entre busca e previsão para gerar uma trajetória em um determinado espaço de ação abstrato a partir de modelos preditivos do ambiente e ações viáveis para alcançar um propósito definido.
- *Ação/ Agir*: aprimoramento das ações planejadas em comandos pertinentes e ade-quados para a circunstância atual e atende aos eventos; tanto o refinamento quanto a reação podem depender de habilidades, ou seja, um conjunto de funções de malha fechada. Uma habilidade desempenha um fluxo ou sequência contínua de entrada de estímulo de sensores, gerando saída para atuadores, com o intuito de excitar forças motoras e controlar a realização correta das ações escolhidas.
- *Observação*: Reconhece, detecta e caracteriza os estados do mundo. Além disso, através de planos e ações, realiza a tarefa utilizando os processos em níveis para tal. Exemplificando, é possível a utilização de sensores para captura dos dados, através da detecção das ações e do planejamento para coletar esta informações.
- *Monitoramento*: é a conferência entre o que é previsto para a realização da ati-vidade do robô e o que é observado no mundo. É a detecção e interpretação de discrepâncias, realização de diagnósticos e de ações para controlar e corrigir os er-ros e desvios quando necessário. O controle do raciocínio de metas é monitorado no nível da missão (INGRAND; GHALLAB, 2017). As metas e compromissos são mantidos em perspectiva; é avaliado a pertinência a partir das evoluções observa-das de novas oportunidades, ou insucessos; implicando, a partir dessa avaliação se alguns compromissos devem ser abandonados, e quando e como atualizar as metas atuais.

- *Aprendizagem*: é possível que um ou vários robôs adaptem e aprimorem os modelos necessários através da experiência adquirida e aprendam ativamente a agir deliberadamente, se aperfeiçoando de forma cada vez mais eficiente em sua tarefa.

Ainda em (INGRAND; GHALLAB, 2017) é discutido que as funções deliberativas podem se sobrepor dependendo da implementação e organização da arquitetura operacional. É necessário analisar os requisitos e a estrutura da malha fechada, desde o loop interno, considerado do mais interno, por conter funções senso-motoras até o loop mais externo.

Em (KAGAN; SHVALB; BEN-GAL, 2019), um agente é definido como sendo um sistema computacional em algum ambiente, capaz de agir de forma autônoma e concluir um determinado objetivo. *Swarm* de robôs ou robótica de enxame é o estudo como um determinado número de robôs podem apresentar um comportamento coletivo e descentralizado a partir de interações locais entre os mesmos e o ambiente.

Esse tipo de organização apresenta características atraentes em várias tarefas, quando comparado a um robô individual pois podem apresentar algumas características desejáveis como paralelismo, permitindo o cumprimento de determinado objetivo de forma mais rápida, e robustez, pois apresenta maior tolerância a falha de apenas um robô e permite concluir a tarefa, sendo adaptável a diferentes tipos de aplicações e objetivos (KAGAN; SHVALB; BEN-GAL, 2019). Entre os desafios dos enxames robóticos está o foco em um atividade do enxame como uma unidade, comunicação e interação entre os agentes, as divisões de trabalho e os problemas de otimização e controle mais complexos que um robô individual (KAGAN; SHVALB; BEN-GAL, 2019) .

Em (DING et al., 2022) é realizado um estudo de caso sobre quatro principais aspectos: aplicações, tecnologias de percepção, comunicações de robôs e mecanismos de controle entre robôs móveis autônomos e robôs tradicionais. São identificados desafios como adaptação ao ambiente, controle robusto e comunicação estável. As tecnologias de percepção têm como principal intuito, a obtenção de informações do ambiente externo através de dispositivos e de diversas formas, com o intuito de auxiliar os robôs móveis a potencializar o conhecimento sobre o ambiente externo. As comunicações do robô são utilizadas entre grupos e enxames para compartilhar informações e suporte entre si. Os mecanismos de controle otimizam e permitem os robôs concluir a tarefa correspondente com base nas tecnologias de percepção e comunicação. Ainda em (DING et al., 2022) é analisado um ambiente complexo em que é necessária observação do ambiente em tempo real a fim de evitar obstáculos e poder mudar a trajetória para conclusão de sua tarefa de maneira otimizada através de uma integração entre percepção, comunicação e controle

por meio de planejamento de trajetória e sistemas multiagentes.

1.4.2 Aplicações em agricultura

Com a globalização e a diminuição da população na zona rural, o custo da produção agrícola aumentou. Assim é cada vez mais necessário aumentar o rendimento do trabalho no campo, e otimizar os resultados de produção e melhorar a qualidade de vida dos agricultores (Empresa Brasileira de Pesquisa Agropecuária, 2023).

Conforme os agricultores foram usufruindo da tecnologia para maiores ganhos, surgiu um viés para utilização de grandes campos para produção, tornando as operações de pequena escala impraticáveis (KING, 2017). Porém, com as evoluções na robótica, tecnologias de detecção e automação criaram a oportunidade de modelo de negócio em áreas menores e viabilizaram economicamente a produção para os pequenos agricultores através da utilização de robôs inteligentes (KING, 2017).

Segundo a (Empresa Brasileira de Pesquisa Agropecuária, 2023), a automação na agricultura resume-se como um sistema no qual os processos operacionais de produção agrícola são monitorados, controlados e executados por meio de máquinas e ou dispositivos, sejam eles, mecânicos, eletrônicos ou computacionais, para ampliar a capacidade do trabalho humano.

Dessa forma, o objetivo da automação é aumentar a produtividade, otimizar o tempo, reduzir perdas e melhorar a qualidade, tanto em produtos como também a vida de trabalhadores rurais (Empresa Brasileira de Pesquisa Agropecuária, 2023). Assim como no Brasil, outros países emergentes, como a Índia, também tiveram a agricultura como uma das atividades protagonistas para o crescimento econômico do país (PURANIK et al., 2019), mas com a era da transformação digital é preciso aproveitar da melhor forma espaços produtivos em que não é possível cultivos em larga escala. Desse modo, (PURANIK et al., 2019) analisa um desafio utilizando IoT e automação, com o intuito de gerenciar a execução do trabalho agrícola e assim permitir que os agricultores possam criar estratégias de cultivo e otimização de produção e resultados. Para isso, (PURANIK et al., 2019) propõe alguns temas: automatização da manutenção, controle de inseticidas e pesticidas, gerenciamento de água e monitoramento de culturas.

Em países da Europa como Holanda e Inglaterra, é habitual a utilização de estufas para produção de alimentos como frutos e vegetais, e a automação contribui com o intuito de reduzir os custos e aumentar a qualidade (KING, 2017) através de dispositivos que

monitoram o crescimento de vegetais. Além disso, vem sendo desenvolvidos colhedores robóticos. Os colhedores robóticos apresentam como principais desafios, a identificação com rapidez e precisão do formato do vegetal, como a colheita de pimenta, em que é necessário evitar cortar o caule principal da planta. A solução neste caso, é o desenvolvimento de um software rápido e preciso através de aprendizado profundo e utilização de câmeras para processamento de imagem e treinamento de uma rede neural (KING, 2017).

Em (JHA et al., 2019), são estudadas distintas práticas de automação agrícola, através de aprendizado de máquina, aprendizado profundo, comunicações sem fio, e IoT para mitigar e resolver questões como falta de irrigação e gerenciamento de água, falta de armazenamento e gerenciamento da produção agrícola, doenças nas culturas e produtividade dos campos devido a utilização de pesticidas nocivos.

Ainda em (JHA et al., 2019) é discutido como a automação das práticas agrícolas melhora e aumenta a fertilidade do solo. Por último, propõe-se um sistema que pode ser implementado em uma fazenda botânica para identificação de folhagens e irrigação utilizando IoT.

A Internet das Coisas (IoT) pode ser definida como sendo uma rede de dispositivos para comunicação de máquina para máquina (M2M) baseado em Internet com e sem fio (KIM; LEE; KIM, 2020). Na Coreia, (KIM; LEE; KIM, 2020) são estudados casos de aplicação de IoT no setor agrícola para expandir sua aplicação e é considerada uma tecnologia revolucionária por poder ser aplicada na produção agrícola o ano todo sem sazonalidades. Para isso, foram utilizados dispositivos de comunicação como Wi-Fi, rede de área ampla de longo alcance (LoRaWAN), comunicação móvel (por exemplo, 2G, 3G e 4G), ZigBee e Bluetooth e realizada comparação e análise de resultados e limitações. As aplicações foram consideradas na automação da agricultura em sistemas de controle, monitoramento, gerenciamento, controle e máquinas não tripuladas (KIM; LEE; KIM, 2020).

Em (TIAN et al., 2020), é estudado o desenvolvimento da automação agrícola em pequenas áreas e lavouras para obter vantagens como baixo custo, alta precisão e eficiência, através de visão computacional. Esta técnica consiste em utilizar uma câmera e um computador para identificação, rastreamento e medição de alvos para processamento de imagem. Como principais desafios, tem-se a necessidade de construir conjuntos de dados em larga escala e a necessidade de profissionais capacitados (TIAN et al., 2020). A robustez no desempenho em ambientes complexos tende a se combinar com outras tecnologias inteligentes como aprendizado profundo, empregada sob a ótica do gerenciamento

da produção agrícola, com base em conjuntos de dados de grande escala, com o intuito de resolver problemas agrícolas e conseqüentemente melhorarem quesitos econômicos, geral e robusto dos sistemas de automação agrícola.

Embora existam diversos exemplos de aplicações para robótica, a aplicação em qualquer escala produtiva está limitada pela dificuldade de modularidade e ausência de infraestrutura. Os modelos de pesquisa e protótipo, por mais avançados que sejam, ainda dependem de conexões sem fio, baterias de longa duração e carregamento próximo bem como plataformas específicas de *software*.

Os projetos também precisam de componentes modulares, na medida em que outros projetos consigam reaproveitar os progressos anteriores e mesmo que se consigam compartilhar recursos de infra-estrutura. O trabalho (HAJJAJ; SAHARI, 2016) investigam esses detalhes da literatura recente apontando os principais obstáculos.

Em (OLIVEIRA; MOREIRA; SILVA, 2021), apresentam-se diversos modelos de robôs comerciais. São todos produzidos em pequena escala e a maior parte no mercado europeu. De qualquer forma, esses projetos apresentam uma tentativa de modularização no que diz respeito à forma de operação e uso de diferentes ferramentas para conjuntos específicos de tarefas. Algumas ferramentas são inclusive compartilhadas de outras áreas de automação e comercializadas em maior escala por fabricantes tradicionais. Uma perspectiva futura de integração com conhecimentos agrícolas nos referidos produtos comerciais é apresentada em (DUCKETT et al., 2018). Com essa filosofia, os trabalhos procuram explorar tarefas comuns de agricultura a ser realizadas por robôs, como aspersão de aditivos químicos, semadura, colheita e poda, (KULBACKI et al., 2018; LYTRIDIS et al., 2021; BECHAR; VIGNEAULT, 2016).

1.4.3 Geração de trajetória

Para os robôs móveis autônomos realizarem sua tarefa, é necessário um processo para orientação e controle do robô, que realize o rastreamento da trajetória, ou que mantenha o robô no caminho gerado (DIRIK M., 2020). Este método é chamado de planejamento de caminho (DIRIK M., 2020), e exige o conhecimento prévio da classificação do ambiente, entre estático ou dinâmico, para assim definir a categoria do algoritmo que irá obter a solução mais adequada. Existem duas categorias para o algoritmo de planejamento de caminho: Algoritmos globais (off-line), para ambientes estáticos em que é exigido o modelo do ambiente e ser completamente conhecido, como A*, algoritmo genético, algoritmo de Dijkstra, PRM, ou algoritmos locais (on-line) (DIRIK M., 2020).

Em (DIRIK M., 2020) é proposta uma estrutura de controle para robô móvel com rodas (WMR), focando na implementação de algoritmos livres em tempo real e que realiza o planejamento de caminho através da captação de dados de um sistema de servos visual com câmera em ambiente dinâmico e tempo real. Resume-se em três etapas, sendo a primeira a estruturação deste algoritmo baseado nos dados capturados pelo sensor de visão; Na segunda etapa, a determinação dos parâmetros iniciais do algoritmo de planejamento de caminho e as coordenadas de caminho utilizando essas informações obtidas em tempo real. Na terceira etapa, o processo para seguir o caminho utilizando a estrutura de controle para manter o robô no caminho gerado. A estratégia de controle minimiza erros sistemáticos ou não sistemáticos para o controle de movimento do robô e realiza comparativos entre o algoritmo de inferência fuzzy tipo 2 de intervalo (IT2FIS), fuzzy tipo 1 e PID Fuzzy. Por fim, o IT2FIS encontrou as melhores soluções em comparação com os outros métodos quanto à análise estatística. O sistema proposto foi implementado de forma bem sucedida em vários mapas, gerando caminhos livres de colisão de forma eficiente e consistente e capaz de guiar o robô no caminho desejado em tempo real.

Similar à (DIRIK M., 2020), (KOSTJUKOV; MEDVEDEV; PSHIKHOPOV, 2021) estuda um procedimento para correção de trajetória de uma plataforma robótica em uma aeronave, com o intuito de limitar a probabilidade de sua detecção em campo de um número finito de fontes repelentes em movimento (RTP). As fontes são definidas por um modelo matemático com fator de contra ação ao RTP. O estudo é baseado na função de probabilidade característica de um sistema de fontes repelentes, analisando a influência das mesmas no RTP em movimento. Este método se baseia na resolução de problema de otimização local com correção de trechos individuais da trajetória inicial, considerando a localização de fontes repelentes específicas com parâmetros pré determinados em sua vizinhança. As fontes são caracterizadas pelo potencial, frequência de impacto, raio de ação e parâmetros do decaimento do campo (KOSTJUKOV; MEDVEDEV; PSHIKHOPOV, 2021).

Para a probabilidade de uma conclusão bem sucedida, o critério utilizado é a otimização da trajetória do alvo. A trajetória é ajustada iterativamente e considera o valor alvo da probabilidade de passagem. A principal restrição à variação da trajetória original é o desvio máximo permitido da trajetória alterada da original. Dessa forma, são buscados os correspondentes máximos locais da probabilidade de passagem do RTP no campo de várias fontes aleatoriamente localizadas e orientadas, em alguma vizinhança da trajetória inicial (KOSTJUKOV; MEDVEDEV; PSHIKHOPOV, 2021).

Desta forma o principal desafio é a seleção da forma geral do funcional em cada

ponto da curva inicial, bem como seus coeficientes de ajuste. Percebe-se que a escolha desses coeficientes é um procedimento adaptativo, cujas variáveis de entrada são valores geométricos característicos que descrevem a trajetória atual no campo fonte. Ainda, procedimentos padrão de suavização mediana são utilizados para eliminar oscilações que ocorrem como resultado da localidade do procedimento proposto. Por fim, resultados da simulação obtiveram a alta eficiência do procedimento proposto para corrigir a trajetória previamente planejada (KOSTJUKOV; MEDVEDEV; PSHIKHOPOV, 2021).

O trabalho de (ALMASRI; ALAJLAN; ELLEITHY, 2016) projeta um sistema de robôs móveis com desafio de prevenção de obstáculos em tempo real e seguir um caminho em linha em ambiente desconhecido a partir de uma técnica de utilização de sensores infra vermelhos de baixo custo e aplicações de controle em tempo real. O seguidor de linha resume-se a um robô autônomo que detecta uma linha, que pode ser visível como uma linha preta em uma área clara ou invisível, em um campo magnético, e continua a seguindo. Existem algumas abordagens para o problema citado, tais como seguimento de linha, seguimento de objeto e rastreamento de caminho. Para realizar a tarefa, (ALMASRI; ALAJLAN; ELLEITHY, 2016) dividiu em três etapas: Processamento de imagem através dos dados para que seja possível capturar a largura da linha, ajustar o robô para seguir a linha pré definida utilizando sensores infra vermelhos próximos ao mesmo e por último, controlar a velocidade do robô baseado na condição existente na linha. Em síntese, (ALMASRI; ALAJLAN; ELLEITHY, 2016) demonstra que o robô obtém sucesso em sua tarefa e consegue seguir curvas congestionadas e evita obstáculos ou objetos que surgem seu caminho. O simulador Webots™ foi utilizado para validar a eficácia da técnica proposta.

Além de planejamento de caminho, outro problema em aberto para robôs móveis com rodas é a geração de trajetória, que consiste em leis de temporização associadas a rodas com o intuito do robô atingir o estado alvo estabelecido (KIM, 2020). É visto em (KIM, 2020) que a trajetória deve ser planejada sempre que um novo estado de destino estiver disponível, caso contrário o robô pode não conseguir responde a tempo a qualquer evento imprevisto ou indesejado. Para que isso aconteça, existem duas condições contraditórias que devem ser atendidas simultaneamente: a otimização do tempo e a eficiência computacional. O artigo apresenta a geração de trajetória ótima de tempo próximo com custos computacionais minimizados, em um caso em que o robô deveria trilhar um novo estado de destino suavemente, em tempo mínimo com restrições física de torque e velocidades máximos existentes, minimizando os custos computacionais. Foi planejado uma linha reta e a utilização de uma curva de Bézier de quinta ordem para gerar uma volta de curvatura

contínua para a trajetória, enquanto as restrições cinemáticas e dinâmicas foram mapeadas para calcular o tempo mínimo ao longo da curva. Ainda foram adotadas abordagens heurísticas para reduzir os cálculos para encontrar soluções ótimas e uma função spline cúbica foi utilizada para estimar funções de restrição com nós selecionados baseados na curvatura do caminho.

Todo o estudo de (KIM, 2020) foi testado em ambiente de simulação e com uma trajetória nominal, ou seja, uma trajetória de referência para uma malha de controle sob as hipóteses de restrição não holonômica. No equipamento que ocorreu o teste, o tempo de computação para gerar a trajetória ótima de tempo completa, foi de apenas da ordem de milissegundos, uma redução considerável em relação a outros métodos existentes, que geralmente levam dezenas de segundos. Por outro lado, as trajetórias propostas apresentaram maior lentidão que verdadeiras trajetórias ótimas de tempo, devido ao fato do torque ser saturado apenas uma ou duas vezes por segmento de caminho por causa de suposições como uma trajetória com perfil de velocidade trapezoidal (TVP) onde apenas três valores constantes máximos (aceleração, velocidade e desaceleração) são necessários para determinar completamente uma trajetória para uma linha reta e uma velocidade constante ao longo de uma curva.

No trabalho de (WALAMBE et al., 2016) é implementado um protótipo de quatro rodas em escala 1:10 que foca na geração de trajetória baseada em curvas spline para planejar o movimento. O carro pode ser modelado como um sistema não holonômico apresentando restrições de movimento lateral. Assim como em (KIM, 2020), a geração de trajetória em (WALAMBE et al., 2016) baseada em spline fornece trajetórias de caminho que sejam contínuas, suaves e otimizadas. Como mencionado em (WALAMBE et al., 2016) o algoritmo de planejamento de movimento é baseado no modelo não holonômico de um robô de quatro rodas que é diferencialmente plano por natureza. Dessa forma, a maior contribuição do trabalho foi o desenvolvimento de trajetórias baseadas em curvas splines que melhoram as singularidades paramétricas decorrentes da abordagem baseada em planicidade diferencial. As trajetórias spline também cuidam das singularidades que são observadas no processo de otimização se a interpolação baseada em polinômios cúbicos for implementada.

Além da simulação, o trabalho de (WALAMBE et al., 2016) realizou implementação em *hardware* do robô com rodas, e utilizando os conceitos de planicidade e nivelamento diferencial, foram derivadas entradas de controle que direcionam o veículo ao longo da trajetória. Em resumo, (WALAMBE et al., 2016) apresentou o trabalho realizado para o desenvolvimento de um algoritmo para geração de trajetória suave e contínua para um

robô do tipo carro. O planejamento de movimento não holonômico obteve trajetórias precisas e suaves, enquanto que usando o nivelamento diferencial, as entradas de controle foram geradas com a integração real das equações diferenciais do sistema. Com o intuito de impedir as singularidades observadas nas trajetórias baseadas em polinômios cúbicos, as trajetórias baseadas em splines são implementadas. As variações entre os resultados obtidos em ambiente simulado e implementação do *hardware* é devido à inexistência de um controlador realimentado.

1.5 Estrutura da tese

Esta dissertação foi organizada da seguinte forma: No capítulo (2), intitulado de Robótica Móvel e Fusão de Sensores, apresenta-se o modelo cinemático e a fusão de dados. O capítulo 3 sobre Geração e Controle de Trajetória, discutindo sobre o método de geração de trajetória por otimização desenvolvido e o controle de movimento e posição e sua atualização conforme o espaço percorrido, ainda em tempo, o pseudo-código do Algoritmo do morcego e o algoritmo de controle e função de custo. No capítulo 4, intitulado Simulação para Robótica Móvel é dado um panorama sobre plataforma de simulação e mostrado como foi realizado a simulação e o programa desenvolvido. No capítulo 5, intitulado Resultados, são analisados e discutidos os gráficos obtidos em ambiente simulado. Por fim, no capítulo 6, são apresentadas conclusões e considerações finais.

2 ROBÓTICA MÓVEL E FUSÃO DE SENSORES

2.1 Robótica móvel

2.1.1 Definições

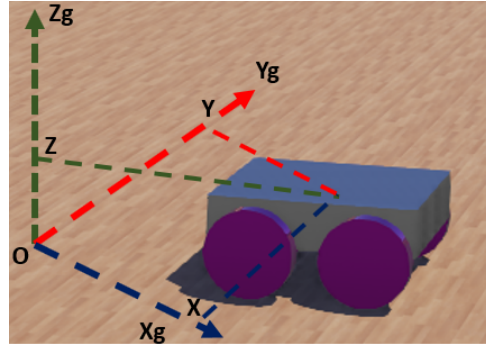
A robótica tem sido aplicada com sucesso na indústria manufatureira, principalmente através de braços robóticos e manipuladores, que permitiam uma linha de montagem e produção escalável, através de tarefas repetitivas. Apesar de todo esse avanço de tecnologias, esse tipo de robô apresenta uma séria desvantagem: não possuir mobilidade. Um manipulador apresenta um movimento dentro de um espaço de trabalho em seu braço, mas está fixo em um ambiente. Um robô móvel apresenta maior flexibilidade em suas funções, tornando-se mais eficaz (SIEGWART, 2004).

2.1.2 Modelagem Cinemática

Conforme apresentado anteriormente, o tipo de robô utilizado neste trabalho, um robô móvel de quatro rodas com direção deslizante, como apresentado Figura 1. As seguintes premissas iniciais são adotadas (algumas serão posteriormente modificadas): o robô está localizado em uma superfície plana com uma base inercial ortogonal (X_g, Y_g, Z_g) . Um sistema local de coordenadas denotado por (X_l, Y_l, Z_l) é atribuído ao centro de massa do robô, que será representado por $CoM = (X, Y, Z)$, no sistema de coordenadas inercial (COSTA, 2019), representado pela Figura 1.

Supondo que o robô se movimenta somente em um plano com velocidade expressa no sistema de coordenadas com a velocidade linear sendo $v = [v_x, v_y, 0]^T$, (COSTA, 2019) e velocidade de rotação angular $w = [0, 0, w]^T$, o vetor de estados que representa as coordenadas do robô, é descrito por $q = [X, Y, \theta]^T$, sendo interpretado como a a posição do CoM , X e Y , e a orientação θ do sistema de coordenadas em relação ao sistema de coordenadas inercial. Dessa forma, $\dot{q} = [\dot{X}, \dot{Y}, \dot{\theta}]^T$ descreve o vetor de velocidades

Figura 1: Representação do sistema de coordenadas inercial do robô móvel de quatro rodas com direção deslizante para a modelagem cinemática.



Fonte: Autora.

generalizadas que expressa as velocidades lineares e angulares do robô. As variáveis \dot{X} , \dot{Y} estão relacionadas ao sistema de coordenadas do vetor de velocidades locais por uma matriz de rotação, pelas equações 2.1 e 2.2, considerando que, o movimento é planar $\dot{\theta} = w$:

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} \quad (2.2)$$

Desde que a equação descreva somente a cinemática do corpo livre, (2.2) não impõe nenhuma restrição no movimento planar do robô com direção deslizante.

Suponha que a i -ésima roda gire com velocidade angular $w_i(t)$, onde $i = 1, 2, 3$ e 4 , que podem ser consideradas com uma entrada de controle. Simplificando, a espessura da roda será desconsiderada e é assumido estar em contato com o plano no ponto P_i . A velocidade lateral v_{iy} , diferente de outros tipos de veículos com rodas, geralmente não é nula. Essa característica vem da estrutura mecânica do robô com direção deslizante necessárias para as mudanças de orientação. Assim, as rodas são tangentes ao percurso somente se $w = 0$, isto é, quando o robô se movimenta ao longo de uma linha reta (COSTA, 2019). É desconsiderado o escorregamento longitudinal entre as rodas e a superfície, desta forma, vale a seguinte relação em (2.3),

$$v_{ix} = r_i w_i \quad (2.3)$$

sendo que v_{ix} é o componente de velocidade longitudinal do vetor de velocidade v_i da i -ésima roda representado no sistema de coordenadas local e r_i é o raio efetivo de rolagem de tal roda (COSTA, 2019).

Deve-se salientar que para desenvolver um modelo cinemático, considera-se todas as rodas juntas. Considere os vetores $d_i = [d_{ix} \ d_{iy}]^T$ e $d_C = [d_{Cx} \ d_{Cy}]^T$ determinadas em relação ao sistemas de coordenadas local com base no centro instantâneo de rotação (ICR) (COSTA, 2019). A equação (2.4) pode ser deduzida:

$$\frac{\|v_i\|}{\|d_i\|} = \frac{\|v\|}{\|d_C\|} = |w| \quad (2.4)$$

ou ainda poderia se escrever de forma mais detalhada em (2.5):

$$-\frac{v_{ix}}{d_{iy}} = \frac{v_x}{d_{Cy}} = \frac{v_{iy}}{d_{ix}} = \frac{v_y}{d_{Cx}} = w \quad (2.5)$$

No sistema de coordenadas local, as coordenadas do centro instantâneo de rotação ICR, pode ser definido por:

$$ICR = (x_{ICR}, y_{ICR}) = (-d_{Cx}, -d_{Cy}) \quad (2.6)$$

Considerando o robô como um corpo rígido, a velocidade tangencial (v_t) de qualquer ponto horizontal no corpo do robô, é designado por:

$$v_t = w \begin{bmatrix} d_{Cy} + d_y \\ -(d_{Cx} + d_x) \end{bmatrix} \quad (2.7)$$

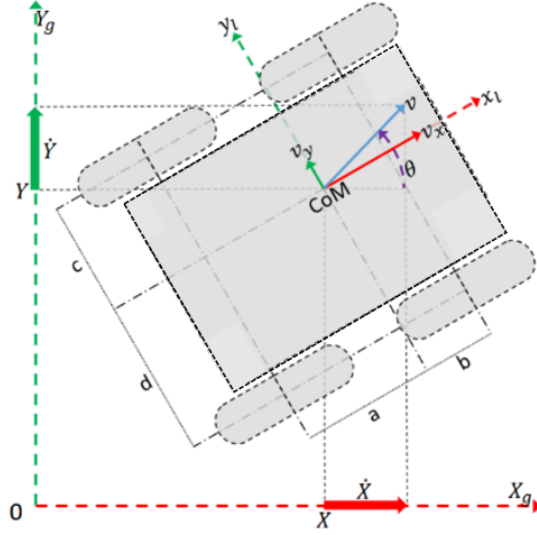
sendo que d_x é a variação da posição do ponto no eixo x e d_y é a variação da posição do ponto no eixo y (COSTA, 2019). As coordenadas do vetor d_i satisfazem as relações:

$$\begin{aligned} d_{1x} &= d_{4x} = d_{Cx} - a \\ d_{2x} &= d_{3x} = d_{Cx} + b \\ d_{1y} &= d_{2y} = d_{Cy} + c \\ d_{3y} &= d_{4y} = d_{Cy} - d \end{aligned} \quad (2.8)$$

sendo os parâmetros cinemáticos positivos do robô representados por a , b e c e d como pode ser visto na Figura 2. As velocidades podem ser obtidas em (2.9), a partir das

equações (2.5) e (2.8):

Figura 2: Representação do centro de massa do robô.



Fonte: (COSTA, 2019)

$$\begin{aligned}
 v_{xL} &= v_{1x} = v_{2x} \\
 v_{xR} &= v_{3x} = v_{4x} \\
 v_{yF} &= v_{2y} = v_{3y} \\
 v_{yB} &= v_{1y} = v_{4y}
 \end{aligned} \tag{2.9}$$

sendo as coordenadas longitudinais das velocidades das rodas direita e esquerda representadas por v_{xR} e v_{xL} , enquanto as coordenadas de velocidade laterais das velocidades da roda da frente e roda de trás são representadas por v_{yF} e v_{yB} .

A partir das equações (2.5) e (2.9), obtêm-se a transformação da equação (2.10), que define as relações entre as velocidades das rodas e a velocidade do robô (COSTA, 2019):

$$\begin{bmatrix} v_{xL} \\ v_{xR} \\ v_{yF} \\ v_{yB} \end{bmatrix} = \begin{bmatrix} 1 & -c \\ 1 & d \\ 0 & -x_{ICR} + b \\ 0 & -x_{ICR} - a \end{bmatrix} \begin{bmatrix} v_x \\ w \end{bmatrix} \tag{2.10}$$

Assumindo o raio efetivo $r_i = r$ para cada roda e relacionando a equação (2.3) e (2.9), permite-se escrever as velocidades angulares das rodas w_w através da equação:

$$w_w = \begin{bmatrix} w_L \\ w_R \end{bmatrix} = \frac{1}{r} \begin{bmatrix} v_L \\ v_R \end{bmatrix} \quad (2.11)$$

onde w_L a velocidade angular da roda esquerda e w_R a velocidade angular da roda direita. As relações entre as velocidades angulares das rodas e as velocidades do robô podem ser desenvolvidas da combinação da equação (2.10) e (2.11), ou seja, representadas por:

$$\eta = \begin{bmatrix} v_x \\ w \end{bmatrix} = \begin{bmatrix} \frac{r(w_R+w_L)}{2} \\ \frac{r(w_R-w_L)}{c+d} \end{bmatrix} \quad (2.12)$$

sendo que η pode ser uma nova entrada de comando introduzida a nível cinemático (COSTA, 2019).

A restrição de velocidade em (2.13), que teve origem em (2.5) será utilizada, a fim de completar o modelo cinemático do robô deslizante.

$$v_y + \dot{\theta}x_{ICR} = 0 \quad (2.13)$$

De (2.13) e (2.2) resulta em:

$$\begin{bmatrix} -sen\theta & cos\theta & x_{ICR} \end{bmatrix} \begin{bmatrix} \dot{X} & \dot{Y} & \dot{\theta} \end{bmatrix}^T = 0 \quad (2.14)$$

Substituindo

$$\dot{q} = \begin{bmatrix} \dot{X} & \dot{Y} & \dot{\theta} \end{bmatrix}^T$$

a (2.14) pode ser reescrita como:

$$\begin{bmatrix} -sen\theta & cos\theta & x_{ICR} \end{bmatrix} \begin{bmatrix} \dot{X} & \dot{Y} & \dot{\theta} \end{bmatrix}^T = \mathbf{A}(\mathbf{q})\dot{q} = 0 \quad (2.15)$$

sendo $A(q)$ uma matriz auxiliar para o cálculo do espaço nulo da equação (2.15). Sendo que velocidade generalizada \dot{q} esteja sempre no espaço nulo de $A(q)$, ela pode ser escrita por (2.16). Onde o operador derivada pode ser representada por (2.17) e (2.18)

$$\dot{q} = \mathbf{S}(\mathbf{q})\eta \quad (2.16)$$

$$\mathbf{S}^T(\mathbf{q})\mathbf{A}^T(\mathbf{q}) = 0 \quad (2.17)$$

$$\mathbf{S}(\mathbf{q}) = \begin{bmatrix} \cos\theta & x_{ICR}\sin\theta \\ \sin\theta & -x_{ICR}\cos\theta \\ 0 & 1 \end{bmatrix} \quad (2.18)$$

Nota-se, desde que $\dim(\eta) = 2 < \dim(q) = 3$, a (2.16) representa a cinemática do robô, que é subatuado. Este é um sistema não holonômico devido à restrição retratada na (2.13) na qual a velocidade v_y depende da derivada de q , \dot{q} , para ser calculada. A partir das equações (2.7) e (2.10) percebe-se que o controle das coordenadas de velocidade de v_y não é possível sem o conhecimento da projeção no eixo x do ICR. Considerar a velocidade linear v_x e a velocidade angular w como sinais de controle, quando comparado com outras modelagem de um robô móvel com esterçamento por escorregamento como (KOZLOWSKI; PAZDERSKI, 2004) onde w e v_y foram utilizadas (COSTA, 2019).

2.2 Sensores

2.2.1 Sensores instalados

A seguir, apresentam-se os sensores instalados nos robôs móveis deste projeto: A câmera é um objeto óptico que possui a finalidade de capturar imagens. O intuito da câmera no projeto é que o robô consiga utilizar a imagem para detectar informações como por exemplo, um possível obstáculo.

O GPS, do inglês, *global positioning sensor*, é um sistema de navegação que, através da recepção do sinal de satélites, permite saber a localização, velocidade e tempo, do objeto o qual está inserido, em qualquer local do planeta.

O acelerômetro é um sensor responsável por medir a aceleração do sistema que ele se encontra. Suas medidas são através das coordenadas cartesianas nos eixos de rotação x , y e z do robô.

O giroscópio é um sensor utilizado para indicar as mudanças de direção de um objeto em movimento e no caso deste trabalho será utilizado um destes aparatos com 3 graus de liberdade, orientados pelos sentidos dos eixos de rotação das coordenadas cartesianas x , y e z do robô.

O magnetômetro é um instrumento utilizado para medir a direção e a intensidade do campo magnético da terra. Neste trabalho, ele é utilizado para corrigir a medida dos sensores acelerômetro e giroscópio, através de vetores e matrizes de rotação para estimativa de posição do robô simulado.

Todos os sensores acima descritos são disponibilizados no *software* Webots™. Como padrão é possível definir seus parâmetros e características físicas.

2.2.2 Fusão de dados

Para estimativa em tempo real de medidas cinemáticas, o procedimento comum é o uso de um filtro de Kalman relacionando as equações dinâmicas do modelo com as medidas possíveis dos sensores. Em problemas usando sensores inerciais, a literatura apresenta diversas abordagens com sucesso relativo com precisão de alguns metros usando dados combinados com as coordenadas geoestacionárias. Porém, para medidas com precisões maiores, ainda há um desafio aberto usando fusão de sensores e mesmo há a dificuldade de se levar em conta a integração numéricas dos dados produzidos pelo giroscópio.

De posse desses sinais, podem-se usar as relações dinâmicas a que estão sujeitos pelo fato de serem produzidos por sensores presos ao mesmo corpo rígido. Os sensores disponíveis produzem leituras de variáveis dinâmicas em relação ao aparelho que esta localizado, e nisso reside sua principal dificuldade. O acelerômetro mede aceleração em relação ao um sistema de coordenadas preso ao aparelho e portanto que está sempre mudando em relação ao referencial inercial do ambiente. O giroscópio mede a velocidade angular com essa mesma característica.

O magnetômetro é o único sensor que possui uma relação direta com o referencial inercial. Isso se deve ao ato de medir campo magnético. Na ausência de campos intensos próximos, avalia o campo magnético da Terra em três eixos, que é constante para as distâncias de poucos metros. Assim, as variações desse campo representam variações do sistema de coordenadas do aparelho, ou seja, rotações puras. Por meio do magnetômetro apenas pode-se obter a medida do ângulo de inclinação e do ângulo de rotação desejados.

Quando se avalia a partir de uma média de alguns pontos o campo magnético medido,

tem-se um vetor inicial m_0 . O problema da álgebra de vetores de encontrar a rotação que converte um vetor em outro que tenham mesmo módulo pode ser aplicado (LANDI; ZAMPINI, 2018). É útil dividir o módulo do vetor do campo magnético para maior estabilidade numérica, uma vez que apenas sua direção é necessária. A direção perpendicular entre os dois instantes do campo magnético pode ser obtida pelo produto vetorial entre ambos. A diferença entre o campo em um instante e o campo do instante inicial quando obtida por uma rotação corresponde à rotação oposta à do aparelho. Esse fato é consequência da constância do campo magnético da Terra.

Ao reescrever a rotação que leva um vetor $m(t)$ a um m_0 como a matriz $R(\theta)$, que é função de três ângulos de Euler $\theta = [\theta, \rho, \psi]$.

$$m(t) = R(\theta)m_0.$$

O problema inverso obtido pelo produto vetorial denotado por Rot

$$\theta = \text{Rot}(m_0, m(t)). \quad (2.19)$$

Por outro lado, a rotação é também medida indiretamente pelo giroscópio, a menos do conhecimento do seu sistema de coordenadas em relação à referência inercial. Essa rotação é também, como consequência do funcionamento do magnetômetro, a taxa de rotação instantânea da direção do campo magnético em sentido oposto (o corpo gira em uma direção, o sensor percebe o campo girando ao contrário). Assim, é possível estimar a velocidade angular nas coordenadas inerciais ($\Omega(t)$) de duas formas. A primeira delas é a taxa de variação instantânea da rotação do campo magnético (compensando por seu módulo)

$$\Omega_1(t) = \frac{1}{\|m(t)\|} \frac{dm(t)}{dt} \quad (2.20)$$

e também pelo próprio giroscópio, que medida em sistema de coordenadas local ($\omega(t)$)

$$\Omega_2(t) = R(\theta)\omega(t) \quad (2.21)$$

que corresponde a uma outra medida, e portanto, outra fonte de informação a se considerar na fusão de dados.

Uma fusão de sensores numericamente estável deve fazer a diferença acumulada ao longo do tempo, ou em uma janela dele, de

$$e(T) = \left\| \int_0^T \frac{1}{\|m(t)\|} \frac{dm(t)}{dt} - R(\theta)\omega(t) dt \right\| < \epsilon \quad (2.22)$$

estar limitada a um valor máximo de erro (ϵ).

O cálculo de $\theta(t)$ pode ser feito usando uma combinação das duas formas apresentadas (2.20) e (2.21). Para levar em conta a dinâmica do problema, é útil observar que a expressão para o erro (2.22) é convexa quando se consideram os termos de velocidade. Ou seja, espera-se uma variação do erro que cresce conforme se afasta da relação correta para as variáveis obtidas.

As variações esperadas entre as medidas devem corresponder ao ruídos que podem se originar das diferenças entre os tempos de amostragem dos sensores mas também às dinâmicas eletrônicas de seus transdutores. Tais equipamentos podem apresentar histerese e atratores dinâmicos. Espera-se que tais efeitos sejam pouco relevantes nos intervalos de tempo considerados. Nesse sentido, propõe-se um tempo de observador não causal que estime a posição angular do aparelho usando todas as medidas disponíveis e colocadas representando uma relação que se aproxima da ideal de forma convexa. A forma convexa para o erro significa supor que suas variações quando combinadas oscilarão em torno do valor esperado. Ou seja, neste caso, um estimador (h) para a inclinação tridimensional $\Theta(t)$ usando $\theta(t)$, $\Omega_1(t)$ e $\Omega_2(t)$.

$$\Theta(t) = h(\theta(t), \Omega_1(t), \Omega_2(t)) \quad (2.23)$$

lembrando que tal estimador pode usar valores de medidas em um horizonte de eventos antes e após t .

Uma abordagem direta sobre o problema seria tratá-lo como minimização do erro $e(T)$ em que as variáveis de decisão sejam os ângulos θ . Como se trata de uma função, isso levaria na prática a encontrar um conjunto de pontos para uma malha de tempo. Produz-se assim o triplo (três eixos) do número de pontos do tempo como variáveis para o problema de minimizar o erro. Essa quantidade de variáveis torna o problema intratável mesmo para um horizonte de tempo T curto.

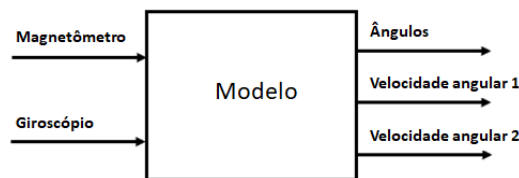
Uma rede neural treinada para contemplar as variações inerentes ao sistema de medida pode se aproveitar das características discutidas na seção anterior usando um intervalo de tempo em que se tomam as entradas dos sensores, que são por sua vez processadas para obter as três medidas vetoriais indicadas na equação (2.23). Ao fornecer como saída os três valores que estimam a posição angular em um instante de tempo, consegue-se aproveitar o fato de que o modelo das relações entre os sensores já foi contemplado fora da rede neural, bem como estabelecer um continuidade entre os pontos de dados, já que se usam pontos anteriores e posteriores como alimentação da rede neural para produzir

os valores de um instante de saída.

Porém, as medidas encontradas pelas equações (2.19), (2.20) e (2.21), que são obtidas com os sensores de forma independente representam a cinemática completa que se deseja medir. Supondo que os erros dinâmicos dos sensores sejam de natureza diferentes e independentes, o ruído será reduzido se ao longo do tempo se adotar a continuidade dos sinais e que essas variáveis medidas mantêm as mesmas relações. A relação de velocidade angular é de grande importância e deve-se evitar o erro comum de relacionar a posição angular com a integração direta da velocidade angular. Essas grandezas não são relacionadas devido às relações de influência que um eixo tem sobre o outro neste caso e a não comutatividade de rotações em três dimensões. Essa questão é tratada com detalhes em (BOYLE, 2017).

Usando um intervalo de pontos das três medidas citadas, a cinemática que se deseja modelar está incluídas nessas equações e as relações entre elas dadas pelo sensores estão incluídas quando o autoencoder é treinado para reproduzir as mesmas relações para várias entradas dos mesmos tipos. A Figura 3 representa a função das equações do modelo apresentado e calcula uma estimativa para os ângulos de rotação e duas estimativas independentes para a velocidade angular.

Figura 3: Modelo que calcula uma estimativa para os ângulos de rotação e duas estimativas independentes para a velocidade angular.



Fonte: Autora.

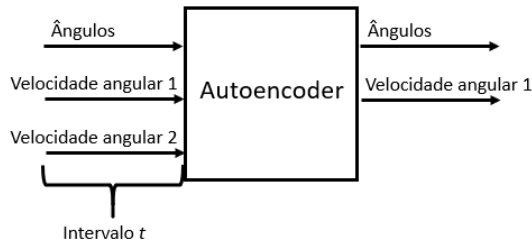
A sequência de passos para essa solução pode ser resumida como segue:

- Treinar uma rede neural do tipo autoencoder usando dados obtidos de um modelo.
- Usar a rede treinada sobre os mesmos dados para obter uma relação melhorada dos valores de rotação e velocidade angular.

Representado pela Figura 4, o autoencoder tenta reproduzir uma saída usando um intervalo de tempo T dos dados selecionados pelo modelo. A estimativa da velocidade angular de saída é feita pela média das duas.

$$\Omega(t) = \Omega_1(t) + \Omega_2(t).$$

Figura 4: Representação do autoencoder e a estimativa da velocidade angular de saída é feita pela média das duas. $\Omega(t) = \Omega_1(t) + \Omega_2(t)$.



Fonte: Autora.

Para treinar a rede, os passos são os seguintes:

- 1 Calcular a estimativa de ângulo de rotação como em (2.19).
- 2 Estimar a velocidade angular como em (2.20).
- 3 Estimar a velocidade angular como em (2.21).
- 4 Escolher uma janela de tempo T para tratar os sinais. Deve contemplar a dinâmica dos sinais. Escolher também o número de unidades na camada intermediária do autoencoder.
- 5 Empilhar as medidas dos itens 1, 2 e 3 em blocos de tamanho da janela.
- 6 Variar um ponto a cada janela, a sobreposição garante a continuidade dos dados.
- 7 Aplicar o treinamento do autoencoder para os dados empilhados. Como o autoencoder deve produzir duas saídas próximas, toma-se a média das duas.

3 GERAÇÃO E CONTROLE DE TRAJETÓRIA

3.1 Controle de trajetória

3.1.1 Método de geração de trajetória por otimização

Em (SERRALHEIRO, 2018), considera-se um Controle Hierarquico Aninhado (NHC) para definir o sistema de controle responsável pelo movimento do robô. Para definir a arquitetura NHC, existe abstração em níveis hierárquicos distintos:

- 1 *Planejamento de missão*: tarefa que o robô deve cumprir, objetivo específico que geralmente são os pontos de controle que o robô deve percorrer para uma determinada tarefa.
- 2 *Planejamento de caminho*: etapa em que o robô encontra o caminho mais objetivo possível a partir de um algoritmo seguindo a regra de otimização definida.
- 3 *Planejamento de trajetória*: considerando o caminho gerado na etapa anterior, nesta etapa ocorre a geração de um perfil de velocidade a ser seguido pelo robô durante o percurso.
- 4 *Rastreador de trajetória*, a quarta etapa, considerada outro nível hierárquico, consiste em realizar o processo necessário para que o robô consiga desenvolver a trajetória definida pelo nível hierárquico anterior. Um esboço dos níveis hierárquicos podem ser observados na Figura 5.

A principal distinção entre caminho e trajetória é que o caminho é apenas espacial e não possui relação temporal, enquanto a trajetória apresenta esta dependência com o tempo (SERRALHEIRO, 2018). Neste trabalho, explora-se a aplicação de controle para sementeira, portanto, o caminho a ser seguido pelo robô é uma distribuição aproximadamente uniforme de pontos sobre uma área. Desta forma, a etapa de planejamento de

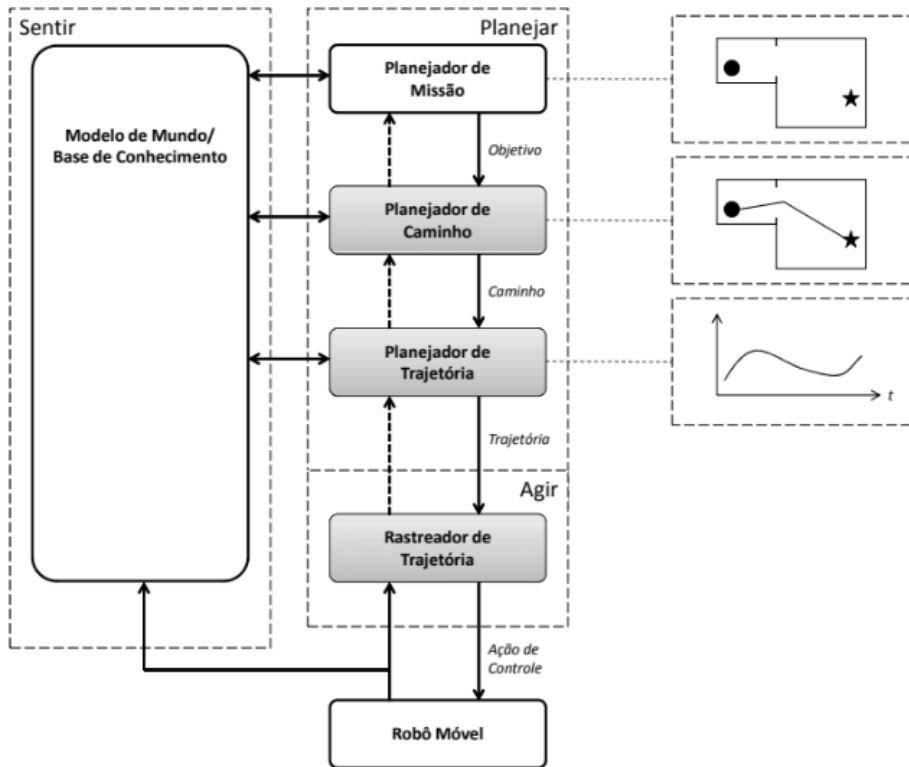


Figura 5: Arquitetura NHC
 Fonte: (SERRALHEIRO, 2018).

missão torna-se evidente. A etapa de planejamento de caminho pode ser aproximadamente resolvida através de uma sequência de semente decidida inicialmente.

As etapas de planejamento e rastreamento de trajetória acontecem em conjunto, na medida em que a sequência de pontos do caminho é rastreada um de cada vez até que o ponto de semente almejado seja alcançado. Uma vez feita a semente, que é uma tarefa estática e pontual, passa-se ao ponto seguinte da sequência como ponto almejado do rastreamento.

3.2 Controle de movimento e posição

O controle de movimento para acompanhar uma trajetória é tipicamente resolvido em robôs móveis terrestres com duas malhas de controle (SERRALHEIRO, 2018). Uma malha de controle ajusta direção e outra ajusta a velocidade. Embora um controle não linear envolvendo todas as variáveis acopladas seja possível, ainda não há resultados práticos robustos para essa abordagem, apenas simulações. Assim, a inteligência de controle fica concentrada na geração de trajetórias eficientes, e fica a cargo de um controlador acompanhar tal trajetória. A solução de um PID é robusta para as oscilações de terreno

que se pretende testar, além de possuir poucos parâmetros para ajuste e que podem ser interpretados diretamente.

3.2.1 Descrição do controlador

Abre-se espaço para que os três parâmetros de um controle PID sejam ajustados pelo algoritmo gerador de trajetória. A cada horizonte de tempo T atualiza-se a trajetória a ser seguida pelo robô e também seus parâmetros de controle. O ajuste dos parâmetros é resultado de uma comparação entre a trajetória desejada previamente e a que se estima que tenha sido seguida. Para tanto, calcula-se a diferença entre essas duas trajetórias, a diferença entre suas derivadas temporais, a diferença de erro estacionário e também o valor de energia consumido. Usando essas medidas, faz-se uma pequena variação nos parâmetros para que o controlador seja aproximadamente melhorado para o próximo horizonte de tempo.

Sejam a trajetória desejada e a trajetória estimada pelos sensores, respectivamente $r(t) = (x(t), y(t))$ e $s(t) = (\hat{x}(t), \hat{y}(t))$, e a velocidade aplicada em cada roda $\omega_k(t)$ (k sendo índice da roda), os critérios para observar o sucesso do controlador podem ser escritos:

$$e_p(T, i) = \int_0^T \|r(t) - s(t)\| dt \quad (3.1)$$

$$e_v(T, i) = \int_0^T \|\dot{r}(t) - \dot{s}(t)\| dt \quad (3.2)$$

$$E(T, i) = \int_0^T \sum_{k=1}^4 \omega_k^2(t) dt \quad (3.3)$$

com $e_p(T, i)$ sendo um erro de posição acumulado pelo horizonte de tempo T na iteração i . Analogamente para o erro acumulado de velocidade $e_v(T, i)$ e a energia consumida como proporcional a $E(T, i)$.

A estimativa instantânea dos erros da posição e da inclinação são $s(t) - r(t)$ e $\theta_r(t) - \theta_s(t)$, com $\theta_r(t) = \arctan(x(t), y(t))$ e $\theta_s(t) = \arctan(\hat{x}(t), \hat{y}(t))$. Assim, espera-se um controlador para o erro $s(t) - r(t)$ e outro para o erro $\theta_r(t) - \theta_s(t)$. Imagina-se o primeiro controlador com os parâmetros proporcional, integral e derivativo K_{pp} , K_{dp} e K_{ip} e o segundo K_{pi} , K_{di} e K_{ii} .

Uma notação matricial é útil para resumir possíveis relações entre os parâmetros de controle e os critérios que se deseja otimizar,

$$\begin{aligned}
\begin{bmatrix} K_{pp}(i) \\ K_{dp}(i) \\ K_{ip}(i) \end{bmatrix} &= \begin{bmatrix} K_{pp}(i-1) \\ K_{dp}(i-1) \\ K_{ip}(i-1) \end{bmatrix} \\
&+ A_p(i) \begin{bmatrix} e_p(T, i) \\ e_v(T, i) \\ E(T, i) \end{bmatrix} \\
&+ B_p(i) \begin{bmatrix} e_p(T, i-1) \\ e_v(T, i-1) \\ E(T, i-1) \end{bmatrix}
\end{aligned} \tag{3.4}$$

em que as matrizes 3X3 A_p e B_p representam relações que permitem ajuste dos parâmetros.

Uma equação semelhante se escreve para o controle de inclinação

$$\begin{aligned}
\begin{bmatrix} K_{pi}(i) \\ K_{di}(i) \\ K_{ii}(i) \end{bmatrix} &= \begin{bmatrix} K_{pi}(i-1) \\ K_{di}(i-1) \\ K_{ii}(i-1) \end{bmatrix} \\
&+ A_i(i) \begin{bmatrix} e_p(T, i) \\ e_v(T, i) \\ E(T, i) \end{bmatrix} \\
&+ B_i(i) \begin{bmatrix} e_p(T, i-1) \\ e_v(T, i-1) \\ E(T, i-1) \end{bmatrix}.
\end{aligned} \tag{3.5}$$

A cada passagem de um horizonte de tempo, esse valores são atualizados.

3.2.2 Atualização da trajetória conforme espaço percorrido

Para que os parâmetros A_p , B_p , A_i e B_i sejam atualizados conforme os intervalos de tempo, é preciso um critério de melhoria baseado nas funções que se deseja otimizar. A proposta aqui é de atualizar tais parâmetros por um procedimento de otimização em que a passagem dos intervalos sejam iterações de melhoria. Entre os diversos algoritmos usados para otimização, um dos usados mais recentemente com robustez em problemas não lineares é o algoritmo do Morcego (DU; SWAMY, 2016).

Para execução do algoritmo do morcego, será considerado os parâmetros das variáveis das equações 3.4 e 3.5 como variáveis de decisão e os erros de posição, erro de inclinação e gasto de energia como funções objetivo para a otimização. Nesse sentido, cada intervalo

de tempo considerado é uma iteração no sentido de obter um controlador que melhora conforme a passagem do tempo. A robustez do algoritmo o torna adaptado às variações que podem haver em um terreno real.

Conforme dito em (YANG, 2010) o algoritmo do morcego é um tipo de algoritmo metaheurístico. Estes algoritmos como otimização de enxame de partículas e têmpera simulada estão se tornando métodos poderosos para resolver muitos problemas difíceis de otimização. Muitos algoritmos heurísticos e metaheurísticos foram derivados do comportamento de sistemas biológicos ou sistemas físicos na Natureza.

O algoritmo é inspirado no processo de eco-localização destes animais, com a função de durante o seu voo detectar presas e evitar obstáculos. Conforme dito em (ANDRÉ; PARPINELLI, 2014), a ecolocalização se baseia na emissão das ondas ultrassônicas e medir o tempo resultante para estas ondas retornarem ao ponto de origem após serem refletidas pela presas. Os parâmetros variáveis a serem estudados são: o tamanho da população (n), o fator de decaimento da amplitude (α) e o fator de incremento da emissão de pulso (γ).

O algoritmo do morcego, explicado em (TSAI et al., 2012), é estruturado a partir de três principais características: comportamento de ecolocalização, frequência e volume. Os morcegos se deslocam em determinada velocidade e posição, o que resulta em uma frequência mínima f_{min} , quando se varia o comprimento de onda λ e o volume A_0 com o intuito de capturar suas presas.

Existem várias maneiras de ajustar o volume. Por uma questão de simplicidade, presume-se que o volume varia de um A_0 grande positivo a um valor mínimo constante, que é indicado por A_{min} que pode ser simulado pela equações (3.6) a (3.8):

$$f_i = f_{min} + (f_{max} - f_{min})\beta \quad (3.6)$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_{best})f_i \quad (3.7)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (3.8)$$

onde f é a frequência usada pelo morcego para encontrar sua presa, e os sufixos min e max representam o valor mínimo e máximo. x_i representa a localização do morcego no espaço da solução, v_i , a velocidade do morcego, t indica a iteração atual, β é um vetor aleatório, desenhado a partir de uma distribuição uniforme, e $\epsilon = [0, 1]$ e x_{best} indica a melhor solução global encontrada em toda a população de morcegos.

A taxa de emissão de pulsos do morcego também é considerada um dos papéis no processo e indicada pelo símbolo r_i no intervalo $\epsilon = [0, 1]$, onde o sufixo indica o i -ésimo morcego. Em cada iteração, um número aleatório é gerado e comparado com r_i . Caso ele seja maior que r_i , uma estratégia de pesquisa local, ou seja, passeio aleatório, será realizado. Uma nova solução para o morcego é gerada pela eq. (3.9):

$$x_{new} = x_{old} + \xi A_i^t \quad (3.9)$$

onde ξ é um número randômico no intervalo $[-1, 1]$, A^t representa a sonoridade média de todos os morcegos na etapa atual. Após a atualização das posições dos morcegos, a intensidade A_i e a taxa de emissão de pulso r_i também são atualizadas somente quando a melhor solução global é atualizada e o número gerado aleatoriamente é menor que A_i . A atualização de A_i e r_i é operada pela equações (3.10) e (3.11):

$$A_i^{t+1} = \alpha A_i^t \quad (3.10)$$

$$r_i^{t+1} = r_i^0 [1 - e^{-\gamma t}] \quad (3.11)$$

onde α e γ são constantes, em (YANG, 2010), para simplificar $\alpha = \gamma = 0.9$.

3.2.3 Pseudo código do Algoritmo do Morcego

O pseudo código do algoritmo do morcego está apresentado no Algoritmo 1, com $U(0, 1)$ sendo uma distribuição uniforme.

Algorithm 1 Algoritmo do Morcego

Data: Número de morcegos M , quantidade de variáveis de decisão N , número máximo de iterações T , frequência mínima f_{min} e frequência máxima f_{max} , ruído A e taxa de pulso r .

Result: Vetor g de variáveis de decisão contendo os melhores valores para a função de aptidão do problema.

Função objetivo $f(\vec{x})$, $\vec{x} = (x_1, \dots, x_N)^T$

Inicializar a população de morcegos x_i , $i = (1, 2, \dots, M)$, e suas velocidades v_i

Definir a frequência de pulso f_i para cada x_i $t \leftarrow 1$;

while $t < T$ **do**

while i de 1 até M **do**

 Gerar nova solução x_1 a partir do ajuste da frequência e atualização de velocidade e posição, conforme as equações 3.6 a 3.8 $rand \leftarrow U(0, 1)$

if $rand > r$ **then**

 Selecionar a melhor solução \vec{x}^* Gerar uma solução local x_1 em torno de melhor solução selecionada **if** $rand < A$, $f(x^*)$ **then**

 Aceitar nova solução x_1 Ordenar os morcegos e encontrar o atual melhor \vec{x}^*

end

$t \leftarrow t + 1$

end

$g \leftarrow \vec{x}^*$

end

end

A utilização do pseudo-código do algoritmo do Morcego neste trabalho, é para o ajuste dos parâmetros do controlador PID, onde são substituídos as variáveis do controlador a cada iteração no intervalo de tempo.

4 SIMULAÇÃO PARA ROBÓTICA MÓVEL

4.1 Panorama

Há várias plataformas de simulação de diversos fabricantes disponíveis para o desenvolvimento em pesquisas industriais e acadêmicas, e existe a possibilidade de interação entre essas plataformas e compartilhamento de informações (FARLEY; WANG; MARSHALL, 2022). Os *softwares* analisados para este trabalho foram Gazebo™, CoppeliaSim™ e Webots™.

As principais necessidades que o *software* deveria atender para o desenvolvimento deste trabalho:

- Tipo de sistema operacional que a plataforma opera (Windows);
- Baixa complexidade de instalação e gratuidade da plataforma;
- Precisão e qualidade de simulação oferecida;
- Integração e linguagens de programação aceitas de código aberto ;
- Ferramentas de controle de movimento em ambiente simulado.

4.1.1 Gazebo™

Gazebo™ é um simulador desenvolvido em 2002 pela *Open Source Robotics Foundation* (Open Source Robotics Foundation, 2014a) e foi o *software* pioneiro a ser utilizado na comunidade (Open Source Robotics Foundation, 2014b). A versão mais atual é a Gazebo 11 e (Open Source Robotics Foundation, 2014a), apesar de ser disponível gratuitamente, possui como sistema operacional padrão o GNU/Linux e o Linux (Ubuntu) (Open Source Robotics Foundation, 2014a), o que limita a utilização deste simulador para este trabalho, já que o sistema operacional disponível é o Windows 10. O Gazebo™ oferece suporte a vários mecanismos de física (Open Source Robotics Foundation, 2014b), que são esquemas

numéricos de simulação de sistemas dinâmicos. Apresenta uma diversidade de modelos prontos com robôs e ambientes, em torno de 240 opções até o presente momento, além de recursos de simulação com ambientes variados tais como subterrâneos, industrial e internos (Open Source Robotics Foundation, 2014b). Analisando a linguagem de programação e a interface de programa de aplicação (API), o Gazebo™ suporta apenas C++ (Open Source Robotics Foundation, 2014a), sendo outra limitação para este trabalho.

4.1.2 CoppeliaSim™

A plataforma de simulação CoppeliaSim™, anteriormente conhecida como V-REP, é desenvolvida pela Coppelia Robotics e sua versão de *software* mais atual é a V4.4.0 rev0 (Coppelia Robotics, 2023) apresentando três perfis de usuário disponíveis para download, sendo os perfis Player e Edu, gratuitos e o perfil Pró, pago. A principal diferença entre esses perfis é a total capacidade de edição não disponível para o perfil Player e a usabilidade comercial, indisponível para o perfil Edu, que é voltado para educação. Já o perfil Pró apresenta todas as funcionalidades da ferramenta (Coppelia Robotics, 2023). Há versões para o sistema operacional Linux (Ubuntu), Mac OS, e Windows. A linguagem de programação padrão é a Lua, porém suporta APIs que disponibilizam linguagens e ferramentas como C, C++, Python, Java, Urbi, Matlab, Octave (Coppelia Robotics, 2023).

4.1.3 Webots™

O Webots™ é uma plataforma de simulação de aplicações de robótica completa e bastante utilizada pois apresenta situações físicas reais inclusas bem como ferramentas para montagem e programação de arquiteturas de robôs industriais e de pesquisa, facilitando muito o cotidiano do usuário. (Cyberbotics, 2021h). Possui uma biblioteca com muitos recursos disponíveis como robôs, sensores, atuadores, materiais e um mecanismo de física compatível (ODE). Possui linguagens de código aberto, ou seja, podem ser reutilizadas sem autorização da empresa proprietária do *software* e existe a possibilidade de integração para importação de modelos prontos. É uma plataforma que apresenta baixa complexidade para instalação e gratuidade, além de operar no sistema operacional Windows.

Entre as três plataformas de simulação analisadas, a principal desvantagem do Gazebo™ é não possuir versão para o Sistema Operacional Windows. O Coppelia limita alguns recursos dependendo do perfil do usuário e para total utilização das ferramentas é necessário um perfil Pró, o que implica em custo alto. Já o Webots™, mostra que

possui apenas um mecanismo de física compatível em seu ambiente quando comparado as outras opções, porém essa característica não demonstrou ser tão limitante quanto as demais desvantagens dos outros simuladores citadas. Dadas estas informações e por atender as principais necessidades, o WebotsTM foi a opção escolhida para o desenvolvimento deste projeto. A versão do *software* utilizada neste trabalho é a R2021a (Cyberbotics, 2021h), (KOUBAA et al., 2018).

4.2 Plataforma de Simulação

Como previamente citado, *software* de simulação WebotsTM apresenta várias opções para estruturas de robôs reais com base em produtos comerciais. Também permite ao usuário construir seus próprios robôs, como realizado neste trabalho. O *software* permite a construção com muitos componentes, como sensores, atuadores e permite a calibragem e alteração dos mesmos, para ambientes reais (Cyberbotics, 2021h).

O *software* é bastante robusto, pois utiliza a especificação dos nós, APIs e campos de linguagem de código aberto, podendo ser reutilizado sem autorização da proprietária do *software*, a Cyberbotics. Além de oferecer suporte caso os usuários desejarem implementar a API em robôs reais, dando maior disponibilidade e estimulando a comunidade robótica a interoperabilidade entre os diferentes aplicativos existentes no mercado de robótica atualmente (Cyberbotics, 2021e). Ele também especifica as funções disponíveis para operar nesses nós a partir de programas do controlador.

Para utilização do WebotsTM, é essencial entender como funcionam os nós e as funções disponíveis. Os nós da plataforma são descritos utilizando a sintaxe VRML97 padrão (Cyberbotics, 2021b) e a importância deles é para a construção e utilização do robô no ambiente virtual. As funções são todas as disponíveis na API (Cyberbotics, 2021b) e apresenta linguagens universais para montagem dos projetos de programação para o controle, sendo elas C, C++, Java, Python e Matlab. O desenvolvimento deste trabalho foi realizado na linguagem Python.

Como já mencionado, a montagem do modelo no *software* é baseada em uma árvore de componentes que servem como representações para os componentes físicos e conexões cinemáticas (Cyberbotics, 2021a).

4.2.1 Gráfico de Nós

Para construção do robô, foi necessário entender como funciona o gráfico de nós, também conhecido como árvore de componentes, dentro da plataforma do *software* Webots™ para construção do ambiente. Ela é dividida nos seguintes conceitos: Relacionamento de Herança, nós abstratos, objetos limitadores, regras de inserção (Cyberbotics, 2021c) e nós sólidos.

O relacionamento de herança autoexplicativo, pois de um nó (pai) será derivado um segundo nó (filho), que herdará todos os campos e funções API do seu nó pai. Em diagrama, a forma mais simples de representar é em um esquema de setas como na Figura 6. Exemplificando o nó *Solid* herda do nó *Transform* todos os campos e funções disponíveis (Cyberbotics, 2021c):

Figura 6: Relacionamento de Herança entre um nó e um nó derivado.



Fonte: Autora.

Os nós abstratos são utilizados para agrupar campos e funções que são compartilhadas por nós derivados. A representação gráfica é através de caixas e linhas tracejadas (Cyberbotics, 2021c).

Um *Solid* (nó Sólido) é a representação de um objeto com propriedades físicas, tais como dimensões, um material de contato e massa (item opcional para simulações). Robôs e classes de *Devices* (dispositivos) são subclasses da classe *Solid*, que foi utilizado neste trabalho (Cyberbotics, 2021c). Para identificar facilmente se é um nó *Solid*, basta observar se o objeto pode ser manipulado, como por exemplo, arrastado, girado, empurrado, etc...

Os objetos limitadores (*boundingObjects*) indicam nós que detectam colisões entre objetos sólidos e geralmente são representados graficamente por caixas verdes (Cyberbotics, 2021c). Este tipo de nó também pode ser construído utilizando os nós *Group* e *Transform*.

4.2.2 Tipos de Nós

4.2.2.1 Nós VRML97

Conforme especificação no site do Webots™(Cyberbotics, 2021d), a plataforma implementa apenas um subconjunto de nós e campos que são especificados pelo padrão VRML97. As características exatas do VRML97 estão sujeitas a um padrão gerenciado pela *International Standards Organization* (ISO / IEC 14772-1: 1997) (Cyberbotics, 2021d).

Os seguintes nós VRML97 são suportados por Webots™: *Appearance, Background, Box, Color, Cone, Coordinate, Cylinder, DirectionalLight, ElevationGrid, Fog, Group, ImageTexture, IndexedFaceSet, IndexedLineSet, Material, Normal, PointLight, PointSet, Shape, Sphere, SpotLight, TextureCoordinate, TextureTransform, Transform, Viewpoint* e *WorldInfo*. Alguns dos tipos de nós VRML97 utilizados neste trabalho: *WorldInfo, Box, Shape, Transform, etc.*

4.2.2.2 Nós específicos do Webots

A plataforma Webots™também adiciona muitos nós, que não fazem parte do padrão VRML97, mas são especializados para modelar experimentos robóticos. A vantagem desses nós específicos do próprio Webots™é que permite descrever simulações robóticas com mais precisão. Esses nós são usados principalmente para modelar dispositivos de robô comumente usados (Cyberbotics, 2021d). A seguir, apresentam-se os nós adicionais do Webots™:

Accelerometer, Altimeter, BallJoint, BallJointParameters, Billboard, Brake, Camera, Capsule, Charger, Compass, Connector, ContactProperties, Damping, Display, DistanceSensor, Emitter, Fluid, Focus, GPS, Gyro, HingeJoint, HingeJointParameters, Hinge2Joint, ImmersionProperties, InertialUnit, JointParameters, LED, Lens, LensFlare, Lidar, LightSensor, LinearMotor, Mesh, Muscle, PBRAppearance, Pen, Physics, Plane, PositionSensor, Propeller, Radar, RangeFinder, Receiver, Recognition, Robot, RotationalMotor, Skin, SliderJoint, Slot, Solid, SolidReference, Speaker, Supervisor, TouchSensor, Track, TrackWheel e *Zoom*.

Alguns tipos de nós específicos do Webots™utilizados neste trabalho: *Accelerometer, Camera, Compass, GPS, Gyro, HingeJoint, HingeJointParameter*.

4.2.3 *World e Scene Tree*

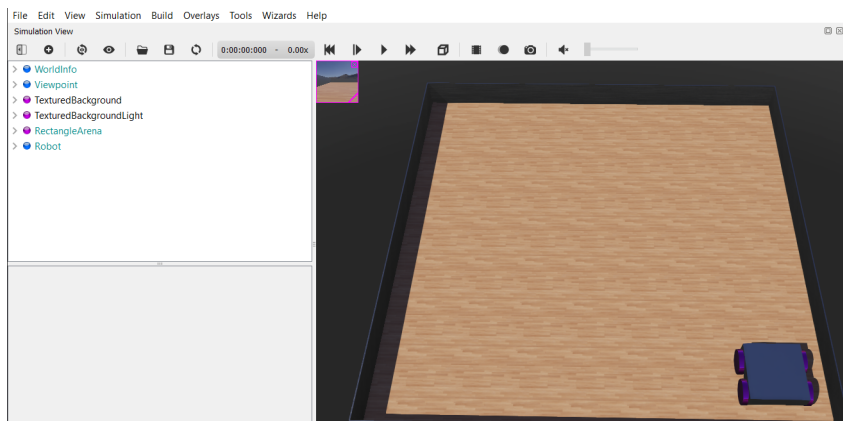
Mundo ou ambiente é um arquivo com as características iniciais da simulação, tais como a gravidade, força de atrito, massa dos objetos, ou seja, as informações do ambiente da simulação. Neste arquivo haverá informações de como os nós interagem entre si e em relação ao ambiente. No mundo, os nós geralmente são chamados de objetos e, são organizados hierarquicamente em uma *Scene Tree*. Um mundo é armazenado em um arquivo com a extensão *.wbt*. (Cyberbotics, 2021f).

A organização e armazenamento dos nós do Webots™ fica localizada em uma estrutura de árvore de cena chamada de *Scene Tree*. A *Scene Tree*, segundo (Cyberbotics, 2021f) pode ser visualizada em duas subjanelas da janela principal: a visualização 3D (no centro da janela principal) é a representação 3D da árvore de cena e a visualização em árvore da cena (à esquerda) é a representação hierárquica da árvore de cena. A visualização *Scene Tree* é onde os nós e campos podem ser modificados e personalizados, o que será feito neste trabalho.

Outros parâmetros importantes quando um mundo é criado que auxiliam na configuração inicial segundo o site (Cyberbotics, 2021f) são listados abaixo e podem ser observados na Figura 7:

- *WorldInfo*: contém parâmetros globais da simulação;
- *Viewpoint*: define os principais parâmetros da câmera do ponto de vista;
- *TexturedBackground*: define o fundo da cena;
- *TexturedBackgroundLight*: define a luz associada ao plano de fundo acima.

Figura 7: Representação da *Scene Tree* e Visualização 3D.



Fonte: Autora.

4.3 Ambiente

O primeiro passo para realizar o projeto no *software* é criar um mundo no Webots™, que é o ambiente onde serão realizadas as simulações. A primeira etapa é adicionar um terreno.

Os tipos de terrenos utilizados, entre as opções existentes no Webots™ para a simulação foram do tipo *Rectangle Arena* e posteriormente o *Uneven Terrain*. Importante salientar que o terreno também é um objeto (nó) e para a implementação no Webots™ do robô, é importante considerar os sistemas de coordenadas no mundo e preso ao corpo do robô, e no momento da construção do mesmo, atentar se a orientação das rodas esta no mesmo sentido do corpo do robô. No caso da simulação com o *Rectangle Arena*, o ponto de referência no eixo de coordenadas global x,y,z no mundo foi de [0,0,0], enquanto no caso da simulação com os terrenos de *Uneven Terrain* foram de [0,-1.3,0] pois este ambiente considera uma altura para alterar a topografia.

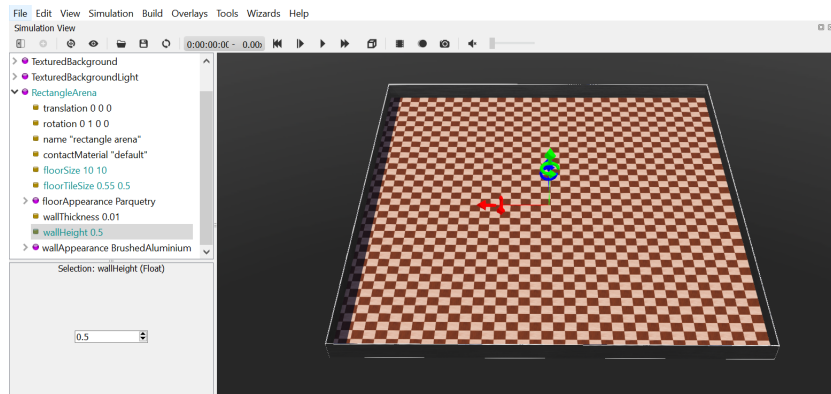
4.3.1 Terreno regular - *Rectangle Arena*

O *Rectangle Arena* é um terreno retangular configurável (Cyberbotics, 2021g) e foi o primeiro ambiente utilizado na simulação pois apresentava uma configuração menos complexa para um primeiro momento.

Entre as opções de campo personalizáveis, o usuário pode definir a dimensão do mesmo através do campo *floorSize*. Neste caso, para testes iniciais de construção, o tamanho definido foi sofrendo variações, como 4x4 m, 10x10 m, 20x20 m, 40x40 m, etc. Nessa opção de terreno, também é possível criar bordas limitantes, ou paredes, através do campo *floorAppearanceParquetry*, item *wallHeigh*. Optou-se por deixar uma borda de 0.5 m de altura como perímetro do terreno. A configuração inicial neste ambiente pode ser observada na Figura 8.

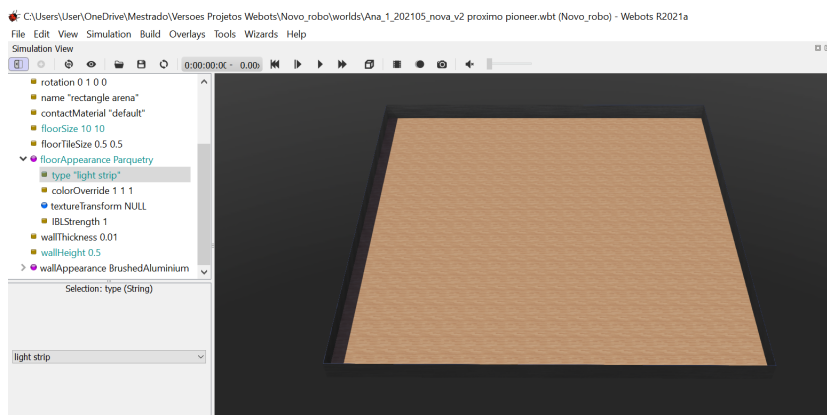
Outros campos disponíveis para configurar o terreno são: o tipo de material que é composto, aparência do solo, espessura, cores. Por padrão, o *Rectangle Arena* vem com uma superfície quadriculada, similar a um tabuleiro de xadrez. Para se aproximar mais da situação proposta de um terreno para semeadura, foi configurado, ainda através da opção *floorAppearanceParquetry*, no item tipo, a opção *light strip* conforme observado na Figura 9.

Figura 8: Configuração Inicial do ambiente *Rectangle Arena*.



Fonte: Autora.

Figura 9: Configuração do ambiente *Rectangle Arena* para semeadura.



Fonte: Autora.

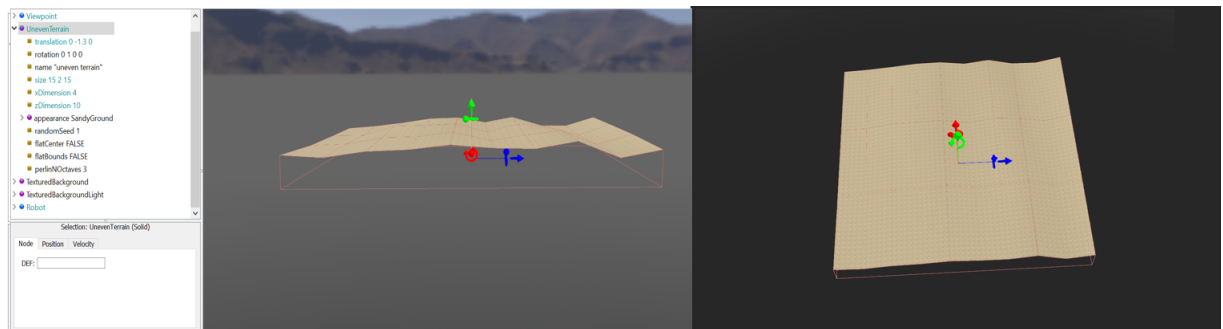
4.3.2 Terreno irregular - *Uneven Terrain*

O *Uneven Terrain* é outra opção de terreno disponível no *software* Webots™, que representa um terreno irregular e é gerado aleatoriamente com base no ruído Perlin (Cyberbotics, 2021g). Ruído Perlin é um tipo de ruído gradiente obtido através de algoritmo de geração processual para obter aparências e texturas mais realistas em terrenos bi ou tridimensionais (GEORGAKOPOULOS-SOARES; PAPAZOGLU; MARKOPOULOS, 2023), (ANDRADE, 2021). Este ambiente é mais próximo do real para a situação proposta de semeadura pois os terrenos de plantio podem apresentar irregularidades no solo e diferentes altitudes em sua topografia.

Assim como o *Rectangle Arena*, o *Uneven Terrain* também teve alguns campos configurados manualmente. Os tamanhos definidos foram variando conforme o trabalho foi evoluindo, com valores como 15x2x15 m e para as simulações finais, 40x2x40 m nas coordenadas x, y e z. Para representar as irregularidades do terreno, foram utilizados os campos *xDimension*, o qual define o número de pontos na matriz de altura na direção x

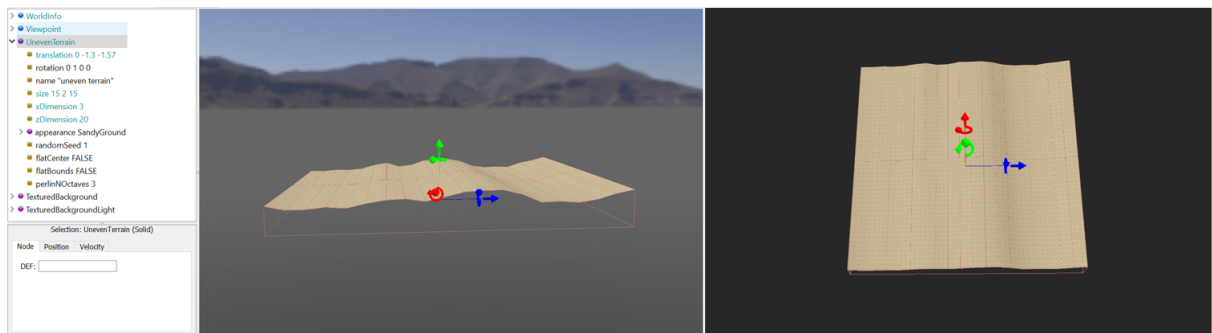
(Cyberbotics, 2021g) e $yDimension$ que define o número de pontos na matriz de altura na direção y (Cyberbotics, 2021g). A aparência do solo foi configurada como montanhosa na opção *mountain* do campo *Texturebackground*. O campo *perlinNoctaves* é mantido com o valor padrão 3 definindo o número de oitavas do ruído gradiente de Perlin (Cyberbotics, 2021g). Para as simulações, as opções de $xDimension$ e $yDimension$ foram de 4 e 10 e posteriormente de 3 e 20 para refletir em terrenos irregulares.

Figura 10: Representação lateral e frontal do Terreno irregular com $xDimension = 4$ e $yDimension = 10$.



Fonte: Autora.

Figura 11: Representação lateral e frontal do Terreno irregular com $xDimension = 3$ e $yDimension = 20$.



Fonte: Autora.

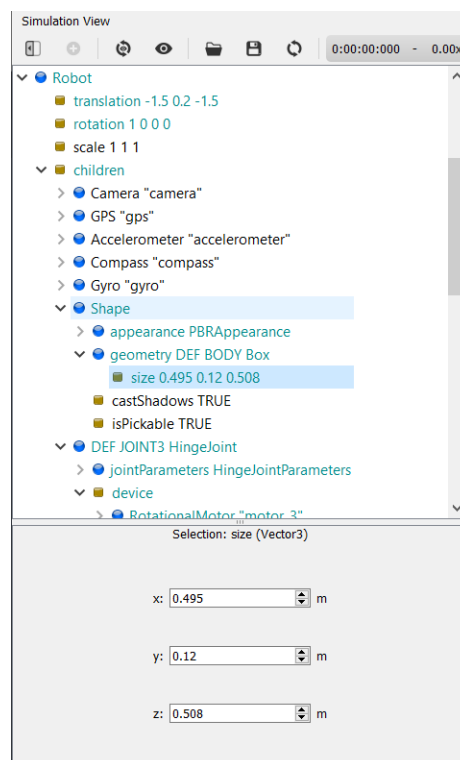
4.4 Robô

Alguns pré-requisitos foram definidos para os robôs neste trabalho: 1) ser um robô de quatro rodas, com controle individual das mesmas e 2) possuir os sensores: acelerômetro, giroscópio, magnetômetro, GPS e câmera. Fazendo uma pesquisa dos robôs existentes no mercado similares ao deste trabalho e que atendessem os pré-requisitos, foi selecionado o Pioneer3AT™ da fabricante Adept™, pois é muito versátil e permite adições de vários componentes necessários.

Após adicionar um terreno ao ambiente da simulação, o próximo passo é adicionar um nó *Robot* (robô), que é uma subclasse do nó *Solid*. A principal característica do nó *Robot*, é ser o nó raiz de uma estrutura hierárquica de nós sólidos e que podem ser ligados entre si. Foram necessários cinco nós neste trabalho: um nó do tipo *Solid* correspondente ao corpo do robô e que é o nó raiz, e outros quatro nós *Joint*, que derivam do nó *Solid*, definidos como cilindros, que representam cada uma das quatro rodas.

O primeiro nó *Solid*, representando o corpo do robô, é chamado de *Shape*, após adicionar este nó na *Scene Tree*, é necessário configurar seus nós derivados: 1) *Geometry*, responsável por definir a forma geométrica (no caso, do tipo *Box*), o tamanho do robô com as medidas 0,495m, 0,12m e 0,508m respectivamente. No caso do nó 2) *Appearance*, que é utilizado para definir características físicas do *Shape*, tais como a tonalidade, textura e propriedades físicas. 3) *Physics*, que permite especificar parâmetros para o mecanismo de simulação de Física. O nó *Physics* especifica a massa, o centro de gravidade e a distribuição de massa, permitindo assim que o mecanismo de física crie um corpo (Cyberbotics, 2022) e calcule forças simulando um ambiente real. A Figura 12 representa a *Scene Tree* do nó *Shape* e seus nós derivados *geometry* e *appearance*.

Figura 12: Representação em *Scene Tree* do nó *Shape* e seus nós derivados *geometry* e *appearance*.

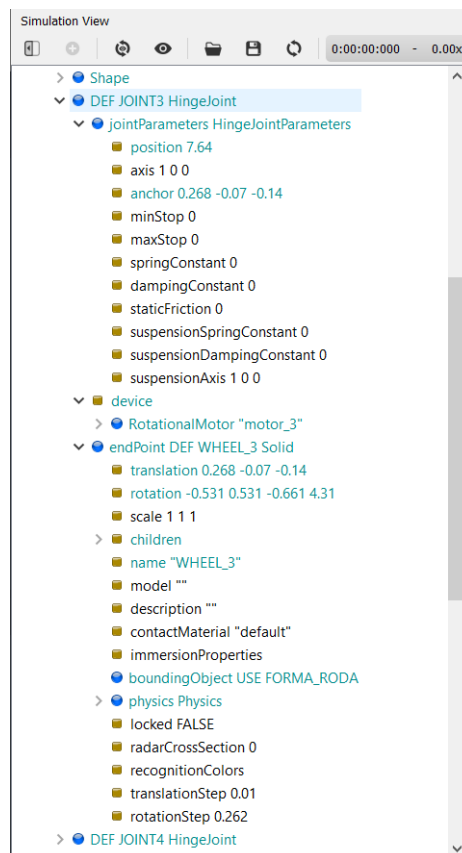


Fonte: Autora.

Para a criação de uma roda, é adicionado um nó *Joint*, que funciona como um elo para

que exista a ligação entre nós *Solid*. O *Joint* possibilita entre nó *Solid* pai e *Solid* filho a definição de graus de liberdade. A quantidade de graus de liberdade, que podem ser um ou dois, depende do tipo de nó *Joint* utilizado. Através do nó *Joint* é possível monitorar ou acionar a estrutura do robô e para isso é necessário adicionar um nó *PositionSensor* ou um nó *Motor* no campo *Device*. *Device* é um tipo de nó abstrato, que é sempre derivado de nós do tipo *Joint*, *Robot* ou *Solid*. Os nós derivados de *Joint* permitem criar diferentes tipos de restrições entre os nós *Solid* vinculados (Cyberbotics, 2021f). O *Joint* utilizado neste trabalho é o *HingeJoint*, que permite modelar, entre outros, os motores do tipo rotacional que serão utilizados para acionar as rodas. Após definir que o nó *Joint* é *HingerJoint*, existem outros três nós derivados deste: 1) o *JointParameter*, que funciona como eixo de rotação e permite configurar a posição que este nó ficará em relação ao corpo do robo; 2) nó tipo *Device*, do tipo *Rotacional Motor*, para controlar as rodas; e 3) o *endPoint*, para configurar este nó foi necessário Adicionar um nó *Solid*, depois um nó *Shape* no campo filho do *Solid* e, por último, adicionar um *Cylinder* no campo *geometry* do nó *Shape* (Cyberbotics, 2021f).

Figura 13: Representação em *Scene Tree* do nó *Joint* e todos seus nós derivados.

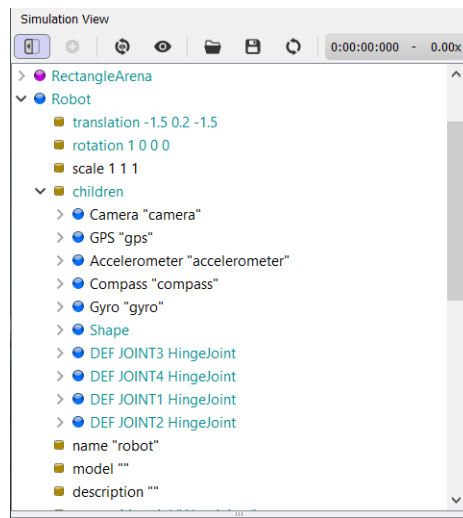


Fonte: Autora.

Após a construção da primeira roda, que começa sempre com um nó *Joint*, as três

rodas subsequentes foram replicadas da mesma forma, modificando as posições dos vetores de localização dos eixos de rotação do sistema de coordenadas dos *Rotacional Motors*. Os sensores e atuadores no Webots™ são em sua maioria construídos por nós *Solids* e *Devices* ao mesmo tempo (Cyberbotics, 2021f) como representa a Figura 13. Todos os sensores deste trabalho são nós derivados do nó *Robot*. São eles: Acelerômetro, Giroscópio, Câmera, GPS e Magnetômetro, conforme visto na Figura 14.

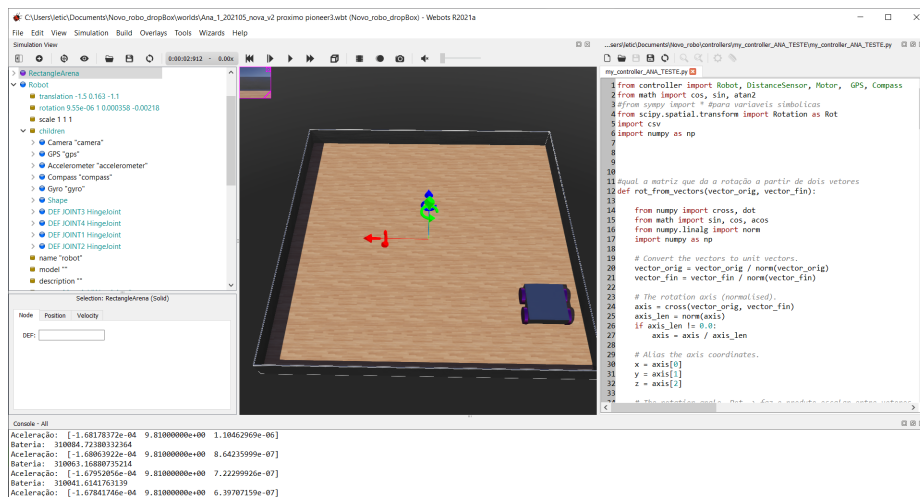
Figura 14: Representação em *Scene Tree* do nó *Robot* e todos seus nós derivados.



Fonte: Autora.

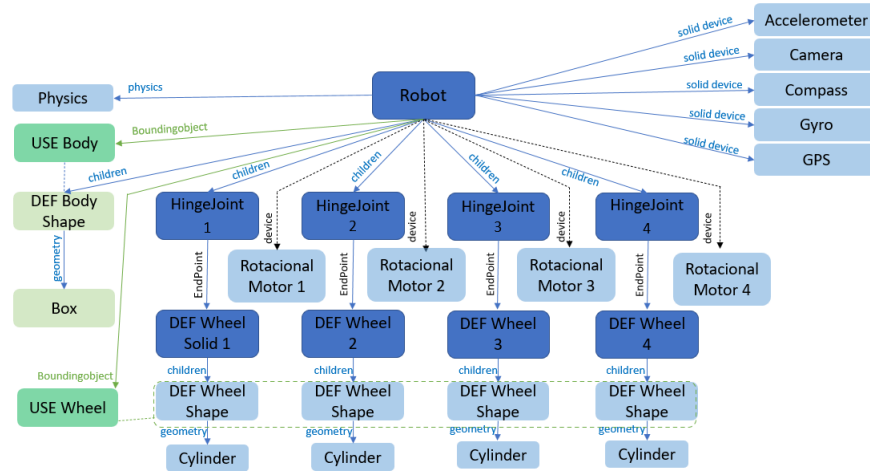
No caso dos sensores, apesar da disponibilidade de realizar alterações através da *Scene Tree*, é mais conveniente que as parametrizações e controles sejam realizadas através do editor do *software*, localizado do lado direito da tela, com comandos e funções de linguagem de programação por questão de praticidade, conforme Figura 15.

Figura 15: Representação em *Scene Tree* do nó *Robot* e todos seus nós derivados.



Fonte: Autora.

Figura 16: Conexões conceituais para montagem do robô no ambiente de simulação.

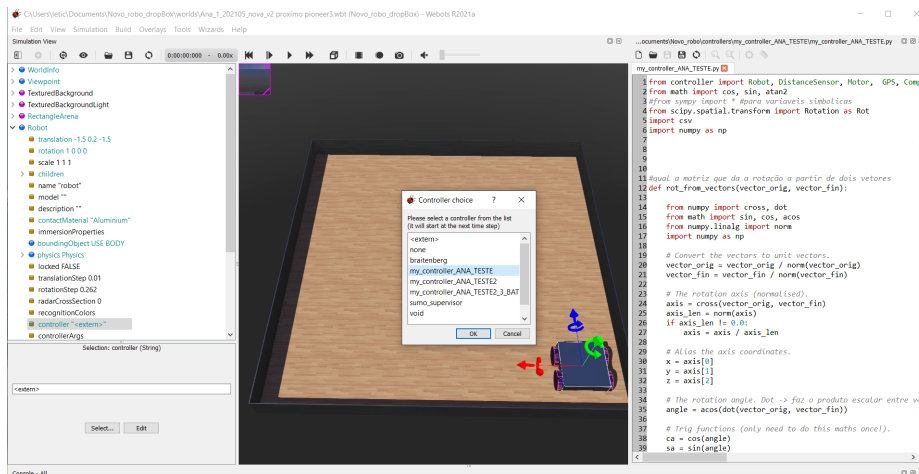


Fonte: Autora.

Um diagrama de árvore que representa o nó *Robot* e todos seus nós derivados é representado na Figura 16.

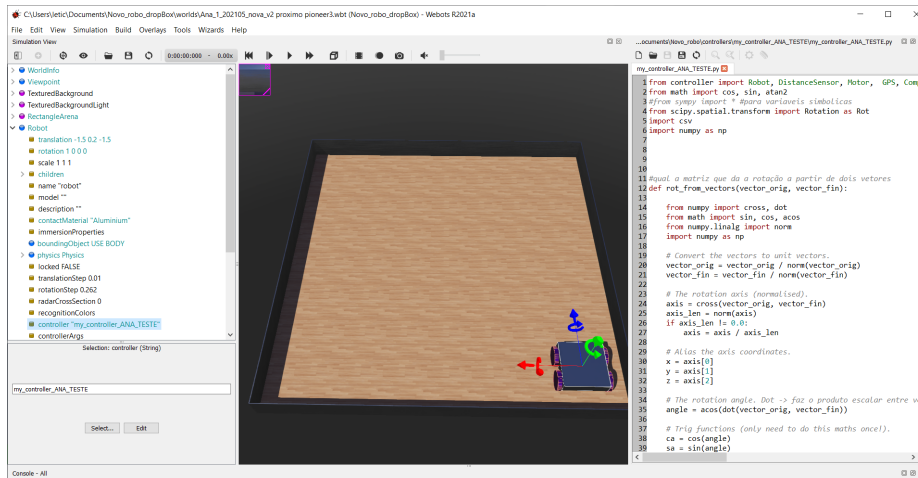
Após desenvolver a lógica de programação para o controle do robô, é necessário realizar testes empíricos do mesmo para treinar e aperfeiçoar o resultado do controle com as parametrizações. Para isso é necessário salvar o código em uma pasta do diretório da simulação do Webots™, e posteriormente seleciona-lo através da *Scene Tree* no nó *Robot*, campo *controller*, conforme as Figuras 17 e 18, após ser configurado.

Figura 17: Representação de como configurar o controle no Robô através da *Scene Tree*.



Fonte: Autora.

Figura 18: Representação do controle já selecionado no nó *Robot*.



Fonte: Autora.

4.5 Programa

A etapa inicial da simulação é inicializar os objetos da programação para que tenham acesso às variáveis físicas do ambiente simulado para leitura e controle. Após a configuração, a simulação é realizada em intervalos discretos de tempo conforme configurado. A cada instante de simulação, as etapas são seguidas como segue:

- Todos os sensores são lidos e as variáveis que eles produzem são armazenadas.
- As variáveis são usadas em um processo de observações para estimar variáveis de posição dinâmica, conforme descrito no capítulo 3.
- Os valores de posição estimados são comparados com a próxima coordenada de trajetória desejada. Se a proximidade com esse ponto foi atingida dentro de uma tolerância, atualiza-se o ponto desejada como o próximo.
- Com as variáveis de posição estimadas, calculam-se os sinais dos controladores e esses são aplicados nos atuadores.
- Reinicia-se o ciclo.

5 RESULTADOS

Os resultados foram gerados através de simulações no Webots™ e o programa que inclui fusão de sensores, controle e atualização do controlador foram escritos em linguagem python.

5.1 Condições do problema

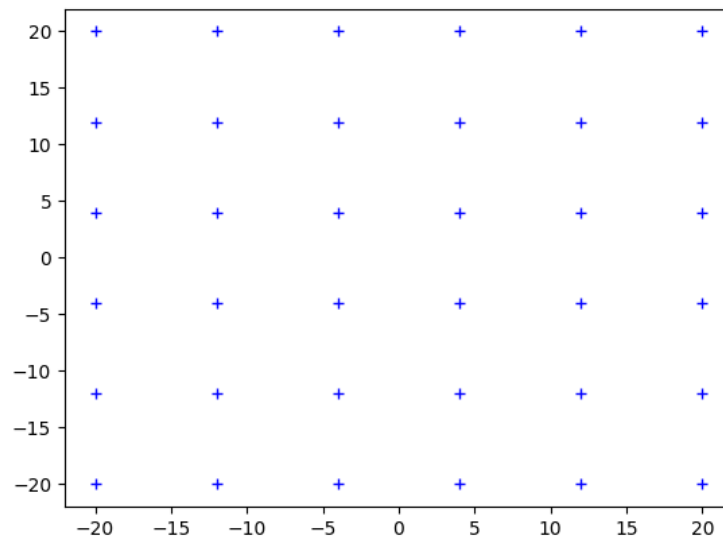
A proposta da tarefa consiste em passar por todos os pontos dentro de uma margem de precisão de localização, o que pode simular uma tarefa de semeadura. O problema foi dividido em dois ambientes de simulação distintos e a cada nova simulação, aumentava-se o nível de dificuldade de modo a se aproximar cada vez mais de situações reais enfrentadas por pequenos agricultores.

- *Situação 1*: Simulação realizada com um robô em terreno regular;
- *Situação 2*: Simulação realizada com dois robôs em terreno regular;
- *Situação 3*: Simulação realizada com dois robôs em terreno irregular;
- *Situação 4*: Simulação realizada com dois robôs em terreno irregular e com um obstáculo.

5.2 Terreno Regular

Nas duas primeiras situações, onde o terreno é regular, foram escolhidos pontos, distribuídos de maneira uniforme dentro de um espaço retangular, onde o robô deveria passar, dentro de uma certa margem de tolerância, conforme apresentado na Figura 19.

Figura 19: Pontos da trajetória a ser seguida em terreno regular.

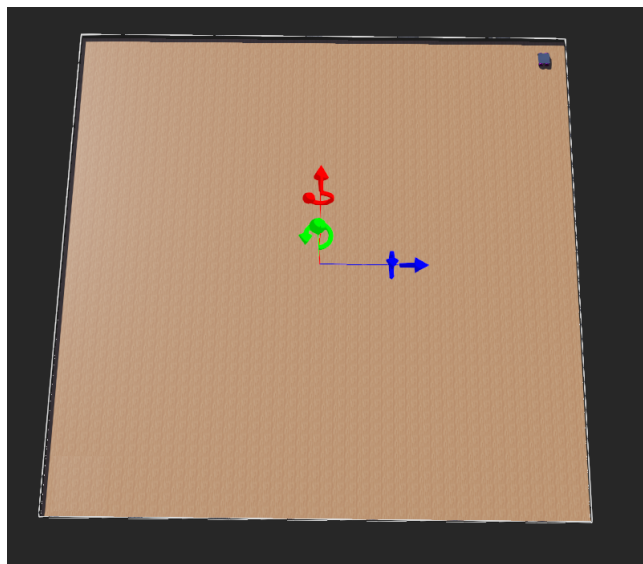


Fonte: Autora

5.2.1 Situação 1 - Terreno regular com um robô

A Figura 20 representa o ambiente da Situação 1 com um robô.

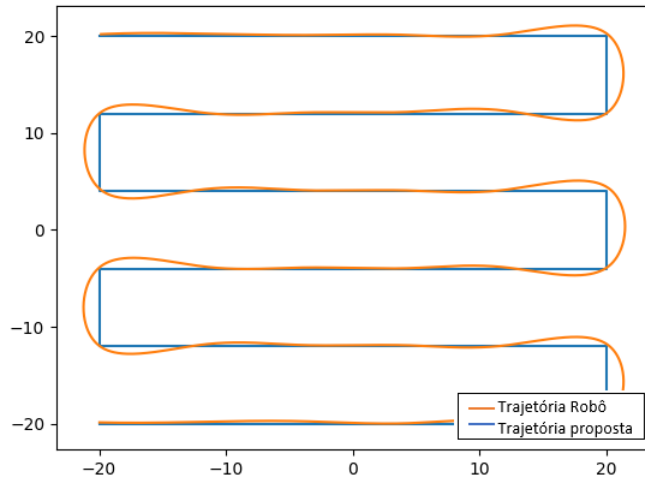
Figura 20: Representação do ambiente em terreno regular com um robô.



Fonte: Autora

Na Situação 1, pode-se observar o resultado da trajetória para um robô na Figura 21.

Figura 21: Trajetória de referência e trajetória real executada pelo robô em terreno regular.



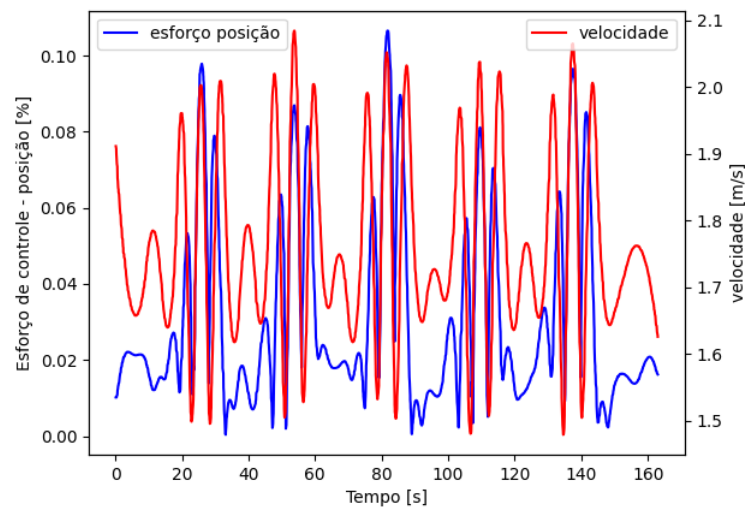
Fonte: Autora

5.2.1.1 Controle e trajetória

Durante as simulações, o esforço de controle aplicado (atuação nos motores) é o percentual da potência dos motores, sendo assim adimensional. Consta-se desta forma, um esforço de controle para a posição do Robô e um esforço de controle para a inclinação do ângulo θ .

Para as simulações da Situação 1, observa-se o esforço de controle para a posição em relação ao comportamento do perfil velocidades dos motores na Figura 22.

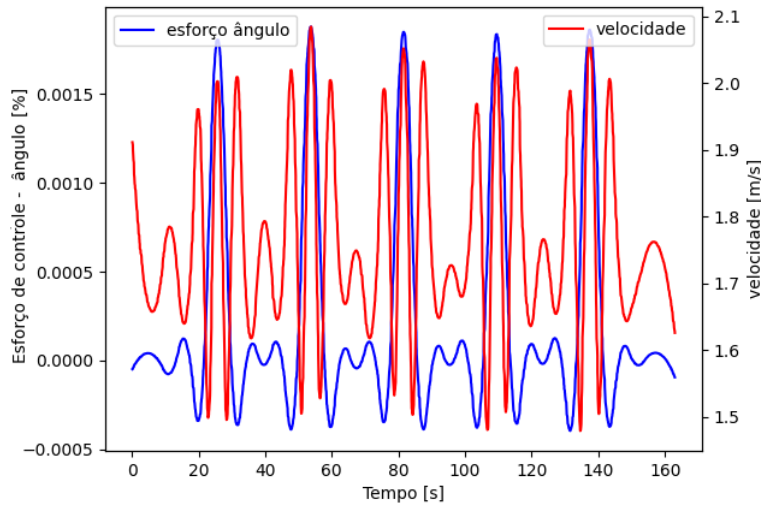
Figura 22: Esforço de controle de posição para um robô em terreno regular quando comparado com o perfil velocidade.



Fonte: Autora

A Figura 23 representa o esforço de controle de inclinação do ângulo θ em relação ao comportamento do perfil velocidade dos motores do robô.

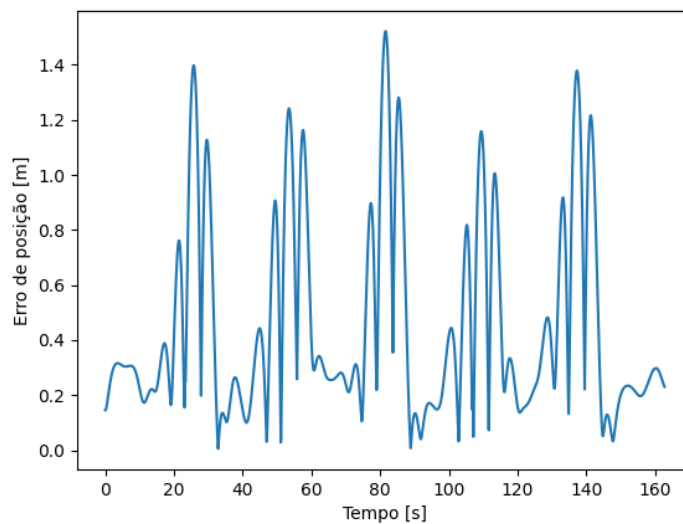
Figura 23: Esforço de controle do ângulo θ para um robô em terreno regular quando comparado com o perfil velocidade.



Fonte: Autora

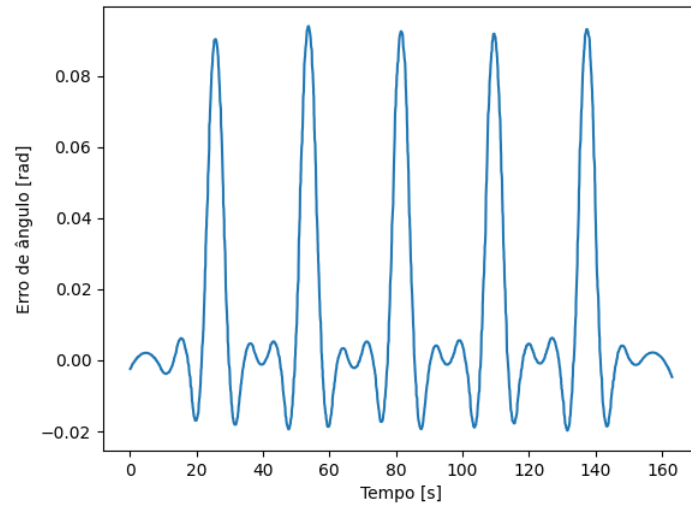
Os erros de posição e de ângulos, que são as entradas dos controladores, respectivamente no caso da Situação 1 são apresentados nas Figuras 24 e 25.

Figura 24: Erro de posição na trajetória de um robô em terreno regular.



Fonte: Autora

Figura 25: Erro do ângulo θ na trajetória de um robô em terreno regular.

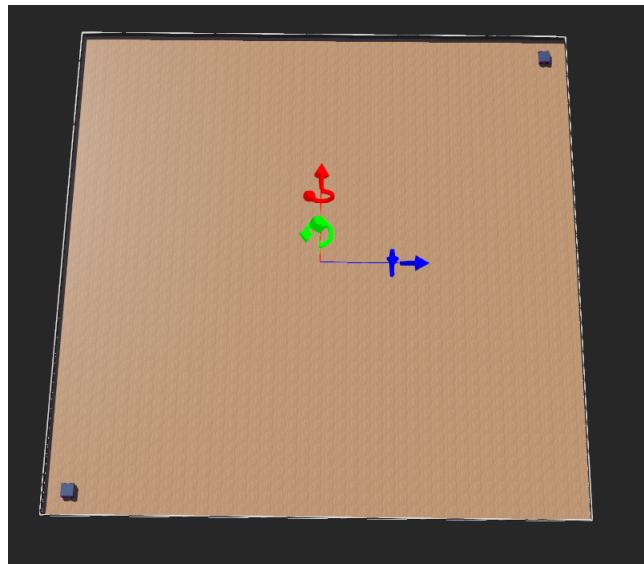


Fonte: Autora

5.2.2 Situação 2 - Terreno regular com dois robôs

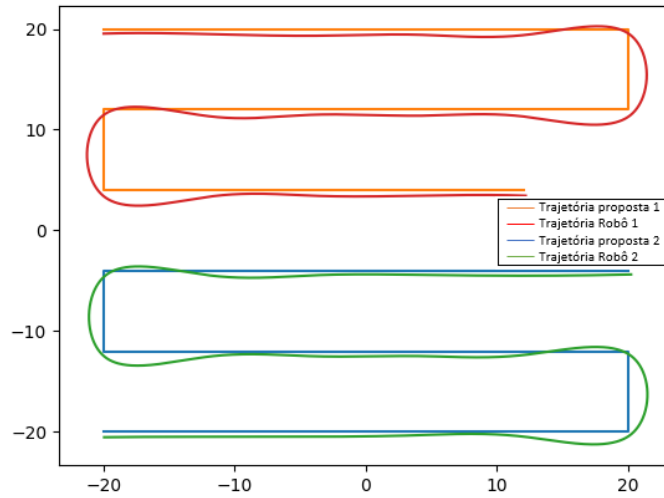
A Figura 26 representa o ambiente já com dois robôs. Sobre o desenho da trajetória, pode-se observar o resultado para dois robôs na Figura 27.

Figura 26: Representação do ambiente em terreno regular com dois Robôs.



Fonte: Autora

Figura 27: Trajetórias de referência e trajetórias reais para dois robôs em terreno regular.

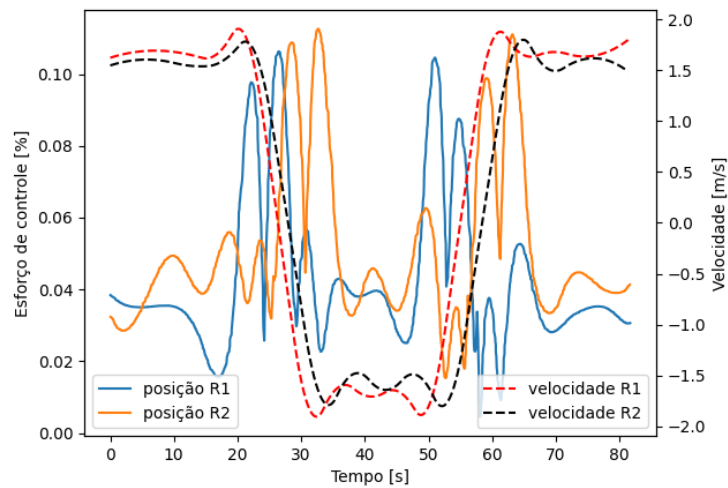


Fonte: Autora

5.2.2.1 Controle e trajetória

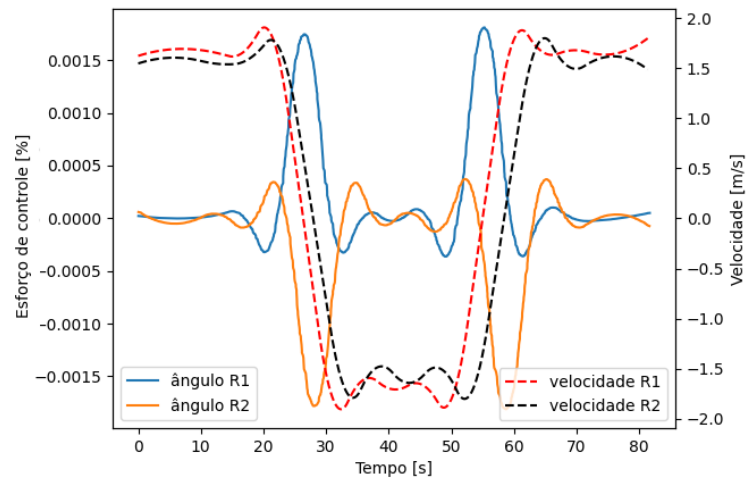
Nas simulações da Situação 2, os esforços de controle aplicados em ambos os casos (atuação nos motores) podem ser vistos na Figura 28 referente ao esforço de controle de posição e perfil velocidade e na Figura 29 referente ao esforço de controle do ângulo de inclinação θ e perfil velocidade. Para simplificar a nomenclatura nos gráficos, o Robô 1 poderá ser referenciado como R1 e o Robô 2 como R2.

Figura 28: Esforço de controle de posição para cada um dos dois robôs em terreno regular quando comparado com o perfil velocidade.



Fonte: Autora

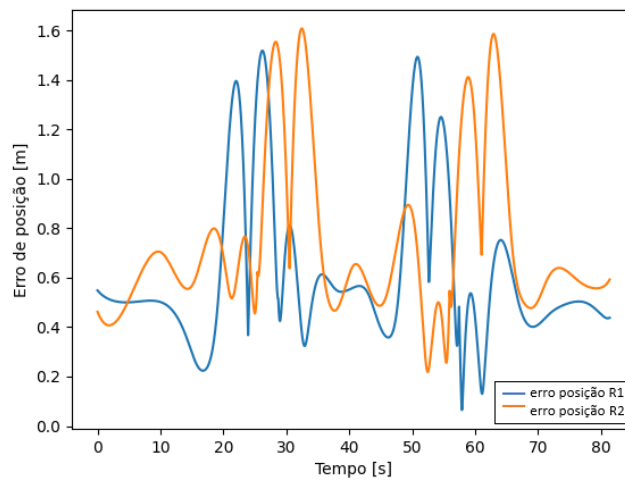
Figura 29: Esforço de controle do ângulo θ para cada um dos dois robôs em terreno regular quando comparado com o perfil velocidade.



Fonte: Autora

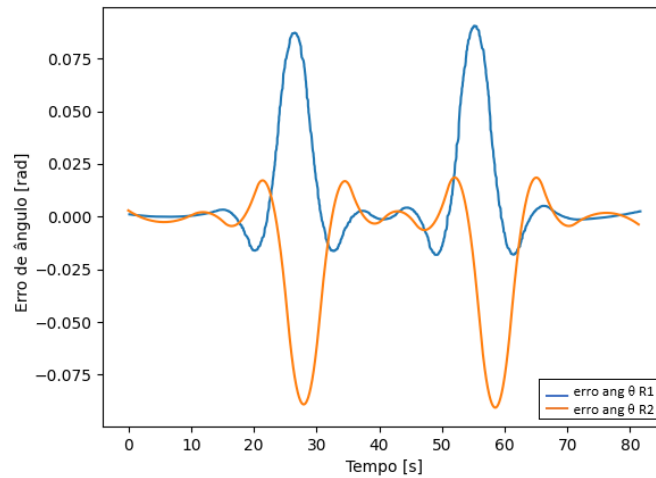
Os erros de posição e de ângulos, são apresentados nas Figuras 30 e 31 para a Situação 2.

Figura 30: Erro de posição na trajetória de dois robôs em terreno regular.



Fonte: Autora

Figura 31: Erro de ângulo θ na trajetória de dois robôs em terreno regular.



Fonte: Autora

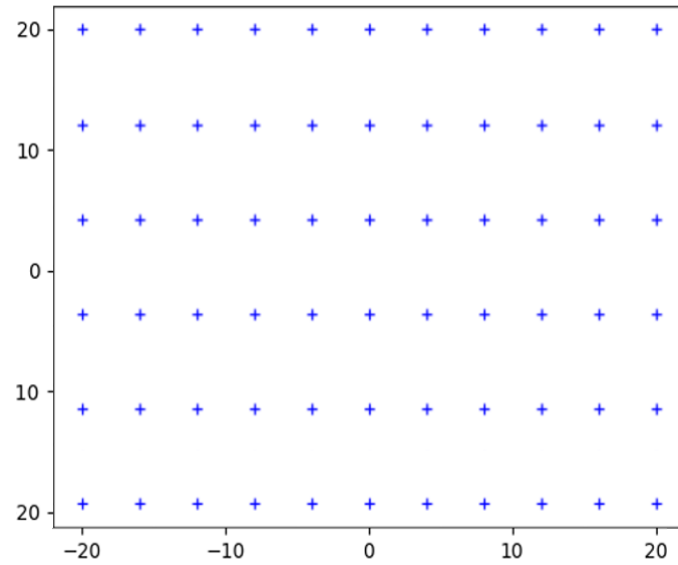
Quando comparado a Situação 1, terreno regular com um robô e a Situação 2, o mesmo terreno regular, porém com dois robôs, observa-se o comportamento parecido porém na situação dois a atividade é feita praticamente na metade do tempo já que a atividade é distribuída para mais de um robô. Este comportamento um pouco mais linear se aplica já que o terreno não apresenta irregularidades, fazendo com que os esforços encontrados para ambas situações sejam semelhantes. Apesar disso, é observado nas simulações erros de posição, e erros de ângulo, principalmente na existência das curvas para mudança de sentido e direção para realizar a semeadura.

5.3 Terreno irregular

Este ambiente é o que mais se aproxima dos desafios comumente enfrentados por agricultores, com irregularidades no solo e diferentes altitudes em sua topografia. Para as simulações, conforme visto no capítulo 4 as opções de *xDimension* e *yDimension* foram de 4 e 10, representado para a Situação 3 e posteriormente de 3 e 20 para a Situação 4. Esta última caracterizará uma irregularidade maior.

Para as simulações em ambiente irregular, optou-se por diminuir a quantidade de pontos a serem percorridos para observar melhor o comportamento de cada um dos robôs em intervalos de tempo parecidos com as simulações em terreno regular. Os pontos da trajetória a ser seguida podem ser observados na Figura 32.

Figura 32: Pontos da trajetória a ser seguida em terreno irregular.

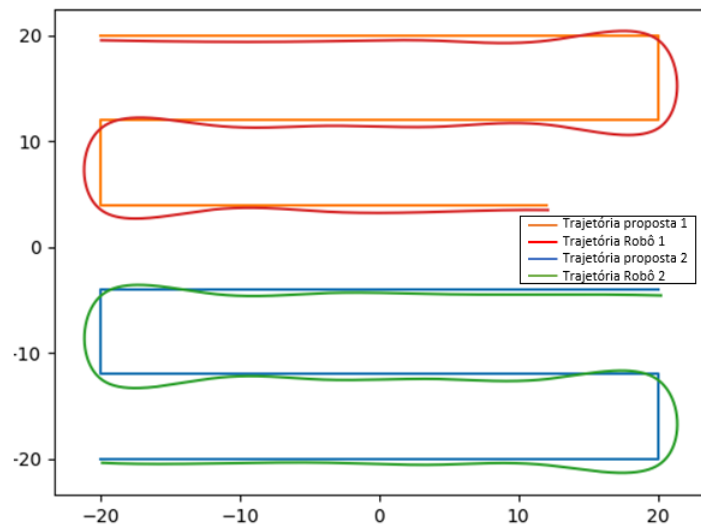


Fonte: Autora

5.3.1 Situação 3 - Terreno irregular com dois robôs

Simulação realizada com dois robôs em terreno irregular $xDimension = 4$ e $yDimension = 10$. As trajetórias realizadas pelos dois robôs são observadas na Figura 33.

Figura 33: Trajetórias de referência e trajetórias reais para dois robôs em terreno irregular.

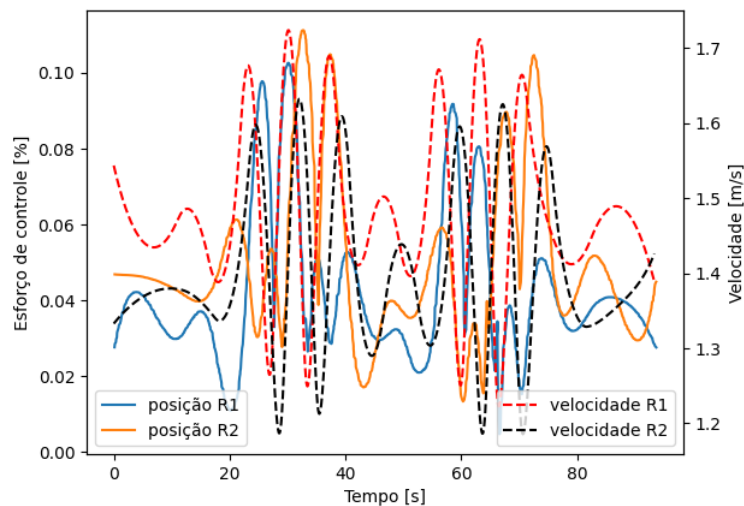


Fonte: Autora

5.3.1.1 Controle e trajetória

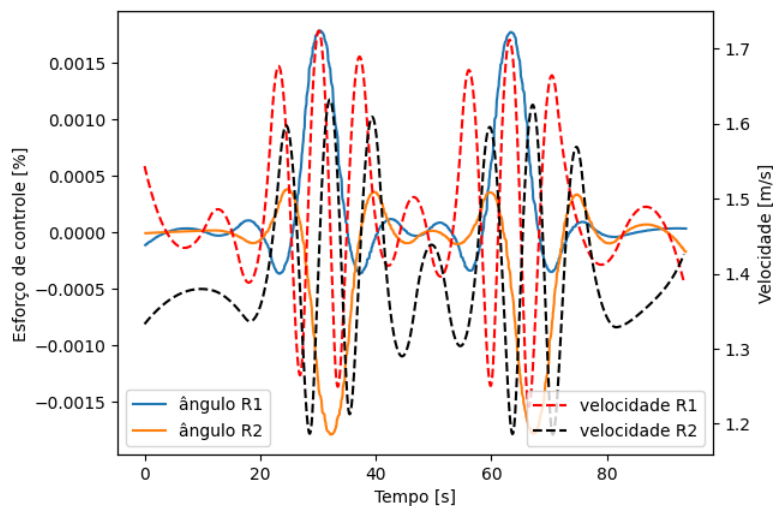
Durante tais simulações, o esforço de controle de posição comparado com o perfil velocidade em terreno irregular está aplicado na Figura 34. O esforço de controle do ângulo de inclinação θ e perfil velocidade em terreno irregular é apresentado na Figura 35

Figura 34: Esforço de controle de posição para cada um dos dois robôs em terreno irregular quando comparado com o perfil velocidade.



Fonte: Autora

Figura 35: Esforço de controle do ângulo θ para cada um dos dois robôs em terreno irregular quando comparado com o perfil velocidade.

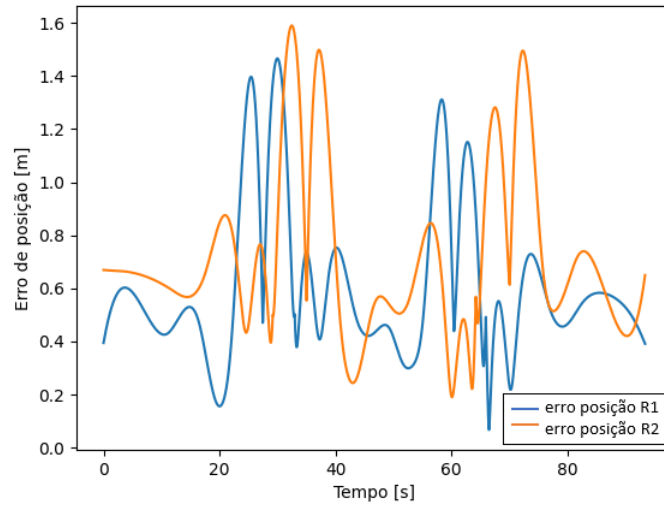


Fonte: Autora

Os erros de posição e de ângulos, que são as entradas dos controladores, respectivamente nos casos de um e dois robôs são apresentados nas Figuras 36 e 37.

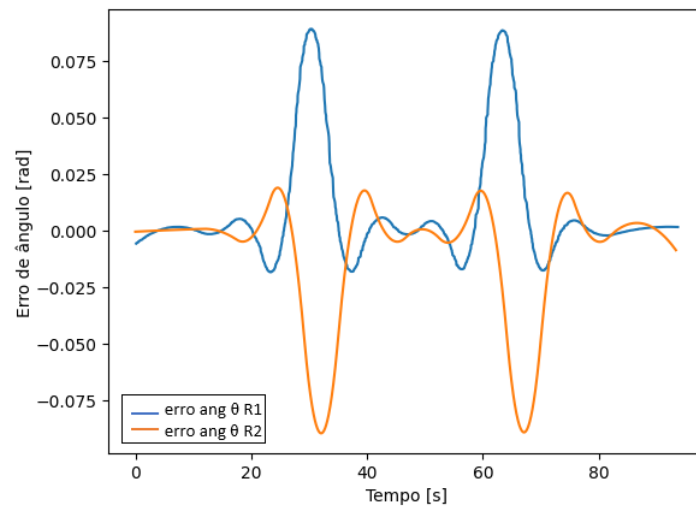
Conforme observado nas figuras, a irregularidade do terreno, representada pela va-

Figura 36: Erro de posição na trajetória de dois robôs.



Fonte: Autora

Figura 37: Erro de ângulo na trajetória de dois robôs em terreno irregular.



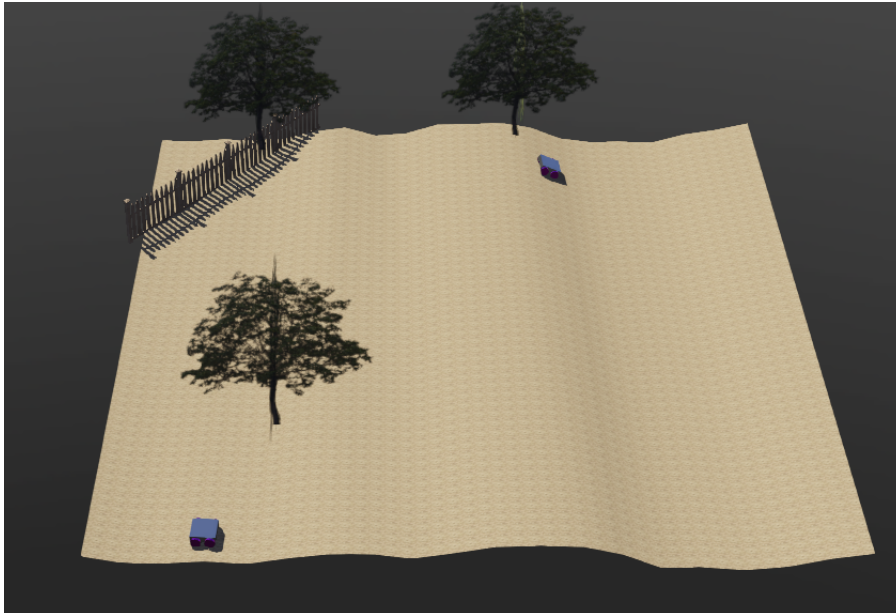
Fonte: Autora

riação da altitude da topografia em uma situação real, adiciona um esforço maior ao controlador, ocorrendo um comportamento diferente em cada um dos robôs, já que ambos enfrentam situações de terreno distintas no mesmo intervalo de tempo. Como resultado, temos esforços de controle defasados e o Robô 1 aparenta enfrentar uma altitude de topografia mais desafiadora, dado sua resposta mais atrasada em relação a resposta do robô 2 e com erros maiores, conforme observado nas Figuras 36 e 37.

5.3.2 Situação 4 - Terreno irregular com obstáculo cerca

Simulação realizada com dois robôs em terreno irregular com Webots™ configurado com $xDimension = 3$ $yDimension = 20$. Nesta simulação, além de aumentar o nível de irregularidade da topografia do terreno, foi adicionada uma cerca como obstáculo na trajetória a ser seguida de um dos robôs para observar seu comportamento, representadas pela Figura 38. As árvores existentes no ambiente não impactaram as trajetórias dos robôs pois não coincidiram com pontos onde os robôs deveriam percorrer, representando que em uma situação real é possível manter vegetação e realizar o cultivo.

Figura 38: Representação da simulação em terreno irregular com obstáculo cerca.



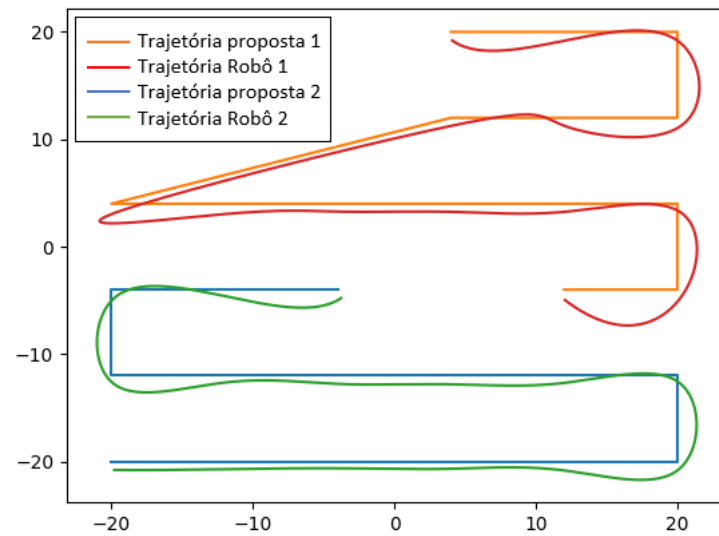
Fonte: Autora

5.3.2.1 Controle e trajetória

O controle para dois robôs em terreno irregular com um obstáculo cerca pode ser observado na Figura 39.

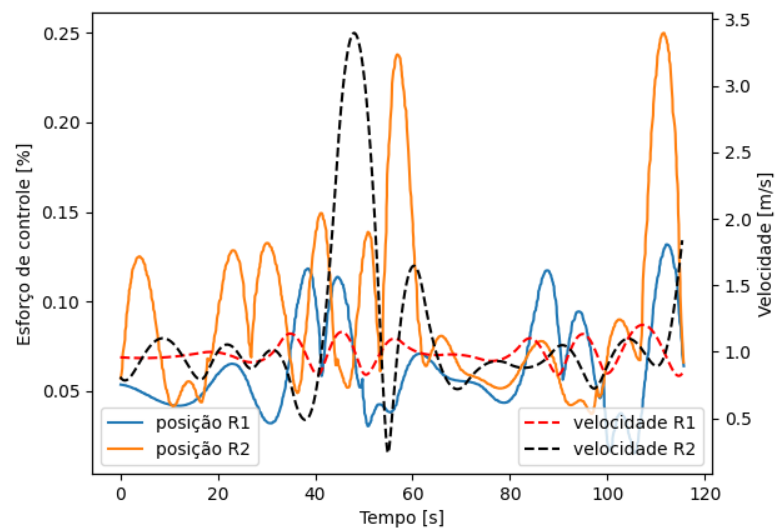
Durante tais simulações, o esforço de controle aplicado referente ao esforço de controle de posição e perfil velocidade esta apresentado na Figura 40. O esforço de controle do ângulo de inclinação θ e perfil velocidade é observado na Figura 41. Os erros de posição e de ângulos, que são as entradas dos controladores, respectivamente nos casos de um e dois robôs são apresentados nas Figuras 42 e 43.

Figura 39: Controle para dois robôs em terreno irregular com obstáculo cerca.



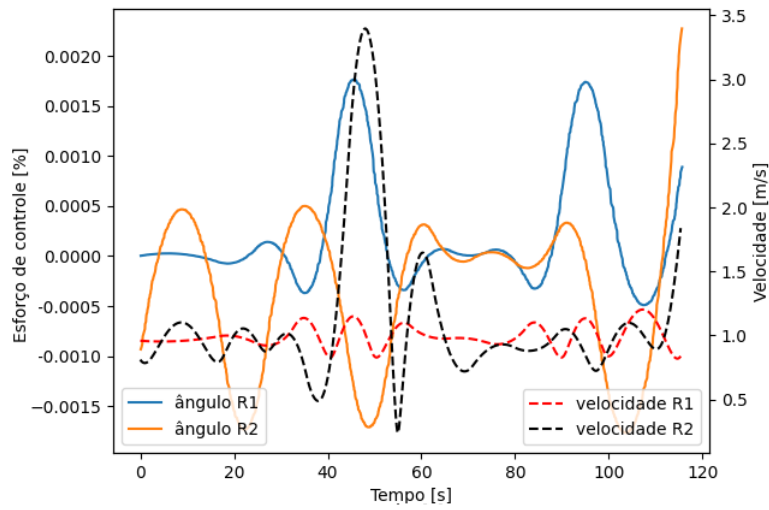
Fonte: Autora

Figura 40: Esforço de controle de posição para cada um dos dois robôs em terreno irregular com obstáculo quando comparado com o perfil velocidade.



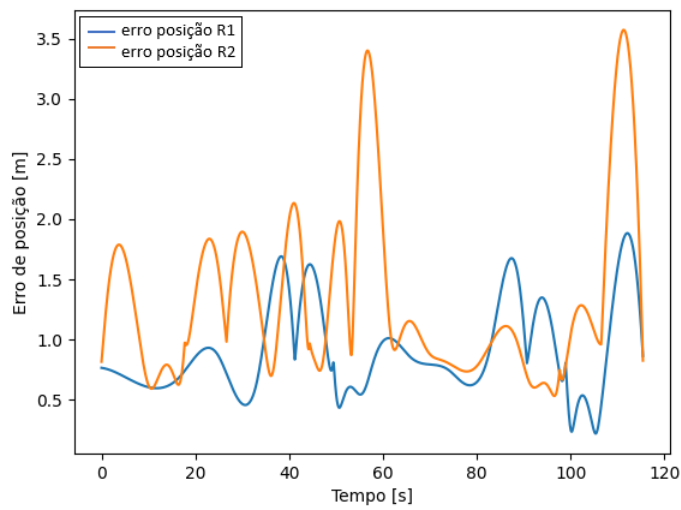
Fonte: Autora

Figura 41: Esforço de controle do ângulo θ para cada um dos dois robôs em terreno irregular com obstáculo quando comparado com o perfil velocidade.



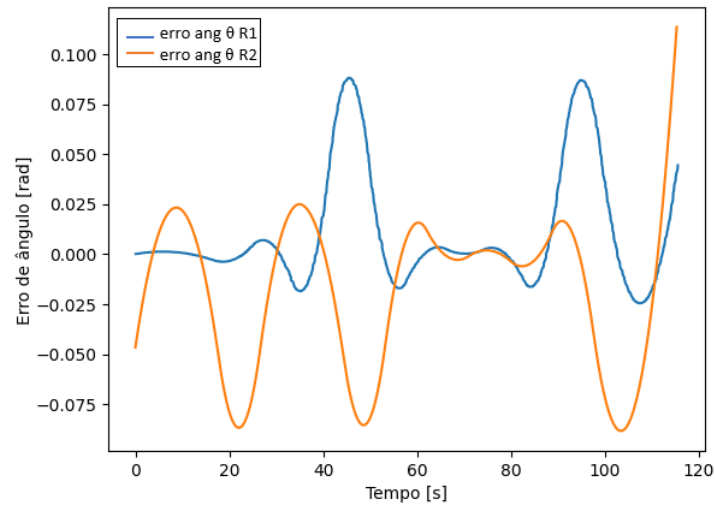
Fonte: Autora

Figura 42: Erro de posição na trajetória de dois robôs em terreno irregular com obstáculo cerca.



Fonte: Autora

Figura 43: Erro de ângulo θ na trajetória de dois robôs em terreno irregular com obstáculo cerca.



Fonte: Autora

Analisando os resultados, percebe-se que, apesar da adição de uma cerca, o robô 1 percorre uma trajetória diferente, quando comparado às situações das simulações anteriores, mas não apresenta um esforço de controle adicional significativo por este motivo. Se observa um aumento na energia despendida, mas não um aumento significativo na amplitude do esforço, já que a energia é proporcional integral do quadrado do esforço.

O que o obstáculo realmente impacta o esforço de controle é referente ao sentido que o robô precisa realizar na trajetória e enfrentar uma maior irregularidade quando comparado com o robô 2. A principal variação observada é referente a maior irregularidade disposta a este terreno, fazendo com que o Esforço de controle aumente para ambos os robôs, principalmente ao Robô 1. Assim, como na Situação 3, como resultado, temos esforços de controle defasados e o Robô 1 aparenta enfrentar uma altitude de topografia mais desafiadora, dado sua resposta mais atrasada em relação a resposta do robô 2 e com erros maiores, conforme observado nas Figuras 42 e 43.

Em linhas gerais, comparando as quatro situações de simulação apresentadas, observa-se a grande sensibilidade do erro de ângulo à curvas agudas, porém, isso parece não ser muito relevante no esforço de controle final, uma vez que se trata de um erro de curta duração, quando comparado a proporção de trajetória representada. Pode-se notar também um erro de posição em trajetórias retas, em princípio isso não seria esperado de uma simulação. Porém, a estimativa de ângulo é obtida de sensores inerciais, que contém erros não lineares durante toda sua execução. Esses sensores são os que definem a trajetória por meio dos sinais de erro para o controlador, da mesma forma que em

uma situação real o robô não teria acesso à sua posição em termos absolutos mas apenas uma posição estimada por tais sensores. Um esforço de filtrar tais leituras em trajetórias retilíneas pode ser feito à custo de perda de desempenho nas curvas ou de um sistema mais complexo. Conforme foi observado nas Situações 3 e 4, de terreno irregular, percebe-se ainda, que a topografia e altitude do terreno exigem um maior esforço aplicado, do que quando comparado até mesmo com a apresentação de um obstáculo.

6 CONCLUSÃO

As aplicações de robótica móvel ainda são limitadas quando se considera a indústria em geral. Em grande parte, os sistemas de navegação autônoma ainda não estão preparados para ambientes dinâmicos e aplicações em espaço aberto não controlado. Existe grande volume de literatura de aplicação para a robótica móvel que tenta formular soluções para as mais diversas aplicações embora com pouco sucesso em situações não controladas.

Uma área com grande promessa de aplicações é a agricultura, que aponta em pesquisa recentes a robótica móvel como auxiliar para tarefas como semeadura, colheita, monitoramento de pragas e aspersão de aditivos. Todas essas tarefas são bem resolvidas com produção de larga escala em monoculturas e terrenos uniformes e planos. São representadas pelas grandes máquinas no campo com suas ferramentas uniformizadas e de grande impacto sobre o solo.

A inserção da robótica móvel em um contexto mais inteligente deverá ser possível com máquinas menores, implementos de aplicação mais geral e lavouras combinadas e não uniformes. Nesse contexto, a navegação autônoma e a geração e controle de trajetória se tornam importantes, quando apoiadas em sensoriamento e identificação de tarefas e obstáculos em tempo real.

A literatura de robótica móvel possui grande número de estratégias para geração de trajetória, porém pressupõe ambientes controlados em que se possam aplicar técnicas de otimização clássicas. As trajetórias possuem formas matemáticas conhecidas e a sua resposta quando aplicadas ao sistema são estimadas com precisão por modelos de equações diferenciais. Esses modelos também consultam a cinemática de formas práticas de robôs disponíveis, porém não tem considerações de atritos e deslizamentos de forma realista.

Dessa forma, este trabalho procurou se situar como proposta de técnica para uma família de aplicações em que há a necessidade de rastreamento de localização por sensores, que utilize a navegação autônoma com considerações de energia limitada e permita trajetórias simples porém arbitrarias. Ainda, o controle de trajetória deve ser robusto e

usando técnicas que admitam não linearidades.

Adotou-se aqui o uso de dois controladores PID para seguir uma trajetória pré-definida com pontos em que se admite uma variação, mas que permite atualização do projeto do controlador em tempo real conforme a reação do sistema ao controle. Os parâmetros do controle são simples (apenas 6), o que permite cálculo em tempo real e controle embarcado, reduzindo a dependência de redes de comunicação de alta velocidade, quase sempre indisponíveis em lavouras de pequeno e médio porte.

Os resultados indicam que esse tipo de abordagem é possível e portanto constitui uma direção de estudos que leva em consideração as limitações do meio. A integração de mais robôs por meio do próprio algoritmo de atualização do controlador simplifica o problema de divisão de trabalho novamente com baixa dependência de redes de comunicação de alta velocidade.

Essas linhas gerais de pensamento podem ser expandidas na direção de aumentar área cobertas por diversos robôs, explorando o limite entre robótica colaborativa e enxame de robôs. Podendo-se no futuro investigar os efeitos da maior divisão de trabalho mesmo em simulação.

REFERÊNCIAS

- ALMASRI, M. M.; ALAJLAN, A. M.; ELLEITHY, K. M. Trajectory planning and collision avoidance algorithm for mobile robotics system. *IEEE Sensors Journal*, v. 16, n. 12, p. 5021–5028, 2016.
- ANDRADE, A. K. P. de. *Geração procedural de terrenos tridimensionais ilimitados na GPU*, 2021.
- ANDRÉ, L.; PARPINELLI, R. S. Tutorial sobre o uso de técnicas para o controle online de parâmetros em algoritmos de inteligência de enxame e computação evolutiva. *RITA*, v. 21, n. 2, p. 90–135, 2014.
- BECHAR, A.; VIGNEAULT, C. Agricultural robots for field operations: Concepts and components. *Biosystems Engineering*, Elsevier, v. 149, p. 94–111, 2016.
- BOYLE, M. The integration of angular velocity. *Advances in Applied Clifford Algebras*, Springer Science and Business Media LLC, v. 27, n. 3, p. 2345–2374, May 2017.
- Coppelia Robotics. *CoppeliaSim*. 2023. Disponível em: <https://www.coppeliarobotics.com/>. Acesso em: 22 fev. 2023.
- COSTA, G. de O. *Identificação de Parâmetros Cinemáticos e Controle Dinâmico de Robôs Móveis com Rodas tipo Skid-Steering Utilizando Múltiplos Sensores Inerciais*. Tese (Doutorado) — Universidade de Brasília, 2019.
- Cyberbotics. *Glossary*. 2021. Disponível em: <https://cyberbotics.com/doc/reference/glossary>. Acesso em: 16 set. 2021.
- Cyberbotics. *Node and function*. 2021. Disponível em: <https://cyberbotics.com/doc/reference/nodes-and-functions>. Acesso em: 16 set. 2021.
- Cyberbotics. *Node Chart*. 2021. Disponível em: <https://cyberbotics.com/doc/reference/node-chart>. Acesso em: 16 set. 2021.
- Cyberbotics. *Proto and Keywords*. 2021. Disponível em: <https://cyberbotics.com/doc/reference/nodes-and-keywords>. Acesso em: 01 out. 2021.
- Cyberbotics. *Reference Manual - Introduction*. 2021. Disponível em: <https://cyberbotics.com/doc/reference/introduction>. Acesso em: 16 set. 2021.
- Cyberbotics. *Tutorial 6: 4-Wheeled Robot (60 Minutes)*. 2021. Disponível em: <https://cyberbotics.com/doc/guide/tutorial-6-4-wheels-robot>. Acesso em: 21 abr. 2021.
- Cyberbotics. *Webots User Guide Floors*. 2021. Disponível em: <https://www.cyberbotics.com/doc/guide/object-floors?version=cyberbotics:R2019a>. Acesso em: 05 abr. 2023.

- Cyberbotics. *Webots User Guide Kernel Description*. 2021. Disponível em: <https://cyberbotics.com/doc/guide/index>. Acesso em: 21 jun. 2021.
- Cyberbotics. *Physics*. 2022. Disponível em: <https://cyberbotics.com/doc/reference/physics>. Acesso em: 03 fev. 2022.
- DING, N. et al. A comprehensive review on automatic mobile robots: applications, perception, communication and control. *Journal of Circuits, Systems and Computers*, World Scientific, v. 31, n. 08, 2022.
- DIRIK M., K. A. . C. O. Global path planning and path-following for wheeled mobile robot using a novel control structure based on a vision sensor. *International Journal of Fuzzy Systems*, v. 22, n. 08, p. 1880–1891, 2020.
- DU, K. L.; SWAMY, M. N. *Search and optimization by metaheuristics: Techniques and algorithms inspired by nature*. [S.l.]: Springer, 2016. 1–434 p.
- DUCKETT, T. et al. Agricultural robotics: the future of robotic agriculture. *arXiv preprint arXiv:1806.06762*, 2018.
- DURMUS, H. et al. The design of general purpose autonomous agricultural mobile-robot: “agrobot”. In: *2015 Fourth International Conference on Agro-Geoinformatics (Agro-geoinformatics)*. [S.l.: s.n.], 2015. p. 49–53.
- Empresa Brasileira de Pesquisa Agropecuária. *Automação e agricultura de precisão*. 2023. Disponível em: <https://www.embrapa.br/tema-mecanizacao-e-agricultura-de-precisao/nota-tecnica#:~:text=A%20automa%C3%A7%C3%A3o%20agropecu%C3%A1ria%20pode%20ser,a%20capacidade%20de%20trabalho%20humano>. Acesso em: 16 fev. 2023.
- FARLEY, A.; WANG, J.; MARSHALL, J. A. How to pick a mobile robot simulator: A quantitative comparison of coppeliasim, gazebo, morse and webots with a focus on accuracy of motion. *Simulation Modelling Practice and Theory*, v. 120, p. 102629, 2022.
- Fendt. *Robôs semeadores de última geração: O Fendt Xaver atinge a maioria*. 2020. Disponível em: <https://www.fendt.com/br/2-fendt-xaver>. Acesso em: 16 jun. 2021.
- GAO, P. et al. Improved position estimation algorithm of agricultural mobile robots based on multisensor fusion and autoencoder neural network. *Sensors*, v. 22, n. 4, 2022.
- GAO, X. et al. Review of wheeled mobile robots’ navigation problems and application prospects in agriculture. *IEEE Access*, v. 6, p. 49248–49268, 2018.
- GEORGAKOPOULOS-SOARES, I.; PAPAZOGLU, E. L.; MARKOPOULOS, A. P. On the use of the perlin noise function to calculate the laser absorption coefficient by rough surfaces. *Simulation Modelling Practice and Theory*, v. 124, p. 102722, 2023.
- HACKENHAAR, N. M.; HACKENHAAR, C.; ABREU, Y. V. de. Robótica na Agricultura. *Interações (Campo Grande)*, v. 16, n. 1, p. 119–129, 2015.
- HAJJAJ, S. S. H.; SAHARI, K. S. M. Review of agriculture robotics: Practicality and feasibility. In: IEEE. *2016 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*. [S.l.], 2016. p. 194–198.

- INGRAND, F.; GHALLAB, M. Deliberation for autonomous robots: A survey. *Artificial Intelligence*, v. 247, p. 10–44, 2017.
- JHA, K. et al. A comprehensive review on automation in agriculture using artificial intelligence. *Artificial Intelligence in Agriculture*, Elsevier, v. 2, p. 1–12, 2019.
- KAGAN, E.; SHVALB, N.; BEN-GAL, I. *Autonomous mobile robots and multi-robot systems: Motion-planning, communication, and swarming*. [S.l.: s.n.], 2019. 11-13 p.
- KIM, J. Trajectory generation of a two-wheeled mobile robot in an uncertain environment. *IEEE Transactions on Industrial Electronics*, v. 67, n. 7, p. 5586–5594, 2020.
- KIM, W.-S.; LEE, W.-S.; KIM, Y.-J. A review of the applications of the internet of things (iot) for agricultural automation. *Journal of Biosystems Engineering*, Springer, v. 45, p. 385–400, 2020.
- KING, A. Technology: The future of agriculture. *Nature*, v. 544, n. 7651, p. S21–S23, 2017.
- KOSTJUKOV, v.; MEDVEDEV, M.; PSHIKHOPOV, v. Method for optimizing of mobile robot trajectory in repeller sources field. *Informatics and Automation*, v. 20, n. 7, p. 690–726, 2021. Disponível em: <https://elibrary.ru/item.asp?id=46276504>.
- KOUBAA, A. et al. *Robot Path Planning and Cooperation*. [S.l.]: Springer International Publishing, 2018.
- KOZLOWSKI, K.; PAZDERSKI, D. Global positioning systems, inertial navigation, and integration. *International journal of applied mathematics and computer science*, v. 14, p. 477–496, 2004.
- KULBACKI, M. et al. Survey of drones for agriculture automation from planting to harvest. In: IEEE. *2018 IEEE 22nd International Conference on Intelligent Engineering Systems (INES)*. [S.l.], 2018. p. 000353–000358.
- LANDI, G.; ZAMPINI, A. *Linear Algebra and Analytic Geometry for Physical Sciences*. [S.l.]: Spinger, 2018.
- LYTRIDIS, C. et al. An overview of cooperative robotics in agriculture. *Agronomy*, MDPI, v. 11, n. 9, p. 1818, 2021.
- MAHMUD, M. S. A. et al. Multi-objective path planner for an agricultural mobile robot in a virtual greenhouse environment. *Computers and Electronics in Agriculture*, v. 157, p. 488–499, 2019.
- MOYSIADIS, V. et al. Mobile robotics in agricultural operations: A narrative review on planning aspects. *Applied Sciences*, v. 10, n. 10, 2020. ISSN 2076-3417.
- OLIVEIRA, L. F.; MOREIRA, A. P.; SILVA, M. F. Advances in agriculture robotics: A state-of-the-art review and challenges ahead. *Robotics*, MDPI, v. 10, n. 2, p. 52, 2021.
- Open Source Robotics Foundation. *About Gazebo*. 2014. Disponível em: <https://gazebo.org/about>. Acesso em: 20 fev. 2023.

Open Source Robotics Foundation. *Features*. 2014. Disponível em: <https://classic.gazebosim.org/>. Acesso em: 20 fev. 2023.

PURANIK, V. et al. Automation in agriculture and iot. In: IEEE. *2019 4th international conference on internet of things: smart innovation and usages (IoT-SIU)*. [S.l.], 2019. p. 1–6.

RUBIO, F.; VALERO, F.; LLOPIS-ALBERT, C. A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, v. 16, n. 2, 2019.

SERRALHEIRO, W. A. de O. *Arquitetura de Controle de Movimento para um Robô Móvel sobre Rodas visando Otimização Energética*. Tese (Doutorado) — Universidade de São Paulo, São Paulo, 2018.

SIEGWART, I. R. N. R. (Ed.). *Introduction to Autonomous Mobile Robots*. [S.l.]: Massachusetts Institute of Technology, 2004.

TIAN, H. et al. Computer vision technology in agricultural automation—a review. *Information Processing in Agriculture*, Elsevier, v. 7, n. 1, p. 1–19, 2020.

TSAI, P. W. et al. Bat algorithm inspired algorithm for solving numerical optimization problems. In: *Mechanical Engineering, Materials and Energy*. [S.l.]: Trans Tech Publications Ltd, 2012. v. 148, p. 134–137.

WALAMBE, R. et al. Optimal trajectory generation for car-type mobile robot using spline interpolation this work is carried out under the research project grant sanctioned under the wos-a scheme by department of science and technology (dst), govt. of india. *IFAC-PapersOnLine*, v. 49, n. 1, p. 601–606, 2016.

WANG, D. et al. A review of deep learning in multiscale agricultural sensing. *Remote Sensing*, Multidisciplinary Digital Publishing Institute, v. 14, n. 3, p. 559, 2022.

YANG, X.-S. A new metaheuristic bat-inspired algorithm. In: *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*. [S.l.]: Springer Berlin Heidelberg, 2010. p. 65–74.