Pontifícia Universidade Católica
do Rio de Janeiro

Igor Ferreira da Costa

Detecção Visual de Fileira de Plantação com Tarefa Auxiliar de Segmentação para Navegação de Robôs Móveis

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Engenharia Elétrica, do Departamento de Engenharia Elétrica da PUC-Rio.

Orientador: Prof. Wouter Caarls

Rio de Janeiro
Agosto de 2023

Pontifícia Universidade Católica
do Rio de Janeiro

Igor Ferreira da Costa

Detecção Visual de Fileira de Plantação com Tarefa Auxiliar de Segmentação para Navegação de Robôs Móveis

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Engenharia Elétrica da PUC-Rio. Aprovada pela Comissão Examinadora abaixo:

Prof. Wouter Caarls
Orientador
Departamento de Engenharia Elétrica – PUC-Rio

Antonio Candea Leite
Norwegian University of Life Sciences

Karla Tereza Figueiredo Leite
Universidade do Estado do Rio de Janeiro

Raul Queiroz Feitosa
Departamento de Engenharia Elétrica – PUC-Rio

Rio de Janeiro, 29 de Agosto de 2023

Igor Ferreira da Costa

Graduado em Engenharia Elétrica pela Universidade Federal Fluminense

Ficha Catalográfica

Costa, Igor Ferreira da

Detecção Visual de Fileira de Plantação com Tarefa Auxiliar de Segmentação para Navegação de Robôs Móveis / Igor Ferreira da Costa; orientador: Wouter Caarls. – 2023.

80 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, 2023.

Inclui bibliografia

1. Engenharia Elétrica – Teses. 2. Navegação Autônoma. 3. Deep Learning. 4. Robótica Agrícola. I. Caarls, Wouter. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. III. Título.

CDD: 621.3

Agradecimentos

A Deus, que está sempre comigo e é fonte de força, paz e sabedoria.

Ao meu orientador, Wouter Caarls, que aceitou me guiar ao longo desta jornada e esteve sempre disposto a ensinar e ajudar. Muito obrigado por toda ajuda e conhecimento passado ao longo dos últimos dois anos.

Ao professor Antonio Candea Leite, que, juntamente do professor Wouter, me proporcionaram a oportunidade de realizar parte deste trabalho na Norwegian University of life Sciences através do projeto UTFORSK EnTechAgri. Muito obrigado por toda ajuda e companhia no tempo que estive nesta viagem, ficará para sempre marcado.

A CAPES, a PUC-Rio e a NMBU, através do projeto UTFORSK EnTechAgri, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

Aos meus pais, que sempre me apoiaram e deram o suporte necessário ao longo da minha vida. Sou infinitamente grato a ambos por todo o ensinamento passado.

A minha namorada, Larissa Decarlo, a qual esteve constantemente ao meu lado, apoiando, incentivando e ajudando em todos os momentos. Obrigado por me ajudar a sempre seguir em frente.

Aos meus amigos, obrigado por toda ajuda e estudos realizados juntos. E, também, pelas horas de descontração e diversão. Em especial Vitor Bento, o qual foi fundamental para a decisão de trilhar este caminho.

Aos professores que participaram da Comissão examinadora.

A todos os professores e funcionários da PUC-Rio e NMBU pelos ensinamentos e pela ajuda.

A todos os amigos e familiares que de uma forma ou de outra me estimularam ou me ajudaram.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

Costa, Igor Ferreira da; Caarls, Wouter. **Detecção Visual de Fileira de Plantação com Tarefa Auxiliar de Segmentação para Navegação de Robôs Móveis**. Rio de Janeiro, 2023. 80p. Dissertação de Mestrado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Com a evolução da agricultura inteligente, robôs autônomos agrícolas têm sido pesquisados de forma extensiva nos últimos anos, ao passo que podem resultar em uma grande melhoria na eficiência do campo. No entanto, navegar em um campo de cultivo aberto ainda é um grande desafio. O RTK-GNSS é uma excelente ferramenta para rastrear a posição do robô, mas precisa de mapeamento e planejamento precisos, além de ser caro e dependente de qualidade do sinal. Como tal, sistemas *on-board* que podem detectar o campo diretamente para guiar o robô são uma boa alternativa. Esses sistemas detectam as linhas com técnicas de processamento de imagem e estimam a posição aplicando algoritmos à máscara obtida, como a transformada de Hough ou regressão linear. Neste trabalho, uma abordagem direta é apresentada treinando um modelo de rede neural para obter a posição das linhas de corte diretamente de uma imagem RGB. Enquanto a câmera nesses sistemas está, geralmente, voltada para o campo, uma câmera próxima ao solo é proposta para aproveitar túneis ou paredes de plantas formadas entre as fileiras. Um ambiente de simulação para avaliar o desempenho do modelo e o posicionamento da câmera foi desenvolvido e disponibilizado no Github. Também são propostos quatro conjuntos de dados para treinar os modelos, sendo dois para as simulações e dois para os testes do mundo real. Os resultados da simulação são mostrados em diferentes resoluções e estágios de crescimento da planta, indicando as capacidades e limitações do sistema e algumas das melhores configurações são verificadas em dois tipos de ambientes agrícolas.

Palavras-chave

Navegação Autônoma; Deep Learning; Robótica Agrícola.

Abstract

Costa, Igor Ferreira da; Caarls, Wouter (Advisor). **Visual Crop Row Detection with Auxiliary Segmentation Task for Mobile Robot Navigation** . Rio de Janeiro, 2023. 80p. Dissertação de Mestrado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Autonomous robots for agricultural tasks have been researched to great extent in the past years as they could result in a great improvement of field efficiency. Navigating an open crop field still is a great challenge. RTK-GNSS is a excellent tool to track the robot's position, but it needs precise mapping and planning while also being expensive and signal dependent. As such, onboard systems that can sense the field directly to guide the robot are a good alternative. Those systems detect the rows with adequate image processing techniques and estimate the position by applying algorithms to the obtained mask, such as the Hough transform or linear regression. In this work, a direct approach is presented by training a neural network model to obtain the position of crop lines directly from an RGB image. While, usually, the camera in these kinds of systems is looking down to the field, a camera near the ground is proposed to take advantage of tunnels or walls of plants formed between rows. A simulation environment for evaluating both the model's performance and camera placement was developed and made available on Github, also four datasets to train the models are proposed, being two for the simulations and two for the real world tests. The results from the simulation are shown across different resolutions and stages of plant growth, indicating the system's capabilities and limitations. Some of the best configurations are then verified in two types of agricultural environments.

Keywords

Autonomous Navigation; Deep Learning; Agricultural Robotic.

Sumário

Lista de figuras

Lista de tabelas

Lista de Abreviaturas

ADI – Análise Digital de Imagens
ASPP – Atrous Spatial Pyramid Pooling
CNN – Convolutional Neural Network
CPU – Central Processing Unit
DART – Dynamic Animation and Robotics Toolkit
GPU – Graphics Processing Unit
GNSS – Global Navigation Satellite System
LiDAR – Light Detection and Ranging
MLP – Multilayer Perceptron
OGRE – Object-Oriented Graphics Rendering Engine
RGB – Sistema de cores *Red, Green, Blue*
ROS – Robot Operating System
RTK – Real Time Kinematics
SOM – System on Module
URDF – Unified Robot Description Format

1
Introdução

A agricultura desempenha um papel base na economia e sociedade, fornecendo matéria prima para diversas indústrias, como a têxtil, cosmética e, principalmente, alimentícia, de forma a ter um papel econômico fundamental para diversas regiões e países. O Brasil, hoje, é um dos principais produtores agrícolas no mundo, e em janeiro de 2023, considerado um mês de entressafra, atingiu US$ $10,22$ bilhões em exportações no setor, segundo o Instituto de Pesquisa Econômica Aplicada [1].

Tamanha produção demanda vastos campos de plantio que, por sua vez, necessitam de cuidados para garantir a qualidade e eficiência desejada [2]. Aumentar a eficiência permite maior margem de lucro aos produtores ou a obtenção de preços competitivos em mercados de grande disputa. Para isso, o monitoramento da plantação se faz necessário para implantar controle de pragas, estado de saúde e crescimento do plantio [2].

Ao longo da história, melhorar a produção agrícola se mostra um desafio constante. No século XIX, a agricultura viu uma diversificação da produção, a chamada policultura, bem como o aumento no uso de pesticidas e fertilizantes e um incentivo no uso de princípios científicos [3], os quais visam proporcionar uma produção com aplicação de técnicas, práticas e instrumentação moderna.

O século XX viu, como resultado da criação do automóvel, a integração e uso desses automóveis, tratores, em suma, no campo [4]. As máquinas foram, então, refinadas a fim de aumentar sua eficiência, o que possibilitou a mecanização de inúmeras etapas do plantio à colheita. Tal mecanização reduziu a dependência do trabalho manual de campo, reduzindo, também, custos e dependência de mão de obra [4], por vezes escassa em certas regiões.

Esse cenário de mecanização observado pode ser considerado como um precursor da automação e robótica agrícola pesquisada no século XXI [5]. Com o aumento do poder computacional e avanços na eficiência e capacidade de baterias, veículos autônomos, ver Figura 1.1, tem sido foco de desenvolvimento [6] [7] [8]. O uso de sistemas de posicionamento cinemático em tempo real, RTK-GNSS, podem providenciar movimentação pré-planejada com alto grau de precisão e acurácia, entretanto o alto custo de implementação [9] [10], especialmente em escala, é um empecilho.

Figura 1.1: Exemplos de robôs autônomos agrícolas: protótipos (b)Mamut e (c)SoyBot-I; comerciais (a)Robotnik Bacchus e (d)EarthSense TerrariaSentia

1.1 Motivação

Sensores como LiDAR (Light Detection and Ranging) [9], câmeras térmicas [10] e RGB (Red Green Blue) com ou sem detecção de profundidade [9] podem ser empregados de diferentes formas para auxiliar a navegação dos veículos em meio a plantações. Ao extrair diferentes características estruturais do ambiente, esses sensores podem guiar o veículo em tarefas de monitoramento, mapeamento, aplicação de tratamento localizado ou mesmo colheita de frutos ou transporte [9] [11]. Diminuindo a necessidade de mão de obra para tarefas simples [11], porém longas e repetitivas, e reduzindo o impacto de custos de sistemas de navegação sofisticados como o RTK-GNSS.

Para navegar em meio a plantação, um posicionamento preciso do robô é desejável. Diversas soluções comerciais fazem uso de sistemas de posicionamento global e câmeras estéreo para corretamente posicionar o robô no ambiente [11], e esse tipo de controle de rota exige preciso mapeamento da plantação e planejamento do posicionamento do robô. Com um posicionamento baseado em imagens RGB é possível oferecer uma alternativa mais simples, onde dado um ponto de partida, o robô pode guiar-se de forma independente ao longo da fileira contendo apenas mecanismos mais simples de verificação de posição e translado até pontos de início e final de operação.

A abordagem *end-to-end* de redes neurais convolucionais também pode ser explorada para extrair diretamente as características das imagens [12]. Soluções propostas anteriormente fazem uso de transformações [9] [13], máscaras e combinação de sistemas para extrair a direção a ser dada ao robô [11] [13]. Com uma parametrização adequada da linha de direção da fileira de plantação e um controle adequado aplicado, deseja-se introduzir um sistema capaz de extrair diretamente esta linha de direção de deslocamento desejada. A partir disso, tem-se, também, a aplicação do conceito *end-to-end* ao processamento de imagem para obtenção do direcionamento.

1.2 Objetivo

> Detecção robusta de linha de direção em fileiras de plantação a partir de imagens RGB, com plantas em vários estágios de crescimento.

O objetivo do trabalho é desenvolver um sistema de navegação autônomo, robusto e de custo acessível ao utilizar uma câmera RGB para determinar a direção a ser percorrida pelo robô. O sistema de navegação deverá ser capaz de guiar o robô de forma autônoma em meio a fileiras de plantação evitando danos a mesma. O sistema fará uso do *framework* ROS com o objetivo de ser facilmente integrável a diferentes robôs que utilizem a plataforma.

1.3 Contribuições

Foi desenvolvido um modelo de rede neural capaz de inferir, a partir de imagens RGB, a linha referência utilizada para a navegação autônoma do robô em tempo real. Esse modelo possui duas formas de implementação distintas baseadas no posicionamento da câmera perante o campo de plantio. Duas variantes adicionais do modelo proposto são, também, desenvolvidas, onde o uso ou não de máscara de segmentação auxiliar e a complexidade da rede utilizada para a extração de características iniciais podem ser alterados.

Um ambiente de simulação capaz de desafiar o sistema de navegação foi elaborado visando criar cenários que demonstrem as vantagens de cada configuração proposta, assim como a comparação com o sistema atual de navegação do robô. Esse ambiente é composto com características aleatórias de posicionamento e tipos de plantas para maior flexibilidade. Uma ferramenta capaz de extrair dados de treinamento automaticamente a partir da simulação também foi desenvolvida, com objetivo de permitir testes de forma facilitada.

Uma ferramenta para gerar os dados necessários para o treinamento de novas redes em diferentes ambientes também foi aprimorada, vinda de estudos anteriores, assim como as ferramentas necessárias ao treinamento das redes neurais desenvolvidas e necessárias à utilização do algoritmo. Todo a navegação autônoma é disponibilizada como uma função Python que opera de forma autônoma como nó ROS, o que torna possível a integração com diferentes sistemas.

Testes em ambiente controlado e real de plantio foram conduzidos para validar os resultados observados em simulação. No segundo ambiente real de plantio avaliado, o impacto causado pela substituição da câmera utilizada também foi estudado.

1.4 Organização do trabalho

O presente trabalho é constituído de seis capítulos.O Capítulo 2 apresenta os conceitos necessários para familiarização e compreensão geral do funcionamento do sistema proposto, parâmetros escolhidos e metodologia empregada, os quais foram extraídos da literatura e pesquisa no campo. Uma breve revisão sobre conceitos básicos de redes neurais convolucionais é feita, além de uma visão geral das arquiteturas base de rede utilizadas e apresentado o conceito de *transfer learning* e tarefas auxiliares no âmbito relevante ao trabalho. Este capítulo trás, também, outros trabalhos que puderam servir de inspiração ao enfrentar desafios de navegação similares e implantar diferentes técnicas.

O Capítulo 3, por sua vez, aborda a metodologia desenvolvida. Os ambientes simulados utilizados são descritos e a parametrização de valores utilizada para cada cenário é proposta. A modelagem do robô virtual também é feita, baseada numa simplificação do Soybot-II, contendo os diferentes posicionamentos de câmeras experimentados. A arquitetura de rede empregada é descrita com suas variações e a composição e divisão do conjunto de dados utilizados para treinamento é caracterizado nesse capítulo.

O Capítulo 4 é responsável por delinear toda a configuração de hardware e software utilizada, parâmetros estabelecidos para o treinamento, ambiente simulado e controle do robô. A topologia do sistema ROS utilizado durante a simulação é descrita, assim como a interação entre os nós, com o objetivo de facilitar futuras integrações. A metodologia de avaliação é apresentada, bem como uma descrição dos testes de campo e ambiente realizados com robôs reais, Soybot-II e Thorvald.

O Capítulo 5 exibe os resultados obtidos nas simulações elaboradas, onde são apontadas as características observadas dos modelos. Uma discussão sobre as diferentes variantes empregadas é conduzida nesse capítulo, de forma a apresentar os ganhos para cada caso. As melhores configurações observadas são testadas em ambientes não virtuais e os resultados debatidos.

Por fim, o Capítulo 6 trás as conclusões obtidas acerca de todo o trabalho desenvolvido, pontuando limitações e oportunidades para estudos futuros.

2
Fundamentação Teórica

Neste capítulo serão abordados fundamentos importantes para a execução do presente trabalho, com início no conceito de Redes Neurais e *Deep Learning*. Em seguida, tem-se principal topologia utilizada ao longo do trabalho, além do conceito de tarefa de aprendizado auxiliar e a utilização de *Transfer Learning* e *Backbones*. Por fim, será apresentado o conceito de navegação autônoma e o estado da arte para esta navegação em campos agrícolas, em especial o robô SoyBot, desenvolvido na PUC-Rio, em parceria com a Agtech Solinftec.

2.1
Redes Neurais Artificiais

O termo Rede Neural Artificial é utilizado para representar o modelo computacional formado por um grande número de unidades menores e simples, chamadas, neste contexto, de neurônios, interligadas entre si em estruturas organizacionais diversas [15]. Para cada neurônio, ver Figura 2.1, as entradas X_i são somadas multiplicadas pelos respectivos pesos W_i, resultando numa soma ponderada Z, e a saída é dada pelo valor da função de ativação Φ, avaliada no valor do somatório Z [12].

$$y = \Phi(Z) \tag{2-1}$$

Onde

$$Z = W_0 \cdot X_1 + W_1 \cdot X_2 + ... + W_n \cdot X_n = \sum_{i=1}^{n} X_i \cdot W_i \tag{2-2}$$

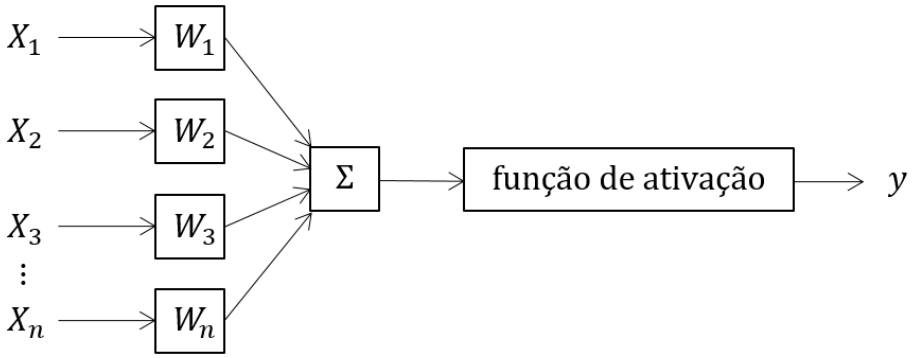

Figura 2.1: Diagrama gráfico do neurônio artificial

As redes neurais artificiais são um tipo de modelo de aprendizado de máquina que é treinado por meio de exemplos, tendo sua configuração interna e previsão atualizada com base nos dados de treinamento fornecidos. O principal objetivo desse processo é capacitar o modelo a detectar padrões complexos e informações relevantes nos dados de treinamento. Ao assimilar tais padrões, a rede neural pode realizar inferências ou previsões em novos dados que não foram previamente apresentados a ela.

O modelo *Multi Layer Perceptron*, MLP, conecta camadas compostas pelos neurônios artificiais entre si. Esse encadeamento de neurônios, acrescido do uso de funções de ativação não lineares, permite a esse modelo lidar com problemas de natureza não linear [15].

Normalmente, uma MLP simples, conforme Figura 2.2, é composta por uma camada de entrada, uma camada oculta e uma de saída, onde a camada oculta e de saída são compostas por diferentes números de neurônios totalmente interconectados. Para obter a saída desejada do modelo, um processo iterativo de ajustes dos pesos W_i de cada neurônio da rede é realizado durante o treinamento da rede.

Dado vetor de entrada X, o valor da saída $Y_{previsto}$, comparado ao valor Y_{real}, é avaliada segundo uma função de custo que quantifica numericamente quão correta é a previsão do modelo naquele ponto do treinamento. Esta função deve ser selecionada de acordo com o problema trabalhado, sendo diretamente ligada à natureza do problema e dos dados, além de ser maximizada ou minimizada de acordo com o processo utilizado.

Esse processo de minimização ou maximização se dá por algoritmos de otimização, comumente em *Deep Learning*, por exemplo, pode ser citado o Gradiente Descendente Estocástico (SGD), Adam ou Adagrad. Assim como a função de custo, a escolha do algoritmo de otimização pode afetar a qualidade

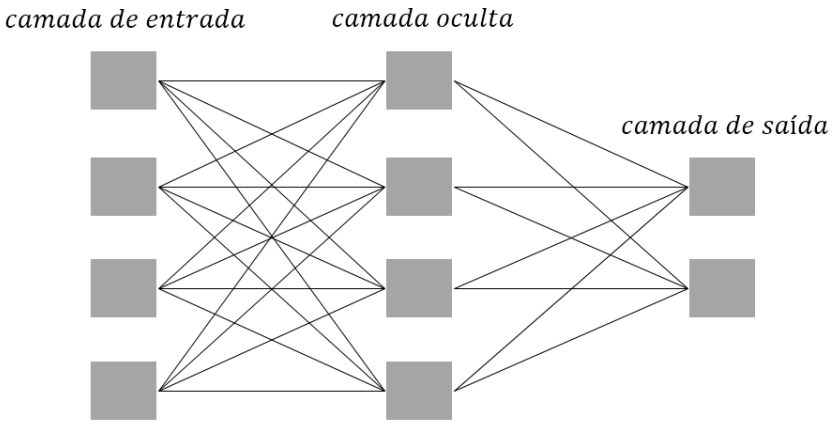

Figura 2.2: Exemplo de uma MLP simples com uma camada oculta

e velocidade do treinamento, visto que cada algoritmo atualiza os pesos do modelo de acordo com os gradientes da função de custo de forma diferente.

Deep Learning pode ser classificado como uma extensão do conceito de redes neurais [16]. Diferente das MLPs, com apenas algumas camadas ocultas, redes neurais profundas possuem um número maior de camadas, onde essa complexidade permite a modelos deste tipo extrair características de alto nível dos dados de entrada, exemplificado na Figura 2.3. Assim, características relevantes podem ser extraídas diretamente dos dados sem interferência humana ou etapas adicionais, processo comumente descrito como ponta a ponta (*end-to-end*).

Para realizar uma classificação de maçãs e laranjas, por exemplo, características básicas como formato, cor e tamanho poderiam ser determinadas por especialistas e obtidas por algoritmos adequados. Esses parâmetros formam os dados de entrada para uma MLP, que retorna a classificação do objeto em questão, isto é, laranja ou maçã. Em *Deep Learning*, uma única entrada de alto nível, como uma foto contendo uma maçã ou laranja, é oferecida à rede e cabe ao modelo detectar internamente características e informar a classificação da entrada.

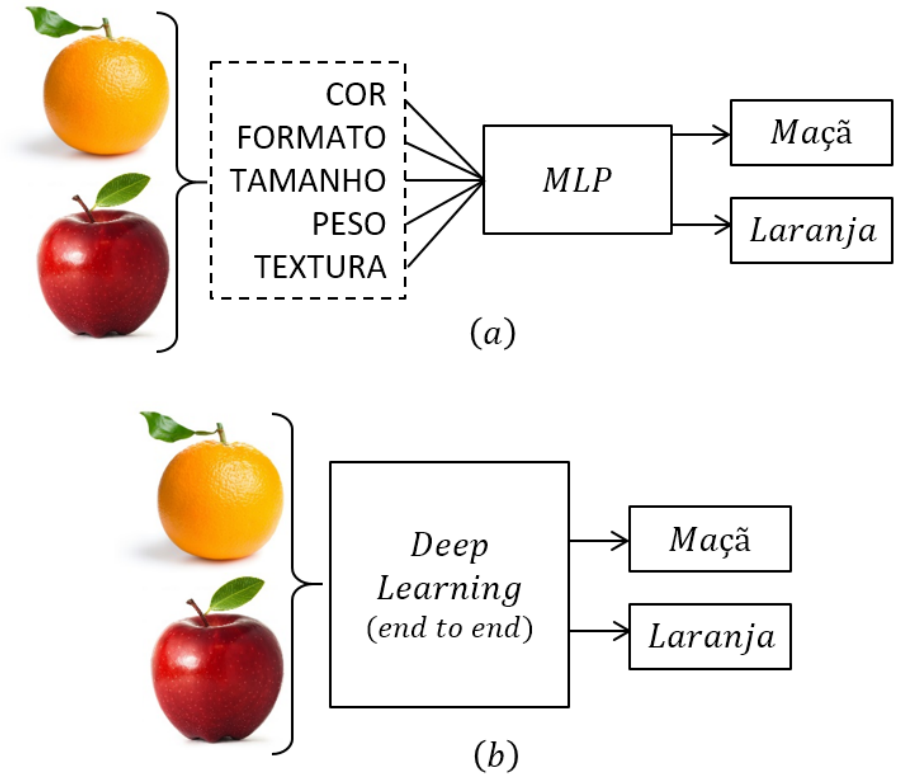

Figura 2.3: Comparação para classificação de maçãs e laranjas: (a) Modelagem clássica com extração prévia de características. (b) Conceito *end-to-end* aplicado em *Deep Learning*

2.2 Redes Neurais Convolucionais

Redes Neurais Convolucionais, CNN, surgem em *Deep Learning* como uma evolução das MLPs, fazendo uso da operação matemática de convolução. A convolução pode ser descrita como somatório do produto entre duas funções dada uma sobreposição, seguido do deslocamento entre elas. Graficamente, conforme Figura 2.4.

Formalmente a convolução discreta é denotada pelo simbolo $*$, descrita matematicamente por

$$(f * g)(x) = h(x) = \sum_{j=0}^{\mathrm{k}} f(j) \cdot g(k - j). \tag{2-3}$$

Essa operação possui propriedades de dependência espacial úteis para redes neurais, em especial para aplicações de visão computacional, onde características locais podem ser melhor detectadas. Em uma imagem, por exemplo, dado um pixel inicial observado, as chances de um pixel adjacente conter informações relevantes ao pixel inicial são grandes se comparadas a outros *pixels* distantes. Desta forma, o campo receptivo de um neurônio fica conectado a apenas uma parte local da imagem e é responsável por detectar características desta região menor [15], conforme visto na Figura 2.5.

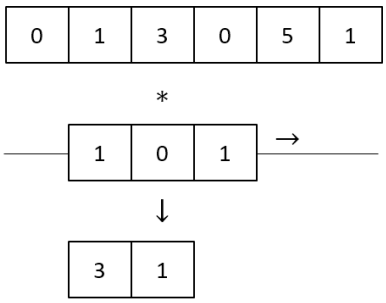

Figura 2.4: Visualização gráfica de uma convolução unidimensional

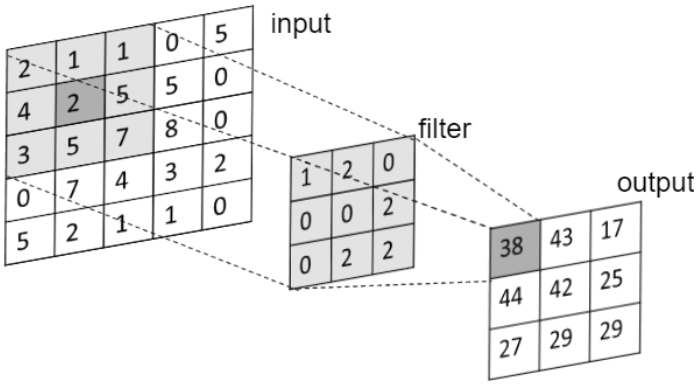

Figura 2.5: Visualização gráfica de uma convolução bidimensional
Fonte: Saifullahi et al. (2020)

A camada de amostragem (*pooling*), outra camada característica das CNNs, é usada para reduzir dimensionalidade dos dados, aumentar a robustez contra pequenas variações e combater o *overfitting* [15], problema onde a rede memoriza os dados de treino, passando a obter excelentes resultados apenas no conjunto de treino com fraco poder de generalização. A redução no tamanho é feita selecionando um valor representativo para cada região da imagem através de uma função de *pooling*, comumente máximo (*max-pooling*, exemplo Figura 2.6) ou média (*average-pooling*) [17].

Entretanto, a redução na dimensionalidade pode causar perda indesejada de informação e arquiteturas modernas, como a ResNet, utilizam ferramentas adicionais, como pular camadas para melhorar a capacidade de representação do modelo [18].

2.2.1 ResNet - Rede Neural Convolucional com Ativação Residual

A Rede Neural Convolucional com Ativação Residual, ResNet, é uma arquitetura proposta por Kaiming He et al. em 2016, projetada para resolver o problema de degradação da precisão em arquiteturas de redes profundas, em que ao adicionar mais camadas à rede neural a precisão da rede piora em vez de melhorar.

O modelo propõe a adição de conexões residuais, que permitem que a rede salte algumas camadas e transmita diretamente as informações da entrada para a saída de um bloco contendo uma ou mais camadas. Isso é feito por meio de uma conexão residual, isto é, adicionando a entrada do bloco convolucional à saída do mesmo [18], como no exemplo da Figura 2.7.

Essas conexões residuais permitem que a rede mantenha as informações originais da entrada em camadas mais profundas, melhorando a capacidade de representação da rede, prevenindo a degradação da precisão e o desaparecimento do gradiente em arquiteturas de redes profundas [18].

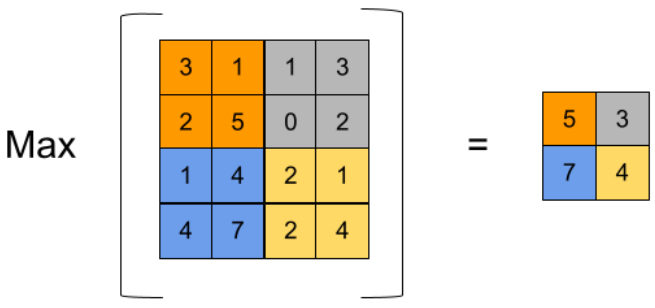

Figura 2.6: Visualização gráfica de uma camada *max pooling*

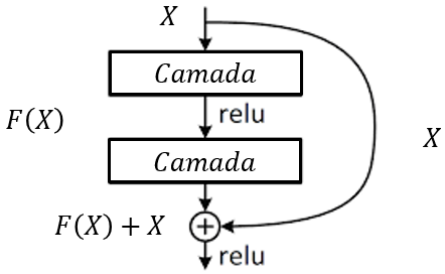

Figura 2.7: Bloco básico da topologia de rede ResNet

2.2.2 Mobile Net

Redes Neurais modernas revolucionaram tarefas de processamento de imagem, como reconhecimento de objetos, categorização e classificação, e, com isso, possibilitam grande precisão e flexibilidade [19]. Entretanto, com o aumento da complexidade dos modelos, o custo do poder computacional requerido também se elevou, indo muito além do poder computacional muitas vezes disponível em dispositivos embarcados ou móveis [20].

A MobileNet é uma arquitetura de rede neural convolucional otimizada para dispositivos com recursos limitados, utiliza filtros menores e técnicas como *Depthwise Separable Convolution* para reduzir o número de parâmetros e aumentar a eficiência, em termos de computação e uso de memória [19] [20]. *Depthwise Separable Convolution* separa a convolução espacial em duas etapas, a convolução profunda e a convolução de ponto, conforme Figura 2.8.

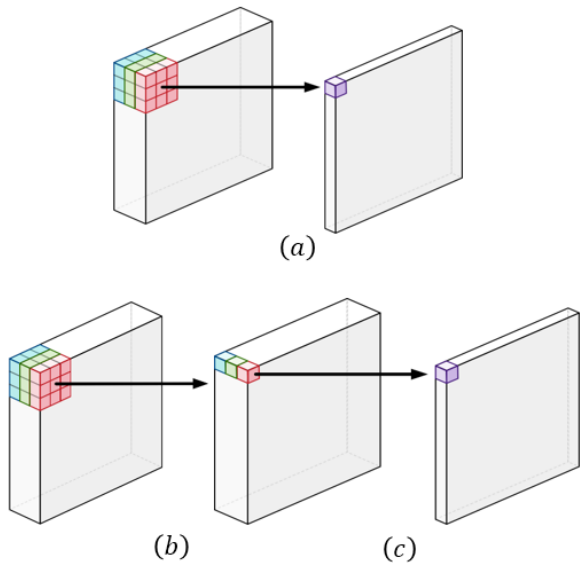

Figura 2.8: Comparação entre a convolução padrão e a proposta pela Mobile Net. (a) Convolução padrão; (b) convolução profunda; (c) convolução de ponto; (b)+(c) Depthwise Separable Convolution

A convolução profunda aplica um único filtro para cada canal de entrada separadamente, e a convolução de ponto combina as saídas da convolução profunda. Assim, tem-se o efeito de separar a convolução padrão em duas camadas, uma para filtrar e outra para combinar, enquanto a convolução padrão executa os mesmos passos de forma única. De forma contrária ao aparente, separar estas operações reduz a complexidade computacional do modelo e o número de parâmetros necessários, obtendo até 9 vezes menos operações computacionais ao usar um *kernel* 3x3, conforme demonstrado por Howard et al. 2017.

A segunda versão da MobileNet, proposta em 2019 por Sandler, et al., introduz o bloco *Bottleneck Depth Separable Convolution with Residuals*, exemplificado na Figura 2.9. Este bloco, primeiramente, expande a profundidade do tensor, aplica a *Depthwise Convolution* e o condensa para a mesma dimensão de entrada, através da camada de projeção, onde é conectado com um canal residual para facilitar a propagação do gradiente através das camadas [20], de forma similar ao proposto anteriormente pela ResNet.

2.2.3 DeepLab

Proposta por Liang-Chieh Chen et al. em 2016, a arquitetura de DeepLab propõe a técnica Atrous Spatial Pyramid Pooling (ASPP), exemplificado segundo Figura 2.10, que permite que a rede neural tenha uma melhor compreensão de detalhes finos na imagem sem perder contextos mais amplos [21].

As convoluções Atrous, também conhecidas como convoluções dilatadas, são uma variação da convolução convencional, que introduz lacunas entre os valores da matriz de filtro. Ao fazer isso, a convolução Atrous aumenta o campo receptivo da camada de convolução sem aumentar a quantidade de parâmetros, o que permite que a rede capture informações de contexto mais amplo [21]. A ASPP é composta de múltiplas camadas de convolução Atrous com diferentes taxas de amostragens aplicadas em paralelo.

A DeepLabV3+, vista no diagrama da Figura 2.11, é uma implementação

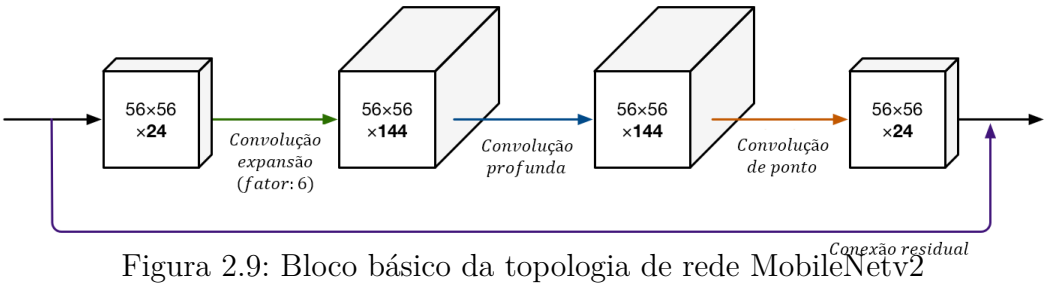

Figura 2.9: Bloco básico da topologia de rede MobileNetv2

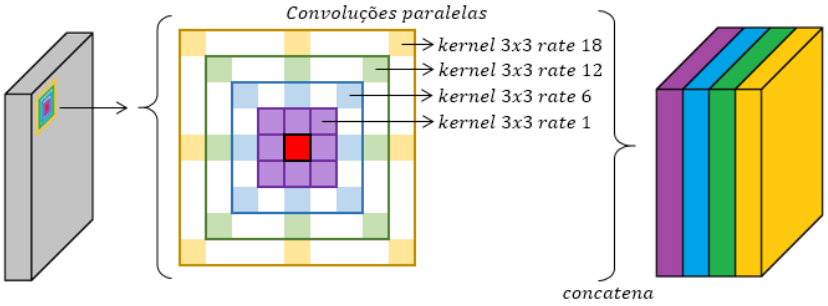

Figura 2.10: Visualização gráfica da camada Atrous Spatial Pyramid Pooling (ASPP). Para classificar o pixel central diversas dimensões de percepção são utilizadas ao variar a taxa de dilatação.

baseada no conceito codificador-decodificador, contendo, também, todas as alterações propostas pelas revisões anteriores da arquitetura [22] [23]. O módulo codificador extrai informação contextual aplicando convoluções Atrous em múltiplas taxas enquanto um decodificador simples refina a segmentação, principalmente ao longo das bordas dos objetos [23].

2.2.4 Transfer Learning

A técnica de transferência de aprendizado, *Transfer Learning*, pode ser caracterizada em usar conhecimento prévio de resolução de um problema para aplicar em outro similar. Em termos de *Deep Learning*, reutilizar camadas iniciais treinadas em um grande conjunto de dados disponível em treinamentos com dados limitados tende a obter melhor performance [24, Capítulo 11].

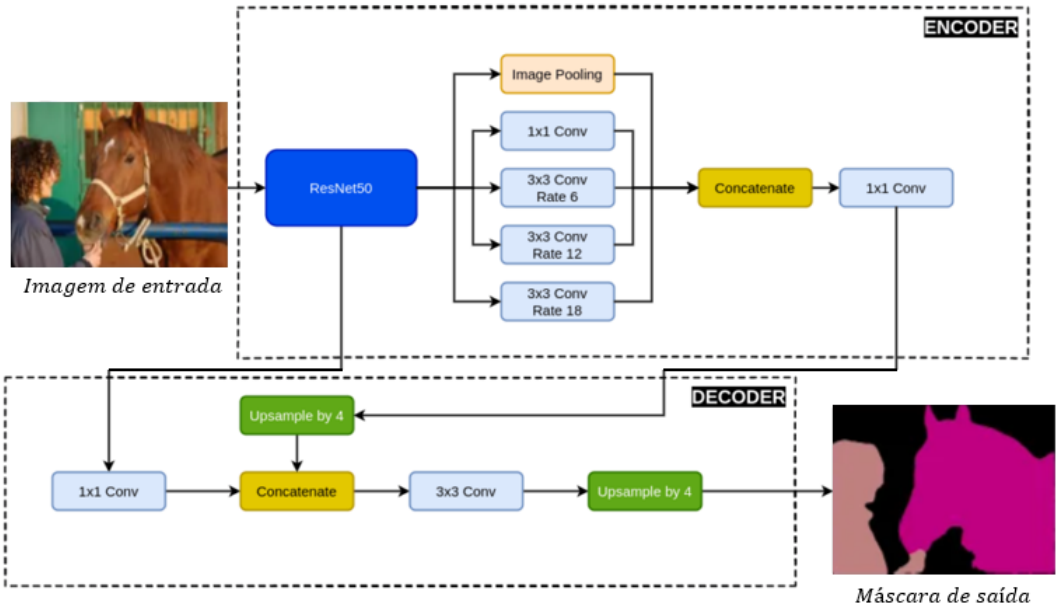

Figura 2.11: Implementação da rede DeepLabV3+ empregada, modelo ResNet50 utilizada como *backbone*

Grandes conjuntos de dados como o ImageNet [25], MNIST [26] ou CIFAR [27] podem ser utilizados para treinar redes e utilizá-las para extração de características em problemas diversos, nesses casos tais redes recebem o nome de *backbone*. Desta forma, por exemplo, uma rede ResNet de 50 camadas treinada com a ImageNet pode ser utilizada de *backbone* para realizar a extração de características iniciais do problema alvo.

2.3 Treinamento com tarefa auxiliar

O treinamento multi-tarefa, é uma técnica de treinamento em que uma rede é treinada para realizar múltiplas tarefas simultaneamente, isto é, a partir de uma entrada duas ou mais saídas são obtidas, conforme Figura 2.12. A utilização de modelos muti-tarefa se aproveita da correlação entre duas atividades relacionadas, detecção de objetos e segmentação semântica, por exemplo, para melhorar a precisão do resultado e reduzir tempos de treinamento e inferência [28].

Uma variação é o treinamento com tarefa auxiliar, neste caso a tarefa auxiliar serve apenas para o modelo obter uma representação mais rica e robusta do problema em questão e facilitar a execução da tarefa principal, como exemplificado na Figura 2.12. Assim, é de grande importância a escolha de uma tarefa com aprendizado simples, fácil obtenção e que suporte bem a tarefa principal durante a fase de treinamento [28] [29]. A tarefa auxiliar

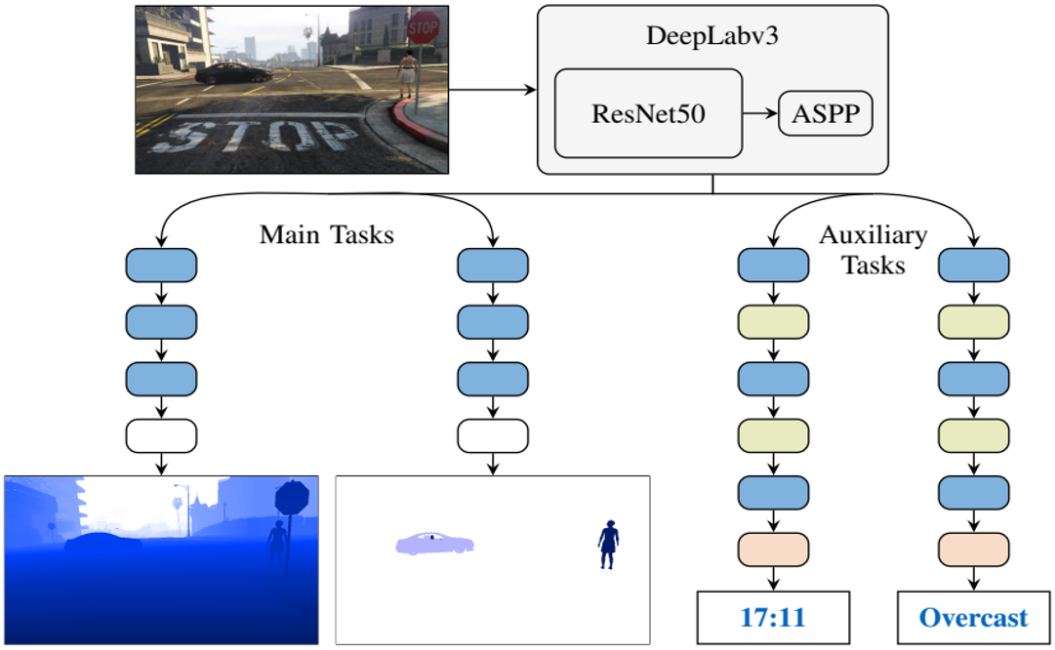

Figura 2.12: Exemplo de tarefa auxiliar para determinar condições gerais para um ambiente

Fonte: Liebel e Körner (2018)

pode ser vista, também, como uma fonte de restrição do problema, reduzindo variação ao regular condições explícitas ao modelo. No entanto, deve-se tomar cuidado pois muitas restrições podem prejudicar a generalização e aprendizado.

De forma a tornar possível o treinamento multi-tarefa, a utilização de uma função de custo também multi-tarefa deve ser aplicada, comparando a saída prevista de cada tarefa com seu respectivo valor real. Essa função de custo é uma combinação das funções de custo individual e, a partir de diferenças no comportamento das tarefas, o balanceamento dessa composição é importante na execução do treinamento [28].

Para poucas tarefas auxiliares ou tarefas simples, esse balanceamento pode ser feito manualmente de forma experimental, porém, conforme Kendall et al [29], o peso para cada uma das funções de custo pode ser incluído nos parâmetros de aprendizado do modelo, entretanto modificar a combinação da função de custo deve ser realizada para evitar soluções triviais.

2.4 Trabalhos relacionados

Robôs agrícolas autônomos, Figura 2.13, podem aumentar a produtividade e eficiência do campo ao executar diversas atividades, em especial aquelas simples, porém longas, como monitoramento de crescimento [11] [30] [31], controle de pestes [3] [32] [33] ou mesmo coleta de amostras [9]. Atualmente, uma solução popular para o deslocamento no campo são sistemas RTK-GNSS, Global Navigation Satellite System com Real time Kinematic [3] [9] [34]. Porém, essa abordagem é suscetível a qualidade da recepção GNSS [11], além de ser custosa [9] [10] e poder demandar precisos mapeamentos devido a evolução do campo durante o plantio [11] [14] bem como desafios de implementação em escala.

A navegação é uma das tarefas mais básicas e complexas para um robô móvel [36] e, hoje, pode ser observada uma tendência de soluções on-

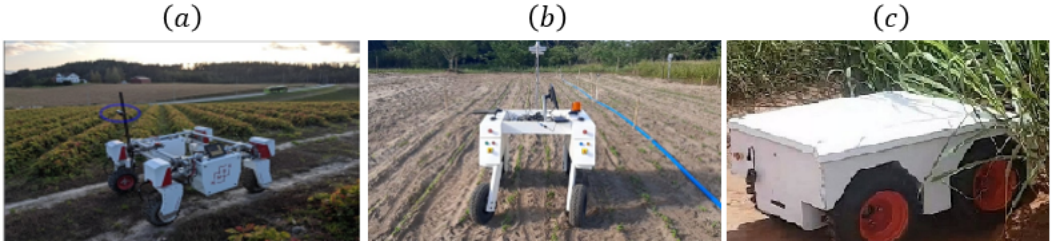

Figura 2.13: Exemplos de robôs agrícolas móveis utilizados nos trabalhos relacionados. (a) Thorvald II [6] [7]; (b) Soybot II [8] [10] [13]; (c) TIBA Robot [42]

board, isto é, executadas diretamente pelo robô em campo [11]. Dito isto, LiDAR e laser scanners são propostos em implementações para montar mapas topográficos [3] [37], ou mesmo como ferramenta suficiente para seguir uma fileira estruturada no campo [38]. Soluções baseadas em nuvem de pontos também foram propostas [39] e, até mesmo, soluções diferentes utilizando um lidar-2D apontado ao solo do campo para montar um mapa de pontos 3D durante o deslocamento do robô [40].

Outra abordagem popular é a utilização de câmeras e processos de segmentação para detecção das fileiras de plantação [11] [34]. Métodos de janelas deslizantes [9] [14] são comumente combinados com diferentes ferramentas para obtenção de máscaras [10] a partir de uma imagem para obter as fileiras de plantação e, com isso, construir referências para as fileiras e caminhos percorríveis.

O uso de Deep Learning para segmentação já foi explorado [9] [41] [42] com resultados promissores, apesar de desafios observados devido a condições de iluminação adversas e irregularidades do campo.

Ponnambalam [9] utiliza a topologia Seg-Net para gerar máscaras de segmentação e, a partir destas máscaras, através da técnica de janelas deslizantes, extrai as linhas suportes dos espaços entre as fileiras de plantação. No campo alvo deste estudo, o robô deve percorrer acima de uma das fileiras, assim as duas fileiras suportes são combinadas para obter a linha utilizada no controle de trajetória.

Silva [41] emprega outra topologia de rede, a U-Net, para gerar uma máscara contendo diversas linhas e seleciona a correta ao acumular, verticalmente, todos os *pixels* da máscara. Desta forma, uma distribuição é obtida ao longo do eixo horizontal da imagem, onde a linha com maior verticalidade é escolhida e processada para utilização posterior no controle da trajetória do robô.

Xaud [42] experimenta, com a utilização de câmeras térmicas, a detecção do espaço entre as fileiras através de uma segmentação que selecione apenas partes com temperaturas específicas da imagem. Esta segmentação gera uma máscara que tem os contornos extraídos e utilizados para aproximar duas linhas, as quais determinam a linha resultante final. Ao longo dos experimentos, uma CNN foi empregada para gerar as mesmas máscaras previamente obtidas com bons resultados.

Destes exemplos, pode ser constatado que utilizar segmentação por redes neurais para detectar fileiras de plantação, ou espaços entre as fileiras, já é possível. Entretanto, os trabalhos observados se atém a utilizar a máscara obtida para inferir a linha que direciona a trajetória, enquanto, na abordagem proposta, temos por interesse obter a linha diretamente através de uma rede

neural convolucional, com a máscara empregada como tarefa auxiliar.

Devido a grande quantidade de dados necessários para testes destas abordagens e, para evitar danos em testes a campos de plantio, o uso de simulações também se faz comum em diversos trabalhos, conforme Figura 2.14. Simulações permitem, ainda, uma avaliação em condição alvo de fácil reprodutibilidade, principalmente ao ser reconfigurável.

No presente capítulo alguns conceitos relevantes ao entendimento do trabalho puderam ser apresentados, como uma introdução ao conceito de redes neurais artificiais e as principais topologias que serão exploradas ao longo do trabalho. O conceito de tarefa auxiliar também tem sua importância apresentada ao trazer mais informações para o modelo durante o aprendizado. Por fim, uma visão geral de trabalhos na área pôde dar uma visão dos desafios e oportunidades observados pelos demais autores do meio, em especial a utilização de redes neurais convolucionais para segmentação de imagens e posterior extração das linhas que guiam o robô ao longo do campo.

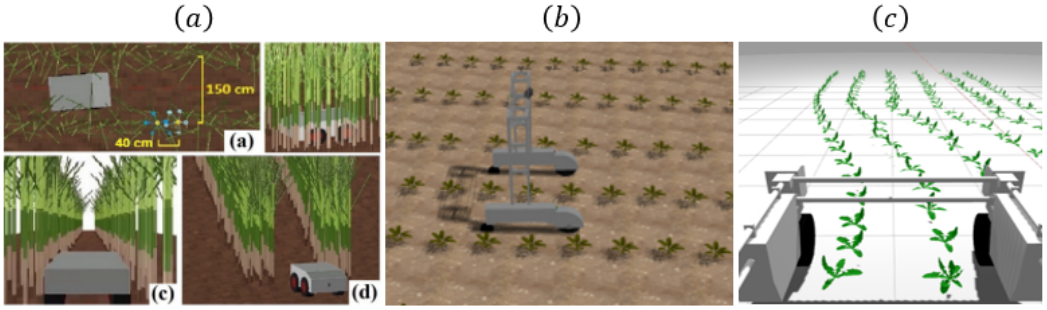

Figura 2.14: Exemplos de simulações realizadas nos trabalhos citados: Xaud, Leite e From (2019) à esquerda; Barbosa (2022) ao centro; Ahmadi, Halstead e McCool (2022) à direita

3
Metodologia Desenvolvida

Neste capítulo ferramentas utilizadas e desenvolvidas para a execução deste trabalho serão apresentadas. A modelagem proposta para a obtenção da linha necessária ao controle do robô é descrita, assim como a construção dos conjuntos de dados necessários, também conhecidos por *Dataset*, para o treinamento dos modelos propostos. O modelo proposto, suas variações, e o ambiente simulado também serão caracterizados, assim como o modelo simplificado do robô utilizado durante a simulação.

3.1
Soybot Simulado

3.1.1
Características Gerais

O robô simulado contém duas rodas frontais responsáveis pela tração, e duas rodas castores. As rodas dianteiras compõem um sistema de tração diferencial, de forma análoga ao robô real. Cada roda possui 10 centímetros de banda de rolagem, 20 centímetros de diâmetro e 1,0 metro de espaço entre o centro das rodas direita e esquerda, espaço necessário para percorrer acima da plantação.

De forma similar ao robô real, o simulado possui quatro pernas independentes, com uma roda por perna e uma base central interligando a estrutura. A base central é localizada um metro acima do solo com dimensões de $1,1$ metro de largura e 80 centímetros de profundidade. As duas pernas traseiras são posicionadas a um pequeno ângulo com a vertical, de forma que o robô ocupe uma área de $1m^2$ no solo.

O peso do robô real completo é estimado em cerca de $150kg$, este valor foi distribuído no modelo com objetivo de obter características inerciais adequadas para o tamanho. A maior parte do peso do robô simulado foi atribuído à base central, onde ficam baterias e equipamentos do robô real, e o restante nas rodas motorizadas.

O algoritmo controlador da movimentação necessita de dois parâmetros para guiar o robô ao longo das fileiras de plantação. O primeiro parâmetro, X,

é a distância horizontal entre a linha estimada a partir da câmera do robô e uma referência vertical central, enquanto o segundo parâmetro θ, é o ângulo entre a linha estimada na imagem e a vertical.

Para os testes realizados, o robô simulado possui duas câmeras operando simultaneamente em posição central e alturas diferentes, conforme Figura 3.1. Devido ao diferente posicionamento das duas câmeras, a modelagem da linha é específico para cada configuração. As câmeras também foram testadas com uso de três diferentes resoluções, o que resultará em diferentes graus de fidelidade da imagem obtida, tempo de processamento e complexidade do modelo proposto.

3.1.2 Câmera Superior

A câmera superior encontra-se a $1,8m$ da superfície e apontada para o solo, de forma a capturar a área diretamente a frente do robô. A câmera produz 30 imagens por segundo com a resolução de 640×480 *pixels* e 8 bits de profundidade de cor por canal. Foi posicionada de forma a obter uma cobertura similar àquela obtida pela câmera superior do Soybot-II conforme Figura 3.2.

Nesta configuração, a distância entre a linha inferida e a vertical é retirada do topo da imagem e o ângulo com a vertical conforme a Figura 3.3. Esta visão tem por objetivo aproveitar a aparência estruturada das fileiras de plantação em contraste com o solo de plantio.

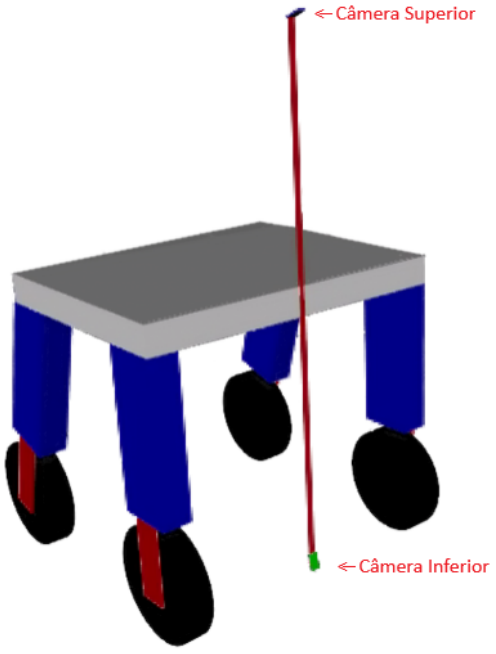

Figura 3.1: Modelo simplificado do robô Soybot-II utilizado durante as simulações com destaque para o posicionamento das câmeras

Figura 3.2: Comparação entre imagens simulada e real, respectivamente

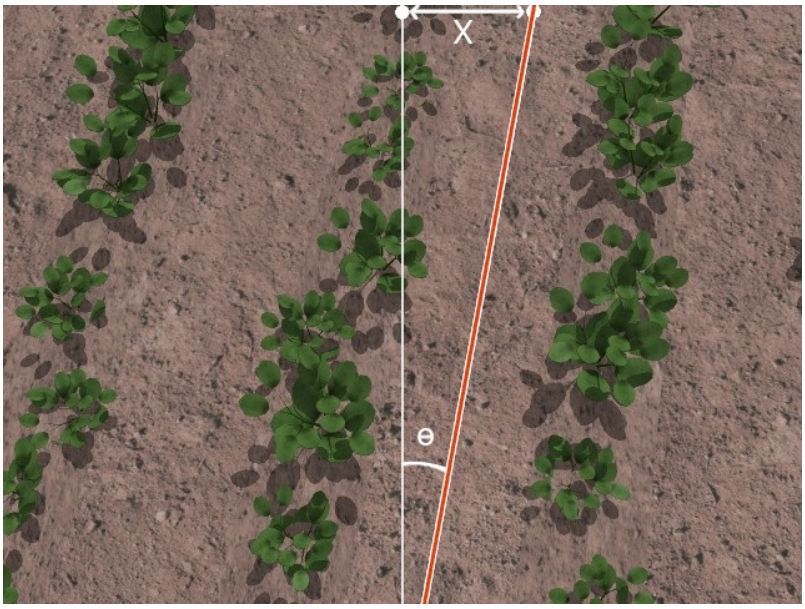

Figura 3.3: Características extraídas da linha superior

3.1.3 Câmera Inferior

A câmera inferior encontra-se a $0,15cm$ da superfície, apontada levemente acima da linha do horizonte de forma capturar uma visão em meio a fileira de plantação. Também produz 30 imagens por segundo com a resolução de 640×480 *pixels* e 8bits de profundidade de cor por canal.

Com o crescimento do plantio, na visão da câmera, paredes de folhas são formadas. Esse efeito visto pela câmera objetiva guiar o robô entre as fileiras. Já em condições de crescimento mais avançadas, túneis podem ser formados a ponto de ocluir o solo na visão da câmera superior. Com esta configuração de câmera, o solo continua presente, inclusive podendo ser diferenciado das fileiras de plantio, conforme Figura 3.4.

Nesta configuração, vista na Figura 3.4, a distância entre a linha inferida e a vertical é retirada da parte inferior da imagem, assim como o ângulo com a vertical.

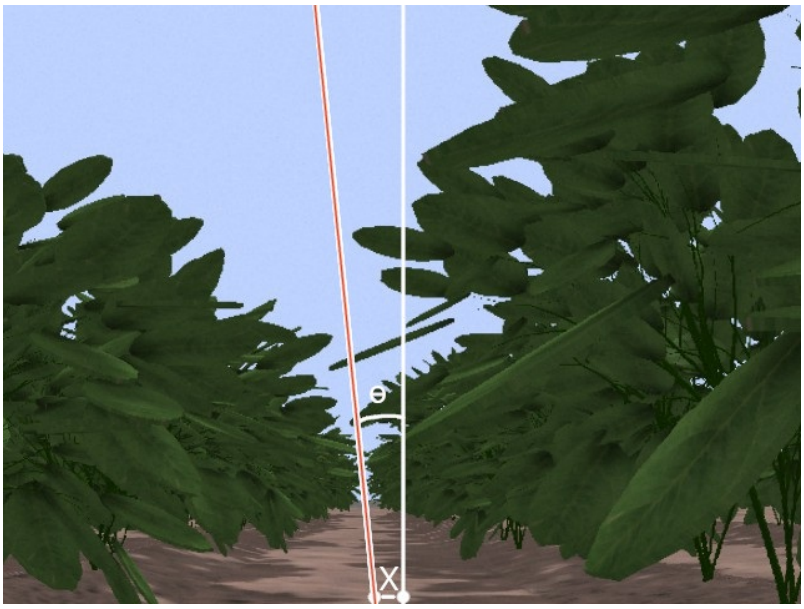

Figura 3.4: Características extraídas da linha inferior

3.2 Conjunto de Dados

Uma forma simples de definir uma reta é através de dois pontos. Então, para parametrizar a linha, dois valores X_i são inferidos de forma a gerar os seguintes pontos: $(X_0, 0)$ e $(X_1, 480)$. Ambos se encontram nas bordas superior e inferior da imagem, respectivamente.

É importante destacar que os valores X_0 e X_1 não são diretamente enviados ao controlador. Então, considerando W e H como a largura e altura da imagem, respectivamente, o deslocamento X pode ser obtido por

$$X = X_n - \frac{W}{2} \tag{3-1}$$

onde $n = 0$ para a câmera superior e $n = 1$ para a câmera inferior. E o ângulo θ pela expressão

$$\theta = arctg\left(\frac{X_1 - X_0}{H}\right). \tag{3-2}$$

Dessa forma, pode ser inferido que a uma linha centralizada e vertical na imagem retorna 0 para os valores X e θ. Essa abordagem permite compatibilidade com os algoritmos já utilizados e flexibilidade na modelagem da linha.

Cada item do *dataset* possui uma imagem RGB, uma máscara RGB codificada por cores, uma máscara categórica e os valores X_0 e X_1 que definem a linha. Das imagens existentes, provenientes de testes anteriores com o robô, as linhas foram manualmente anotadas e as máscaras construídas utilizando filtros de cores regulados para cada ambiente. As máscaras obtidas foram, então, vistoriadas para remoção de artefatos e erros, conforme a Figura 3.5, e possuem quatro categorias: fileira central ou alvo, plantas, solo e céu.

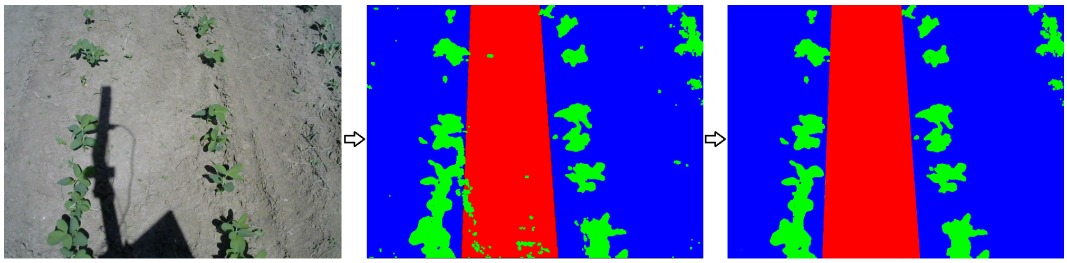

Figura 3.5: Processo de produção e limpeza das máscaras geradas

Para aumentar a quantidade de imagens disponíveis para o treinamento, imagens também foram extraídas a partir da simulação criada, descrita na próxima seção. De posse de ferramentas disponíveis em ROS, e maior controle sobre iluminação e texturas utilizadas, as máscaras e linhas podem ser extraídas diretamente sem nenhum pós processamento necessário.

O conjunto de dados utilizado para a câmera superior contém 50% de dados reais enquanto na câmera inferior se utilizou apenas dados simulados, devido a escassez de boas imagens. Para ambas as câmeras foi realizada a divisão de 70% dos dados para o treinamento, 20% para validação e 10% para testes, com o número de dados descrito na tabela 3.1.

3.3 Ambiente Simulado

Foi elaborado um campo de testes contendo cinco diferentes estágios de crescimento lado a lado das plantas, conforme Figura 3.6. O ambiente é executado no Simulador Gazebo 11, utilizando o motor gráfico Ogre 2.1 e motor físico DART, padrões do simulador. O robô simulado foi construído a partir de blocos básicos no formato Unified Robot Description Format (URDF), com interfaces gerenciadas pelos *plugins* de comunicação com o sistema ROS.

O campo de testes possui 30 fileiras de plantação, 6 fileiras por estágio, com 10 m de comprimento. Cada fileira possui plantas espaçadas de aproximadamente 25 cm entre si e um espaço entre fileiras de 50 cm.

Com o programa Blender, 23 modelos de plantas foram construídos baseados nas imagens reais existentes e distribuídos nos diferentes estágios de crescimento, segundo a tabela 3.2. Os cinco estágios de crescimento tornam

Tabela 3.1: Divisão dos subconjuntos de dados para treino, validação e testes para cada câmera

	Treino	Validação	Teste	Total
Câmera Superior Real	886	257	130	1273
Câmera Superior Simulada	914	261	133	1315
Câmera Inferior Simulada	1121	338	158	1572

Figura 3.6: Vista superior, no software Gazebo, do campo de testes elaborado contendo os 5 graus de crescimento

a identificação das fileiras, quando visto de cima, gradualmente mais difícil, enquanto a visão inferior converge para um túnel formado pelas folhas das plantas, conforme Figura 3.7

Ao posicionar cada planta no ambiente simulado, um ruído obtido é adicionado à posição da planta, a partir da curva normal, com desvio padrão de 3 cm e média 0 cm é adicionado a posição da planta. Além disso, cada planta possui 15% de chance de não ser inserida no ambiente simulado e é rotacionada para um ângulo aleatório entre 0 e 360 graus. Desta forma, diferentes campos de teste distintos são gerados de forma aleatória, porém com a estrutura desejada preservada.

O simulador permite, também, o ajuste das condições de iluminação da cena. Durante a coleta de imagens para a construção do conjunto de dados, diferentes posições para o sol foram utilizadas de forma a gerar diferentes situações de sombreamento e balanço de cores.

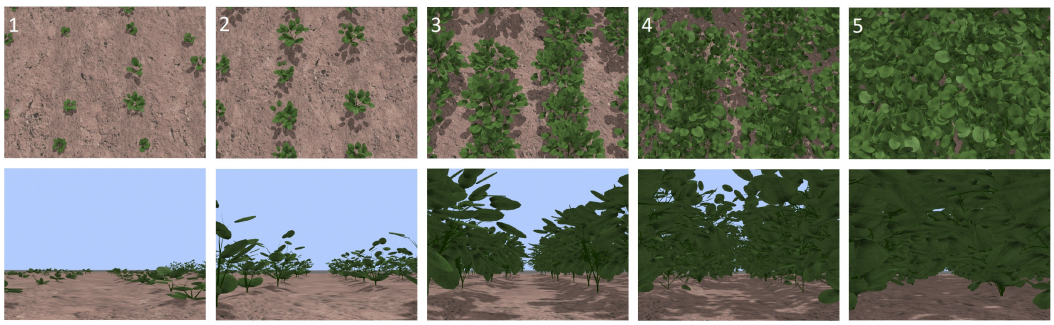

Figura 3.7: Visão do robô dos cinco estágios de crescimento do campo de teste simulado nas câmeras superior e inferior

Tabela 3.2: Número de modelos de plantas diferentes elaboradas para cada estágio de crescimento

Crescimento	Modelos
Estágio 1	8
Estágio 2	4
Estágio 3	3
Estágio 4	6
Estágio 5	2
Total	23

3.4 Modelos Propostos

Três topologias de redes distintas foram concebidas para avaliar diferentes circunstâncias durante a execução do presente trabalho. A rede DeepLabV3+ foi selecionada por ter demonstrado, em testes iniciais, boa capacidade de realização da tarefa de segmentação dos dados existentes.

O modelo base constitui na adição de uma segunda saída à rede, responsável por inferir a linha utilizada para o controle do robô. O modelo base utiliza a ResNet50 como *backbone* e possui elevado número de parâmetros ao utilizar resoluções mais altas. Desta forma, o segundo modelo avaliado utiliza a MobileNet em sua segunda versão como *backbone* para redução do número de parâmetros de treinamento e tempo de inferência observado. Por fim, a ultima variante desativa a saída de segmentação da DeepLabV3+ para avaliar o desempenho, no que se refere à utilização da tarefa auxiliar.

3.4.1 Modelo Base

A biblioteca Keras, da linguagem Python, possui exemplos de implementação de modelos populares de redes neurais. O modelo base proposto foi construído ao modificar o topologia apresentada pela biblioteca Keras, da linguagem Python, para a DeepLabV3+. A ResNet50 faz, nessa implementação, o papel de *backbone* para extração de características iniciais, tendo carregados pesos pré treinados no *dataset* ImageNet [12].

Em paralelo a saída segmentada da rede, a segunda saída, que é responsável por inferir a linha que controlará o robô, foi adicionada. É composta de uma camada *flatten*, duas camadas totalmente interconectadas, contendo 25 e 20 neurônios cada, e a camada de saída. A implementação da DeepLabV3+ selecionada toma o tensor da última camada convolucional da ResNet50, para aplicar a convolução Atrous. Este mesmo tensor é enviado para a inferência da linha e, assim, o modelo ganha a configuração vista na Figura 3.8.

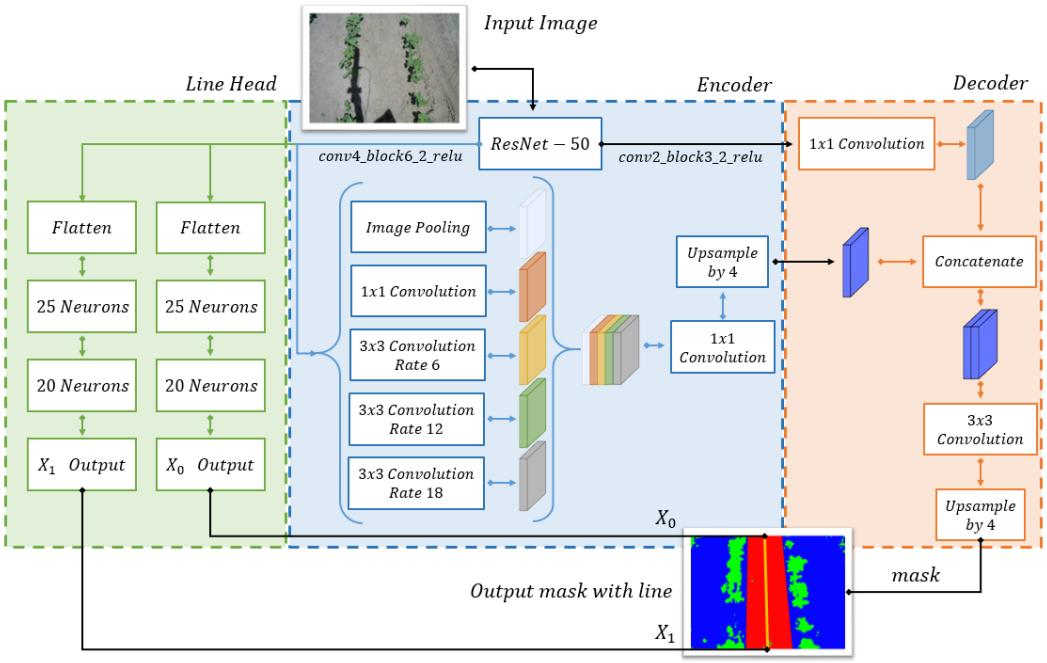

Figura 3.8: Representação gráfica da topologia do Modelo Base proposto

3.4.2 Modelo Reduzido

Como alternativa direta do modelo base, o modelo reduzido é proposto ao empregar a segunda versão da MobileNet em lugar da ResNet50 como *backbone* para extração de características iniciais. Assim como a Resnet50, a biblioteca possui uma versão da MobileNetV2, com pesos pré treinados, disponível.

Para aplicar a estrutura do modelo base a este novo *backbone*, a camada nomeada "*block_2_add*" foi enviada à convolução inicial do *decoder* da DeepLabV3+ enquanto a camada nomeada "*block_13_depthwise_relu*" foi utilizada na convolução Atrous e para inferência da linha. As camadas foram selecionadas de forma a manter dimensões de tensores compatíveis com os previamente utilizados pelo modelo base, assim mantendo similar topologia, conforme Figura 3.9.

Desta forma, uma significativa redução nos parâmetros de treinamento pôde ser obtida, conforme Tabela 3.3, para que, assim, tempos de inferência, treinamento e requisitos de memória e processamento possam ser aliviados.

Tabela 3.3: Número total de parâmetros treináveis dos modelos para cada resolução

	128×128	256×256	512×512
Modelo Base	12,6 mi	15,1 mi	24,9 mi
Modelo Reduzido	6,9 mi	8,4 mi	13,9 mi
Sem Tarefa Aux.	9,1 mi	11,6 mi	21,4 mi

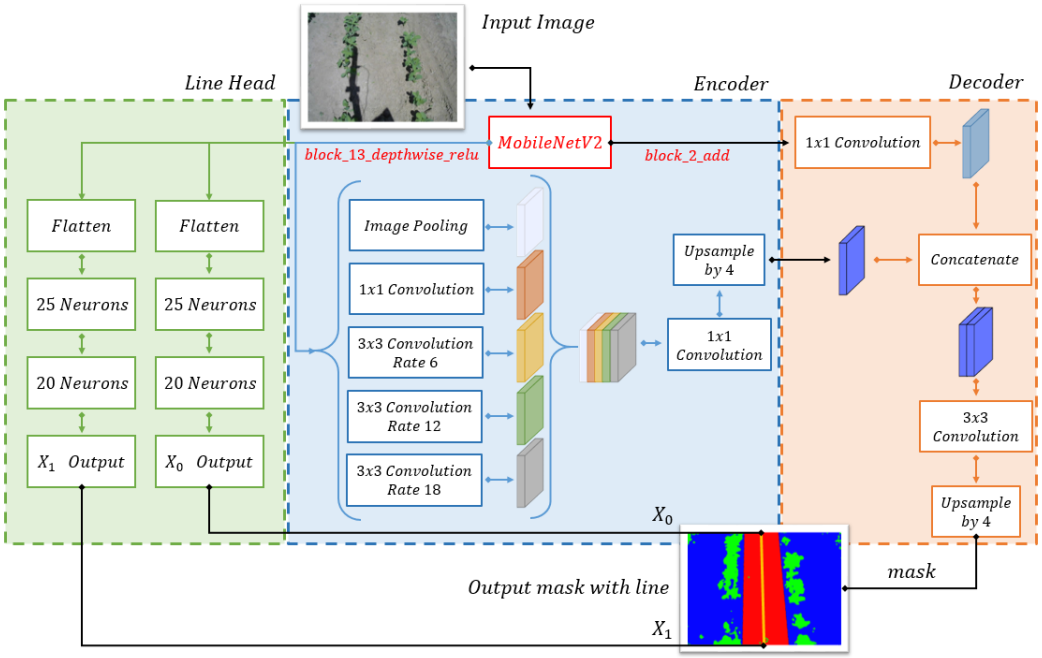

Figura 3.9: Alterações realizadas no Modelo Base com a troca do *Backbone*

3.4.3 Modelo Base sem tarefa auxiliar

O terceiro modelo explorado tem por objetivo avaliar o desempenho da saída responsável pela previsão da linha sem o auxílio da tarefa auxiliar. Nesta variação, tanto o *encoder* como o *decoder* da DeepLabV3+ são removidos, restando apenas a ResNet50 utilizada para extração inicial de características e as camadas adicionais descritas no modelo base.

A camada da ResNet50 utilizada como saída também é mantida, assim como no modelo base, a fim de manter a similaridade entre os modelos, neste modo nenhuma máscara é gerada. A Figura 3.10 demonstra a estrutura existente comparada ao modelo base.

Evoluindo sobre os conceitos teóricos apresentados no capítulo 2, o capítulo 3 apresenta as configurações de redes propostas pelo trabalho. Três topologias de rede neural que terão seu desempenho comparados foram apresentadas, elucidando suas diferenças e variações nas configurações empregadas. Neste capítulo também foi apresentado a estrutura dos dados de treinamento utilizados, assim como a modelagem da linha responsável por guiar o robô. A previsão desta linha é a tarefa fundamental realizada pelo sistema proposto, que será testado nos capítulos seguintes no campo inicialmente aqui apresentado.

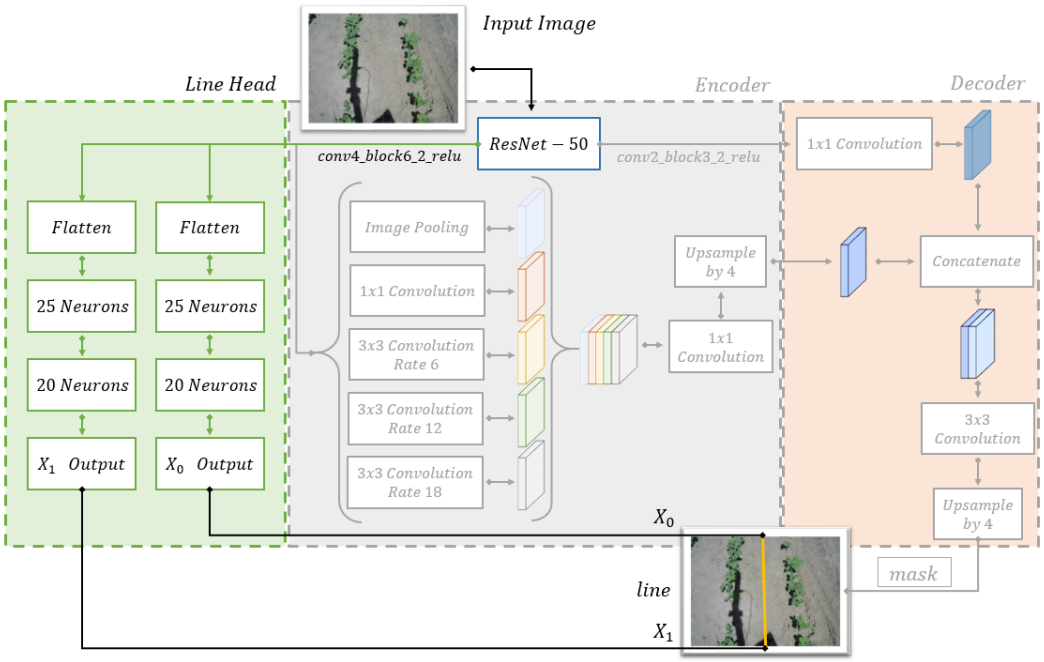

Figura 3.10: Parte desativada do Modelo Base com a remoção da tarefa auxiliar

4
Configuração Experimental

Neste capítulo, parâmetros de treinamento dos modelos, simulação e controle serão apresentados, e, também, a topologia do sistema ROS. A metodologia de avaliação utilizada na performance dos modelos é discutida, assim como as comparações a serem realizadas. Por fim, também são apresentados os parâmetros utilizados para os testes em campo.

4.1
Hardware Utilizado

O módulo Jetson AGX Orin Developer Kit, da Nvidia, é o computador principal do robô e utilizado, inclusive, nas simulações. Este SOM, System on Module, contém 12 núcleos Arm Cortex-A78AE para processamento geral e uma GPU baseada na microarquitetura Ampere da Nvidia com 2048 núcleos CUDA. O módulo possui 64 Gigabytes de memória unificada, onde CPU e GPU compartilham o recurso, operando a 3200MHz. Possui, além disso, 64 Gigabytes de armazenamento interno não volátil.

O robô Soybot-II, na configuração simulada e utilizada nos testes controlados, possui uma câmera RGB Logitech C270 que pode ser posicionada conforme a necessidade. Durante os testes realizados com o robô Thorvald-II foi utilizado um Laptop Lenovo Gaming 3. O laptop possui processador AMD Ryzen 7 5800H com 8 núcleos x86-64 para processamento geral e uma GPU RTX3060, baseada na arquitetura Ampere da Nvidia, com 3840 núcleos CUDA.

4.2
Software Utilizado

A versão mais recente disponível da primeira edição do *framework* Robot Operating System, ROS, foi utilizada no momento da escrita do presente trabalho, Noetic Ninjemys, sob o sistema operacional Linux Ubuntu 20.04 LTS.

Python, na versão 3.8.10, foi a principal linguagem empregada. Os dados foram processados com as bibliotecas Numpy, na versão 1.24.3, OpenCV, versão 4.2.0 e Pandas 2.0.2. O Treinamento foi realizado com a API Keras, disponível na biblioteca Tensorflow versão 2.7.0.

4.2.1 Robot Operating System - ROS

ROS é um conjunto de bibliotecas *open-source* desenvolvido para operar como um intermediário entre camadas superiores de código e o hardware do robô. Desta forma, sistemas podem ser construídos de forma modular, o que facilita o desenvolvimento e testes de aplicações robóticas.

O sistema ROS opera com uma topologia baseado em nós, onde cada nó pode escrever ou ler mensagens de um tópico. Na Figura 4.1, o diagrama do nó proposto pode ser observado, onde a imagem é recebida através do sistema de mensagens ROS e, internamente, realiza-se a preparação da imagem, a inferência do modelo e a aplicação do controle. Assim, envia novamente ao sistema ROS, através de outro tipo de mensagem, a velocidade desejada ao robô. Essa velocidade é recebida por outro nó, responsável por realizar a atuação nas rodas.

Com essa construção modular, qualquer parte do sistema pode ser facilmente substituída, simulada ou mesmo registrada e reproduzida. Comunicação entre nós também pode ser realizada através de rede local, possibilitando monitoramento, operação remota e cooperação entre diferentes computadores conectados, o que permite uma modularidade ainda maior.

4.2.2 Controle Linha Baseado em Imagem

O algoritmo de controle linha, Row Controller [43], previamente implementado [10] para a operação do robô Soybot-II foi mantido. Este controlador

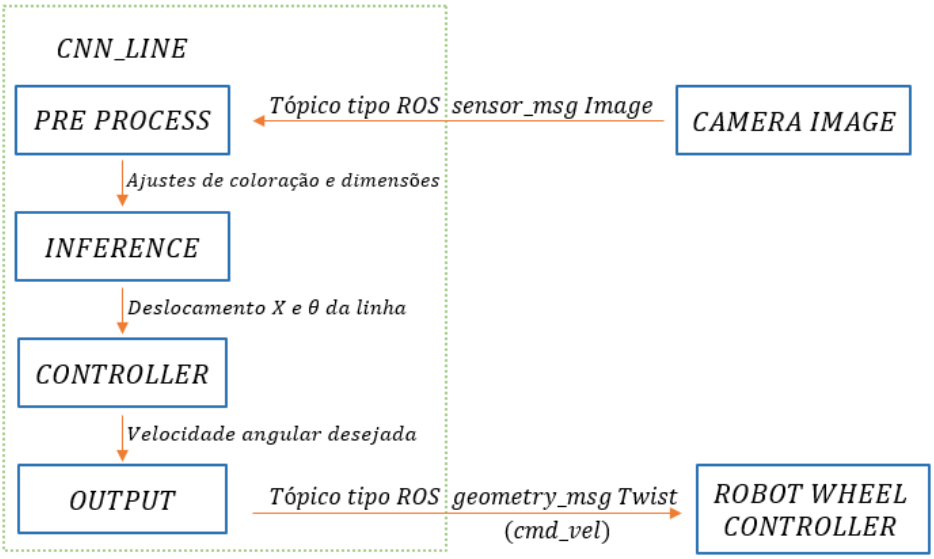

Figura 4.1: Construção simplificada do nó, mostrando as saídas e entradas necessárias a operação e etapas internas intermediárias

determina uma velocidade angular desejada ω_d dado as variáveis de entrada X e θ segundo a equação 4-1. Assumindo que o robô se desloca com velocidade linear constante v_d, a velocidade angular desejada ω_d é

$$\omega_d = -B_r^{\dagger} \left(\begin{bmatrix} \lambda_x X \\ \lambda_\theta \theta \end{bmatrix} + A_r v_d \right), \tag{4-1}$$

onde λ_x e λ_θ são os ganhos de controle, B_r e A_r são matrizes obtidas para a modelagem da câmera e o operador † denota a pseudo inversa de Moore-Penrose.

4.3 Configurações de Treinamento

Devido a existência de múltiplas saídas para os modelos utilizados, uma função de custo personalizada se fez necessária, pois cada saída do modelo possuí características distintas.

As saídas responsáveis pela linha, X_0 e X_1, são variáveis numéricas contínuas. Desta forma, a raiz do erro quadrático médio, RMSE, foi a métrica selecionada para ambos os valores, enquanto a *sparse categorical cross entropy* foi a métrica aplicada às máscaras para o cálculo da função de custo.

Assim, a função de custo é composta da soma das parcelas individuais de cada saída do modelo, a saber: X_0, X_1 e a máscara da tarefa auxiliar, conforme equação

$$Loss = CE_{\text{mask}} + rmse_{X_0} + rmse_{X_1}. \tag{4-2}$$

O treinamento foi realizado por até 300 épocas utilizando o otimizador ADAM com taxa de aprendizado de 10^{-5}. Os erros médios absolutos dos parâmetros X_0, para a câmera superior, e X_1, para a câmera inferior, são utilizados para avaliar a previsão da rede, mantendo os pesos de treinamento com melhor desempenho na etapa de validação, guardados para uso.

O treinamento pode, também, ser interrompido caso não haja nenhuma melhora registrada por cinquenta épocas, ou em caso de valores inválidos serem detectados durante o treinamento. Segundo as condições descritas, são mantidos os pesos com melhor desempenho registrado e o peso final do treinamento.

É registrado, para avaliação do desempenho do modelo, o valor da função de custo para todas as saídas, o erro absoluto médio, para os valores X_0 e X_1, e a acurácia da previsão da máscara.

As máscaras são codificadas com um valor inteiro para representar a categoria de cada pixel. Ao longo do trabalho, para melhor visualização, as categorias são convertidas em cores, segundo a tabela 4.1.

Tabela 4.1: Legenda para cada categoria utilizada para as máscaras na tarefa auxiliar de segmentação

Categoria	Cor	Valor RGB	Item
0	Vermelho	(255, 0, 0)	Caminho
1	Azul	(0, 0, 255)	Solo
2	Verde	(0, 255, 0)	Plantas
3	Magenta	(255, 0, 255)	Plano de fundo

Os valores X_0 e X_1, responsáveis por definir a linha referência do caminho a ser seguido, são normalizados no intervalo $[0, 1]$ para o aprendizado. Os valores mínimo e máximo admitidos são -640 e 1280, respectivamente, para tornar possível a obtenção de linhas oblíquas, que tem pontos de início e final nas laterais da imagem, conforme exemplo da Figura 4.2.

4.4 Configurações da Simulação

A simulação foi executada em um ambiente customizado no simulador Gazebo 11, com motor gráfico OGRE 2.1 e físico DART. Dentre as configurações padrão do simulador, a velocidade de simulação foi reduzida para a metade da velocidade real, assim assegurando uma simulação com melhor estabilidade da velocidade de execução, e a cor fundo atmosférico foi alterada para a cor código #82C8FF, para simular tonalidade de cores próximas das esperadas em condição de clima claro.

Durante os testes simulados, a posição solar foi definida de forma que a sombra do robô seja capaz de influenciar as imagens em um dos sentidos de deslocamento pelo campo e a intensidade da iluminação aumentada de 108

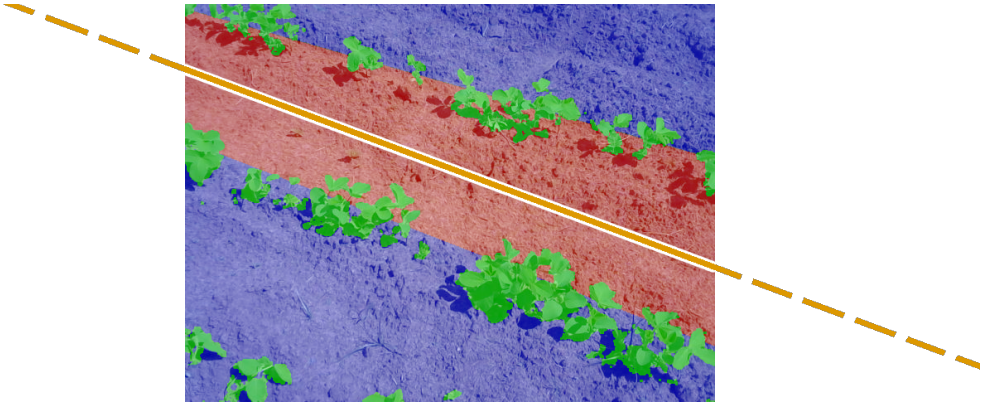

Figura 4.2: Exemplo de máscara com linha de grande angulação, situação não esperada durante a navegação em fileira porém possível em outras situações, como entrada de linha.

para 150.

O robô se desloca por todo o campo de testes em ambas as direções, para cada variação de configuração e modelos propostos. Cada travessia no campo é tratada como um caso individual, assim cada configuração avaliada percorre cinco vãos diferentes nos dois sentidos para cada nível de crescimento, parâmetro e modelo testado, e resulta em 30 testes por configuração.

A posição do robô é registrada, continuamente, na forma de um erro em relação a posição ótima esperada na fileira de plantação. É registrada, também, a saída do modelo, isto é X_0 e X_1, e uma saída teórica ideal obtida diretamente da simulação.

A posição inicial se dá 50 cm antes do início da fileira de plantação e 10 cm deslocado da linha de referência no espaço entre fileiras, escolhido de forma aleatória entre um deslocamento positivo ou negativo, conforme Figura 4.3. Também é inserido um ruído na atuação do robô, conforme a equação

$$error_{t+1} = (1 - \theta) \times error_t + normal\,(\mu, \sigma)\,, \tag{4-3}$$

com $\theta = 0.25$, $\mu = 0$ e $\sigma = 0.01$, de forma a simular possíveis diferenças de desgaste ou situações adversas que levem a um desbalanceamento da direção.

4.5 Configurações dos Testes em Ambiente Controlado

Um ambiente controlado foi elaborado com materiais fictícios disponíveis para execução de testes de campo preliminares. Para os testes de ambiente controlado, não serão produzidos novos modelos, devido a construção similar à simulação executada propositalmente. Entretanto, para os campos

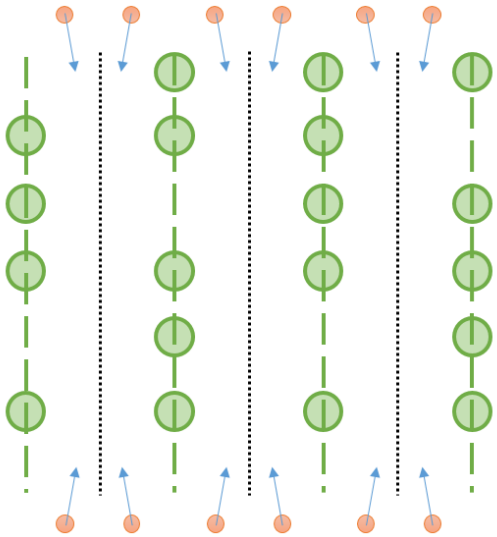

Figura 4.3: Diagrama superior da simulação durante os testes: em verde a fileira de plantação; em preto a linha teórica ideal para o robô; em vermelho as posições iniciais possíveis para cada fileira

reais, modelos serão treinados com conjuntos de dados retirados a partir dos ambientes alvo, visto que a simulação proposta não contempla ambientes com características similares aos disponibilizados.

O robô agrícola Soybot-II foi empregado para conduzir a verificação dos parâmetros de controle e comunicação com demais nós ROS, sendo testado com o melhor algoritmo observado na simulação. De posse de um ambiente de solo comum não pavimentado, majoritariamente plano, folhas de plantas foram posicionadas no solo dispondo um traçado curvilíneo, com leves irregularidades e com sombras projetadas no caminho do robô, visto na Figura 4.4. Estes testes validarão a funcionalidade do modelo proposto no robô real, avaliando a capacidade de seguir um traçado curvo e a estabilidade no trajeto.

O robô Thorvald-II, produzido pela Saga Robotics, em configuração adaptada a Polytunnels, foi disposto frente as fileiras de plantação simuladas, compostas por pequenas plantas dispostas no solo, Figura 4.5. Este teste visa avaliar a adaptabilidade da solução proposta a diferentes robôs com diferentes câmeras e sistemas de controle existentes. Assim, preparando o robô em questão para os testes de campo que serão realizados.

Figura 4.4: Ambiente utilizado para testes iniciais com o robô Soybot-II

Figura 4.5: Ambiente utilizado para testes iniciais com o robô Thorvald-II

4.6 Configurações dos Testes em Zona de Plantio

O robô Thorvald-II foi utilizado em dois ambientes reais disponibilizados pela Norwegian University of Life Sciences - NMBU.

No primeiro ambiente, uma câmera em posição superior deverá guiar o robô seguindo uma única fileira de plantação visível pela câmera, enquanto, no segundo, a configuração de câmera inferior será utilizada dado o tamanho da plantação alvo. Neste segundo ambiente, o robô deverá se guiar a partir da parede formada pelas árvores, na visão da câmera, devendo ser capaz de seguir toda a fileira proposta.

4.6.1 Polytunnel com Plantação de Morangos

O primeiro ambiente é composto por um túnel de plantação de morangos, para fins acadêmicos, com três fileiras de plantação em calhas suspensas, onde, no momento da execução do estudo, apenas duas calhas continham exemplares de plantas, vide Figura 4.6. O robô foi posicionado de forma a percorrer todas as fileiras de plantação em ambas as direções.

Para o treinamento do modelo utilizado nas travessias deste ambiente, imagens foram registradas com a câmera grande angular de um Smartphone Samsung Galaxy S20fe e processadas para possuírem a correta proporção, resolução e posicionamento. Desta maneira, um *dataset* de 200 imagens com linhas de referência e segmentadas, Figura 4.7, foi produzido contendo 20 imagens de teste, 40 de validação e 140 de treinamento.

Figura 4.6: Polytunnel com plantação de morangos utilizada para testes

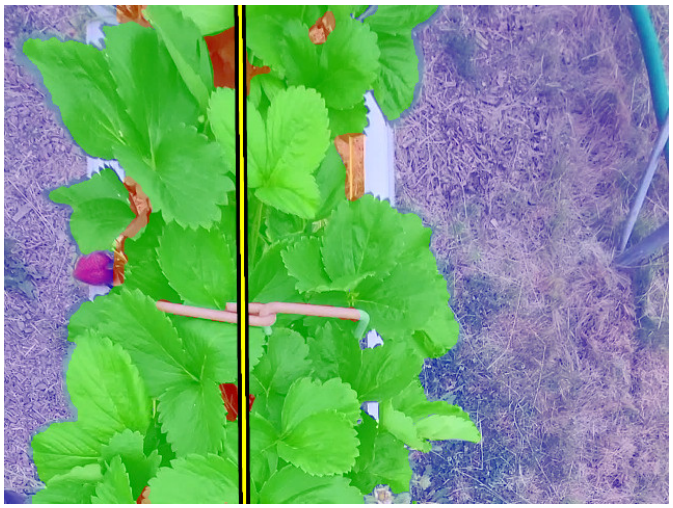

Figura 4.7: Exemplo do dataset produzido para o treinamento - Túnel de Morangos

Esta captura de imagens foi utilizada por indisponibilidade do acesso com o robô à plantação devido a condições climáticas e operacionais adversas. Durante a operação, a câmera AX-FHD Webcam Pro da Axtel foi utilizada e montada em frente ao robô apontada para baixo.

4.6.2 Pomar com relevo

O segundo ambiente utilizado foi um campo de plantio misto de frutas. Diversas fileiras de plantação existem com diferentes tipos de árvores e espaços vazios em meio a plantação. O solo também possui um alto grau de irregularidade e acentuado declive, o que desafia a capacidade do modelo de operar em ambientes de maior escala, observado na Figura 4.8.

Para execução das travessias neste ambiente, foram registradas imagens com três câmeras: RealSense D415i da Intel, Webcam Full-HD da Lenovo e pelo Smartphone Samsung. As imagens foram anotadas apenas com a informação da linha a ser seguida e desta forma, o modelo usa a configuração sem segmentação auxiliar, conforme Figura 4.9.

Figura 4.8: Pomar na NMBU utilizado para testes em ambiente de plantio

Figura 4.9: Exemplo do *dataset* produzido para o treinamento - Pomar

4.7 Avaliação dos Resultados

4.7.1 Metodologia de Avaliação dos Testes de Simulação

O desempenho dos modelos na simulação será avaliado utilizando o erro absoluto da posição do robô em relação à linha referência ao longo da fileira. A média das 30 travessias de cada configuração será utilizada para comparar o impacto da escolha de resolução, da utilização da tarefa auxiliar, da utilização do modelo reduzido e da escolha de câmera para cada estágio de crescimento.

Os valores obtidos para cada condição são apresentados em gráficos do tipo *box-plot*, o que permite uma melhor visualização do desvio padrão, média e existência de falhas completas, observadas como *outliers*.

A travessia é encerrada caso o robô perca completamente a direção e se afaste mais de dois metros da linha referência. Esta condição de teste foi implementada para tornar a execução mais simples e rápida, assumindo que caso o robô não seja capaz de seguir a fileira em que começou, também não será capaz de manter uma direção consistente. Desta forma, é permitido que um erro de troca de fileira ocorra uma vez, conforme Figura 4.10.

Valores de tempo de inferência, evolução da função de custo ao longo do treinamento e alta ocorrência de pontos discrepantes serão avaliadas.

4.7.2 Metodologia de Avaliação dos Testes em Ambientes Reais

Os testes com robôs reais serão avaliados através da comparação com a previsão manual da linha executada, sem ver a previsão do modelo. Assim os

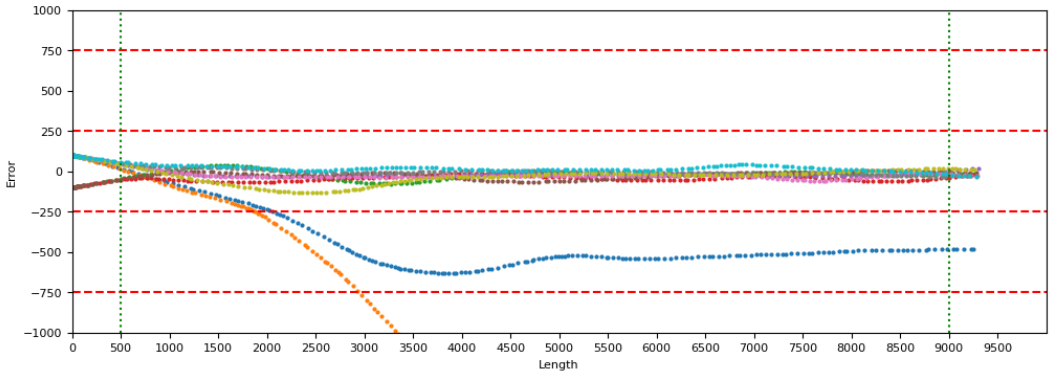

Figura 4.10: Visão superior da trajetória percorrida pelo robô nas 30 travessias de uma das configurações propostas, em uma das travessias houve o erro troca de fileira e em outra ocasião um erro completo

valores X e θ, enviados ao controlador, podem ser avaliados perante aos valores esperados na visão humana.

Para os pomares, uma avaliação cruzada também será conduzida. Três conjuntos de dados são empregados no treino, de um mesmo modelo, obtidos a partir de três câmeras distintas. Esta avaliação visa medir o impacto do uso de diferentes câmeras no processo de treino e inferência.

Todos os dados de treinamento serão extraídos e processados com as ferramentas desenvolvidas para o presente trabalho. Assim temos a seguinte lista de casos avaliados:

- Soybot-II simplificado com câmera superior e inferior testadas em ambiente computacional simulado, com pesos obtidos a partir de dados de simulação e testes prévios.
- Soybot-II em ambiente controlado, com pesos obtidos a partir de dados de simulação e testes prévios.
- Thorvald-II em ambiente controlado, com pesos obtidos a partir de dados de simulação e testes prévios.
- Thorvald-II no *Polytunnel* de morangos com câmera superior. Nesta situação, é utilizada a máscara de segmentação auxiliar e o robô percorre acima da fileira alvo.
- Thorvald-II nos pomares com câmera inferior. Nesta situação, não é utilizada a máscara de segmentação auxiliar, diferentes câmeras foram utilizadas na obtenção dos dados e o robô percorre entre as árvores.

Este capítulo teve por objetivo apresentar toda a configuração experimental utilizada, abordando, inicialmente, os recursos de *hardware* e *software* empregados, com menção especial ao *framework* utilizado, ROS. Os diversos

ambientes explorados neste estudo também puderam ser visualizados, assim como a metodologia de avaliação de desempenho que será empregada para cada ambiente no capítulo 5. Com isso, uma introdução aos ambientes reais e de controle pôde ocorrer, assim como um aprofundamento no funcionamento do ambiente simulado, com o objetivo de elucidar as estratégias e desafios observados para cada um destes.

5
Resultados

Neste capítulo, resultados obtidos são apresentados. Inicialmente, características observadas durante o treinamento dos modelos são discutidas junto a resultados prévios, provenientes de parâmetros rastreados durante o treinamento. Em seguida, é possível discutir o desempenho das variantes do modelo utilizando câmeras em posição superior e inferior, assim como a comparação entre as abordagens. Por último, testes realizados em ambiente controlado para testes iniciais do robô são abordados seguidos dos resultados obtidos em ambientes reais com robôs comerciais disponíveis.

5.1
Treinamento

Durante o treinamento, puderam ser observadas características que, posteriormente, se manifestariam nos testes executados.

A resolução utilizada no modelo teve impacto em sua performance, principalmente na câmera inferior, Tabela 5.1. Na câmera superior os modelos obtiveram resultados mais equilibrados, normalmente dentro do mesmo intervalo de confiança, Tabela 5.2. Entretanto, ainda é possível observar uma tendência na redução do valor mínimo da função de custo com o aumento da resolução,

Tabela 5.1: Valores mínimos médios da função de custo observados durante o treinamento dos modelos para câmera inferior no conjunto de validação

Configuração	Loss Mínimo Médio	Desvio Padrão	Intervalo de Confiança (95%)
128 Sem Aux	0.093	0.0085	(0.075, 0.111)
256 Sem Aux	0.052	0.0023	(0.034, 0.070)
512 Sem Aux	0.041	0.0027	(0.022, 0.059)
128 Base	0.100	0.0041	(0.082, 0.118)
256 Base	0.050	0.0052	(0.032, 0.068)
512 Base	0.032	0.0012	(0.014, 0.050)
128 Reduzido	0.134	0.0077	(0.116, 0.152)
256 Reduzido	0.091	0.0182	(0.073, 0.109)
512 Reduzido	0.061	0.0097	(0.043, 0.079)

Tabela 5.2: Valores mínimos médios da função de custo observados durante o treinamento dos modelos para câmera superior no conjunto de validação

Configuração	Loss Mínimo Médio	Desvio Padrão	Intervalo de Confiança (95%)
128 Sem Aux	0.044	0.0027	(0.026, 0.062)
256 Sem Aux	0.033	0.0049	(0.015, 0.051)
512 Sem Aux	0.021	0.0007	(0.003, 0.039)
128 Base	0.033	0.0022	(0.015, 0.051)
256 Base	0.025	0.0034	(0.007, 0.043)
512 Base	0.024	0.0078	(0.006, 0.042)
128 Reduzido	0.062	0.0043	(0.044, 0.080)
256 Reduzido	0.057	0.0065	(0.039, 0.076)
512 Reduzido	0.053	0.0129	(0.035, 0.071)

apesar de mais testes serem necessários para confirmar este padrão.

Comparando entre os tipos de rede propostos, é possível distinguir uma tendência de melhor desempenho do modelo base em relação ao modelo reduzido, enquanto o impacto da tarefa auxiliar se mostra pequeno na métrica observada. Porém, observando a evolução da função de custo durante as 300 épocas de um dos treinamentos, é possível notar que o modelo com tarefa auxiliar converge mais rapidamente, segundo Figura 5.1. Ainda, é possível notar grande instabilidade do treinamento ao utilizar baixa resolução.

Em geral, os modelos com câmera superior obtiveram menores, e mais consistentes, valores para a função de custo que os modelos com câmera inferior. Assim, é esperado que, para condições na qual a câmera superior seja adequada, mesmo os modelos com menor resolução se mostrem competitivos. Os piores desempenhos serão esperados dos modelos reduzido e, também, pouco impacto da tarefa auxiliar, apesar de potencialmente positivo.

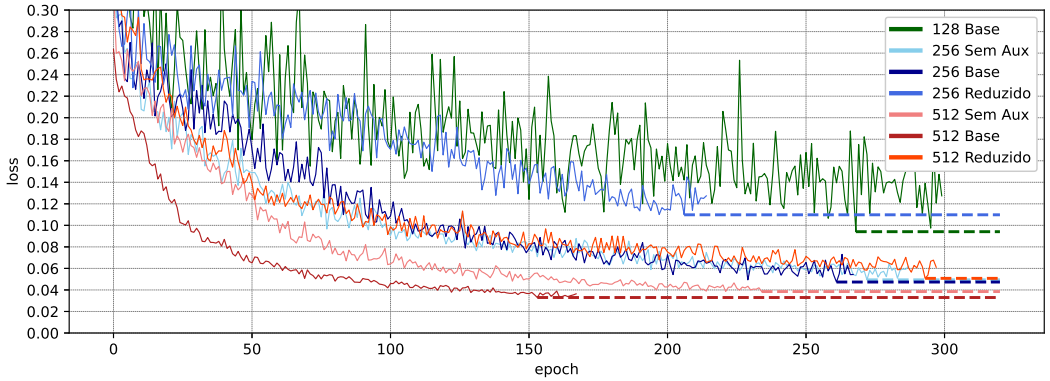

Figura 5.1: Evolução da função de custo ao longo do treinamento para a câmera inferior

5.1.1 Tempo de Inferência

O tempo de inferência também é um fator determinante para o desempenho do modelo, dado que atraso muito longo na atuação do controlador pode provocar instabilidade ou desempenho indesejável. O hardware utilizado para testes possui limitações na execução de atividades que utilizam, majoritariamente, um núcleo de processamento. Desta forma, soluções não paralelizáveis são penalizadas pelas limitações de poder computacional dos núcleos do processador ARM presentes na placa Nvidia Jetson AGX.

A abordagem anterior proposta foi concebida para operar em sistemas onde essa limitação não se dá com tamanha aparência, como em processadores x86. O desempenho da abordagem de múltiplas regiões de interesse com janelas deslizantes, *Multi-ROI*, anteriormente empregada, depende diretamente do número de *pixels* segmentados presentes na imagem. Assim, o tamanho da planta observada na imagem é capaz de impactar, de forma significativa, seu desempenho no sistema utilizado.

A Figura 5.2 compara o tempo de inferência obtido para cada variação do modelo proposto e o tempo de inferência para a abordagem anterior em cada estágio de crescimento. Os modelos baseados em *Deep Learning* tem seu tempo de inferência apenas decorrente das computações necessárias, assim são estáveis ao longo de todos os estágios de crescimento.

O tempo necessário para o processamento da linha de controle depende diretamente da velocidade de publicação das transformadas de coordenadas pelo sistema ROS. Aumentar a taxa de publicação aumenta o esforço geral do sistema durante a simulação, impactando os demais resultados.

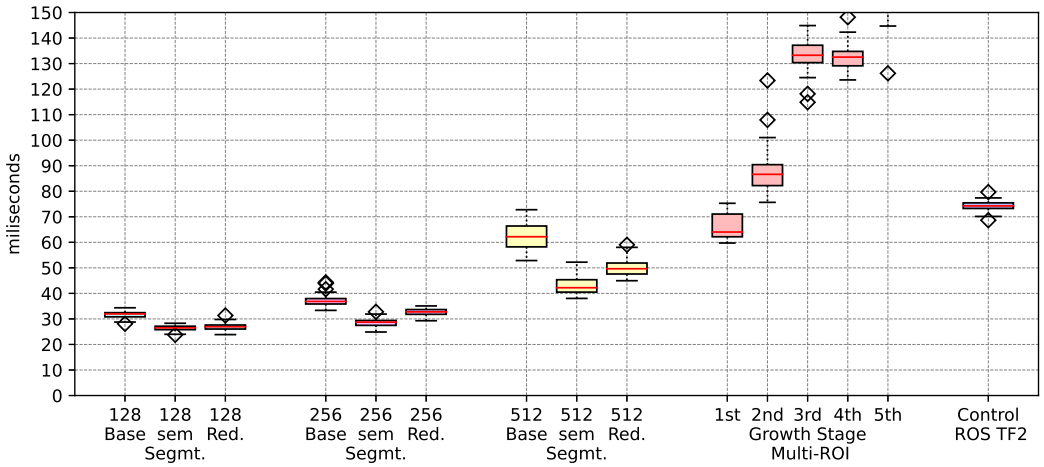

Figura 5.2: Tempo de processamento para cada configuração proposta em comparação com o algoritmo Multi-ROI nos cinco estágios de crescimento e linha controle

As simulações foram executadas com a taxa de publicação das transformadas de 50 hz e observadas latências no entorno de 70 ms. Com elevadas taxas de publicação, cerca de 150 hz, a latência observada foi, ainda, de aproximadamente 50 ms, não justificando o impacto negativo aos demais aspectos da simulação.

5.2 Resultados na Simulação

Para a avaliar o desempenho completo do sistema na simulação, pode ser utilizado o desvio, ou erro, absoluto médio em relação a linha referência ideal para o deslocamento do robô. Desta forma, o tempo de processamento, que impacta a latência entre recebimento da imagem e atuação, a qualidade da previsão da linha e sua consistência ao longo do tempo podem cumprir um fator no desempenho e avaliação do sistema. O desvio é dado conforme a Figura 5.3, obtido, de forma independente, para cada passagem pelo campo nas situações avaliadas.

5.2.1 Câmera superior

Para o campo no primeiro estágio de crescimento, na Figura 5.4, observa-se, mais uma vez, um desempenho inferior novamente do modelo reduzido, primariamente na resolução de 128 pixels, com melhora considerável a cada aumento de resolução. Um resultado menos consistente é observado com o aumento da resolução no modelo base e sem segmentação nas resoluções mais altas, impacto de possíveis problemas de treinamento e alto tempo de processamento.

O segundo estágio de crescimento apresenta melhores resultados de forma geral, vide Figura 5.5. Desempenhos equivalentes podem ser observados entre o modelo reduzido de maior resolução, o modelo base de menor resolução e

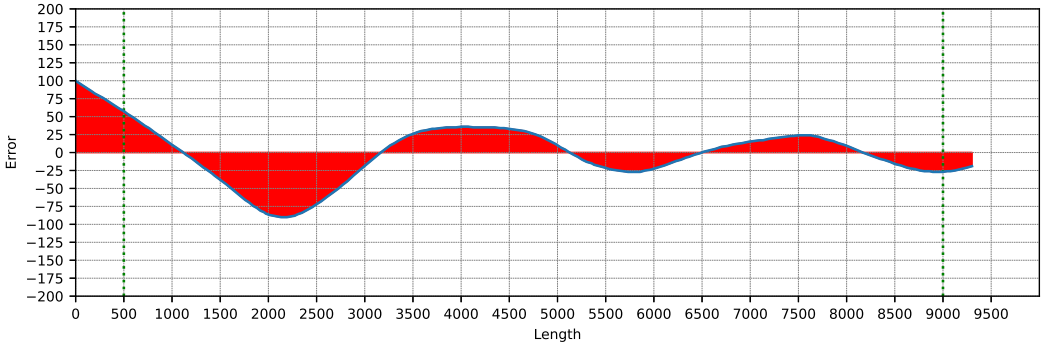

Figura 5.3: Desvio observado ao longo do trajeto percorrido utilizado para o cálculo do erro absoluto médio

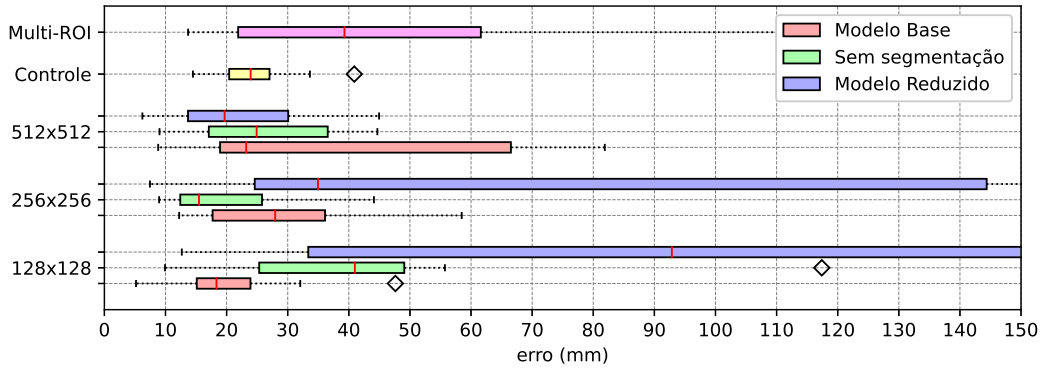

Figura 5.4: Erro absoluto médio observado para cada configuração com câmera superior no Campo 1

o modelo sem segmentação de resolução intermediária. Observa-se, inclusive, que estas configurações apresentaram desempenho superior a linha controle, que é, em teoria, matematicamente ideal. Entretanto, esta possui um alto tempo computacional, conforme observado na Figura 5.2, o que torna possível identificar o impacto positivo de um baixo tempo de processamento.

No terceiro campo, composto de plantas de tamanho intermediário, exemplificado na Figura 5.6, é visto um bom desempenho de quase todas as variações observadas. Em relação aos campos anteriores, uma melhora na consistência do modelo base para maiores resoluções é percebido, enquanto os demais modelos se comportam de forma similar ao observado anteriormente.

No quarto estágio de crescimento, um leve decaimento geral no desempenho ocorre, observado na Figura 5.7, comparado aos estágios anteriores. Dado o tamanho das plantas e a obstrução causada no solo, é esperado um desempenho superior da câmera inferior neste ponto.

Por fim, no último campo, é observado grande perda de desempenho para todos os modelos de câmera superior, na Figura 5.8, conforme o esperado. A alta variabilidade dos resultados, junto ao alto valor do erro, demonstra uma

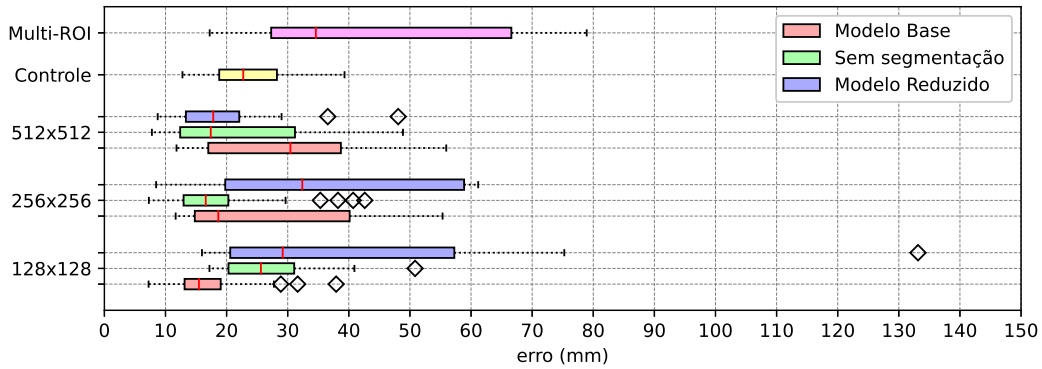

Figura 5.5: Erro absoluto médio observado para cada configuração com câmera superior no Campo 2

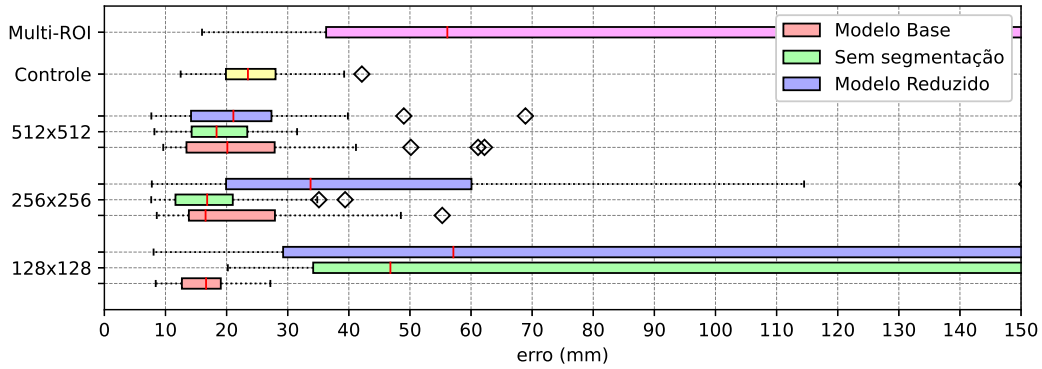

Figura 5.6: Erro absoluto médio observado para cada configuração com câmera superior no Campo 3

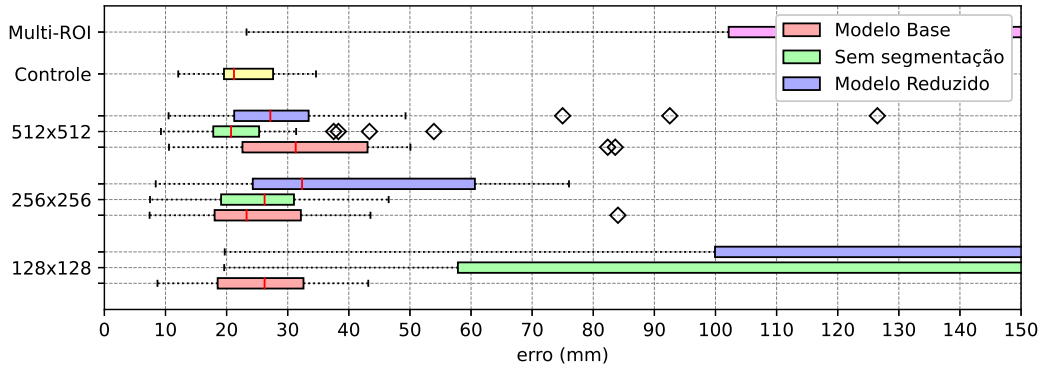

Figura 5.7: Erro absoluto médio observado para cada configuração com câmera superior no Campo 4

condição no qual o modelo não é capaz de operar independente da configuração utilizada.

Ao longo dos campos testados, o algoritmo *Multi-ROI* demonstra eficácia similar aos propostos no primeiro e segundo estágios de crescimento. No terceiro estágio, o alto tempo de processamento observado causa considerável impacto no desempenho. Ao se distanciar da posição alvo, mais de duas fileiras de plantação podem ocorrer no campo de visão da câmera, o que prejudica o método. Por fim, no quarto e quinto estágios, há uma falha completa deste sistema pois todas as janelas deslizantes estão com grande número de *pixels* segmentados, o que prejudica o desempenho e a metologia empregada para a detecção de plantas.

5.2.2 Câmera inferior

O campo 1, contendo as menores plantas para este posicionamento de câmera, é desafiador ao não formar o efeito túnel ou parede desejado para esta configuração. Assim como na câmera superior, uma melhora do desempenho é vista com o aumento da resolução baixa para a média e alguma degradação

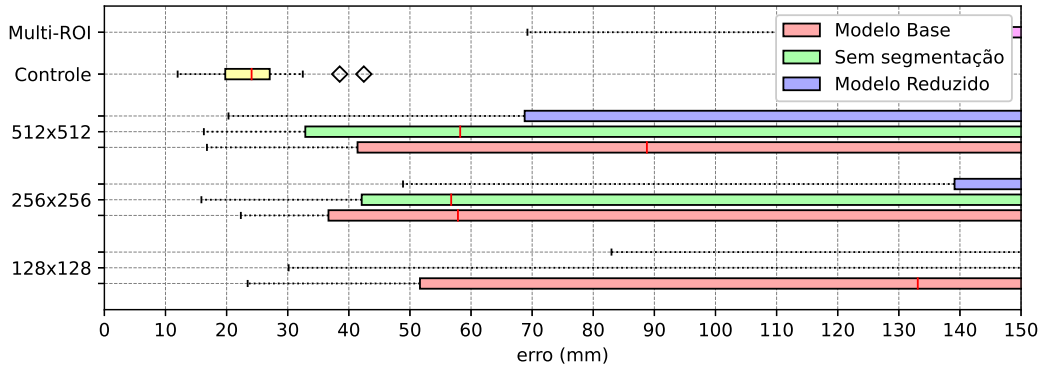

Figura 5.8: Erro absoluto médio observado para cada configuração com câmera superior no Campo 5

pode ser observada no desempenho de maior resolução, exceto para o modelo reduzido, como visto na Figura 5.9.

Os campos com plantas no segundo, terceiro e quarto estágios de crescimento apresentam resultados similares, como ilustrado nas Figuras 5.10, 5.11 e 5.12. Um bom desempenho é obtido em todas configurações testadas, sendo a resolução de 512 *pixels* levemente superior na maioria das configurações. No quarto campo, uma pequena queda de desempenho pode ser notada, de forma majoritária na resolução de 128 *pixels*, mas ainda com um bom desempenho.

No quinto campo, Figura 5.13, o valor médio para os erros se manteve próximo aos observados nos campos anteriores, apenas a menor resolução demonstra uma maior queda no desempenho, ainda que pequena. Nesta configuração, também é observado um aumento no número de *outliers*, entretanto o desempenho é consideravelmente superior à outra configuração de câmera utilizada.

A câmera inferior apresentou, de forma geral, erros menores que a superior no posicionamento do robô, entretanto isso pode ser atribuído a ganhos de controle favoráveis a esta configuração. Foi observado, inclusive,

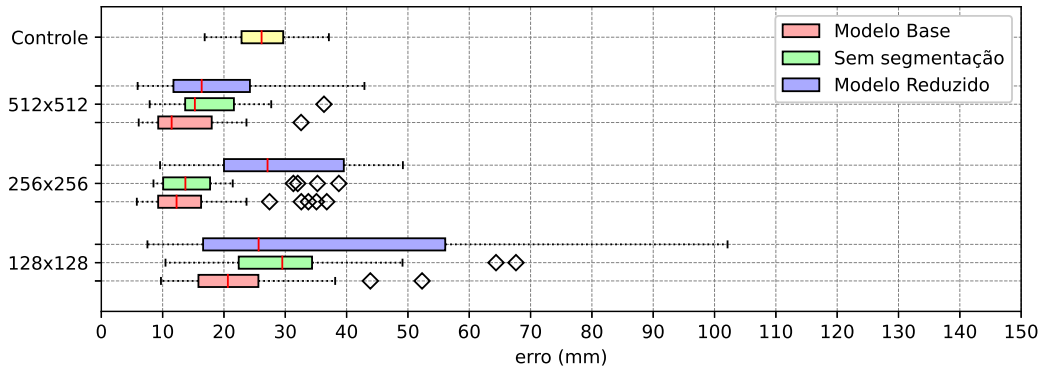

Figura 5.9: Erro absoluto médio observado para cada configuração com câmera inferior no Campo 1

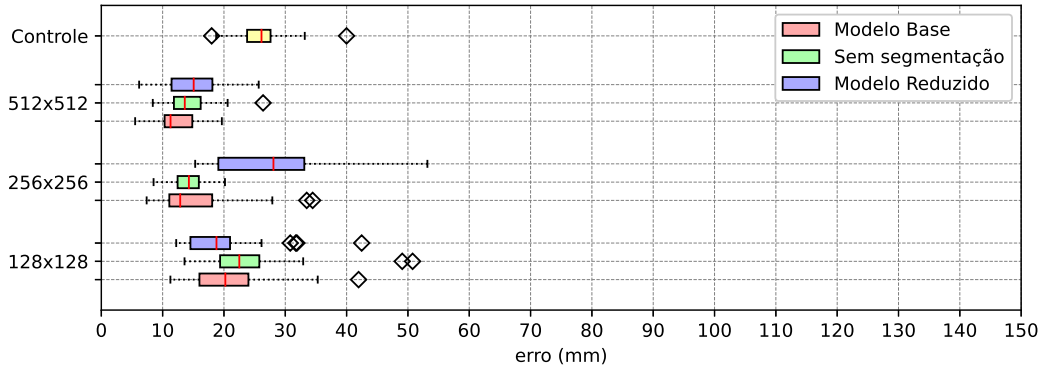

Figura 5.10: Erro absoluto médio observado para cada configuração com câmera inferior no Campo 2

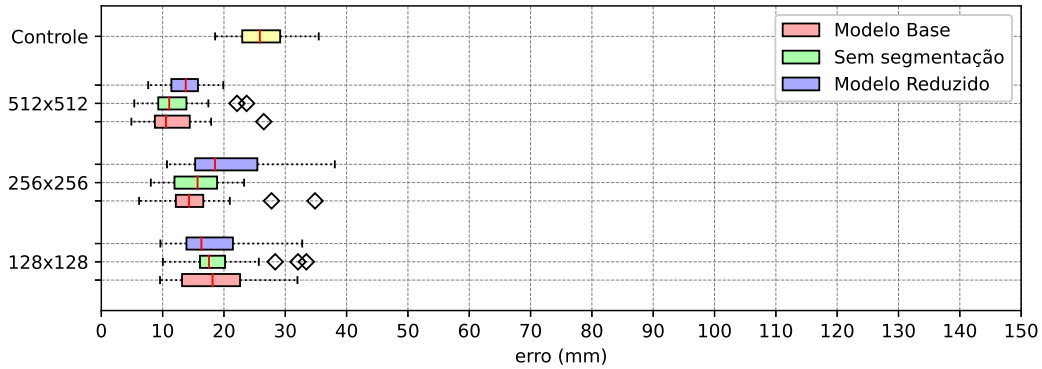

Figura 5.11: Erro absoluto médio observado para cada configuração com câmera inferior no Campo 3

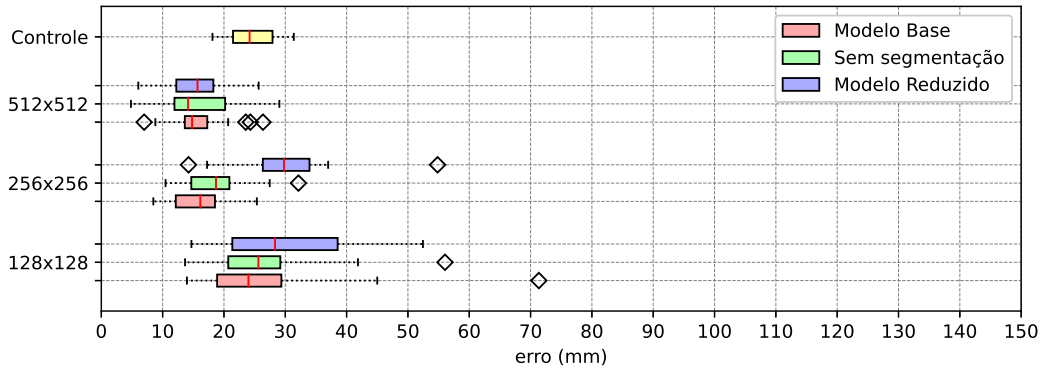

Figura 5.12: Erro absoluto médio observado para cada configuração com câmera inferior no Campo 4

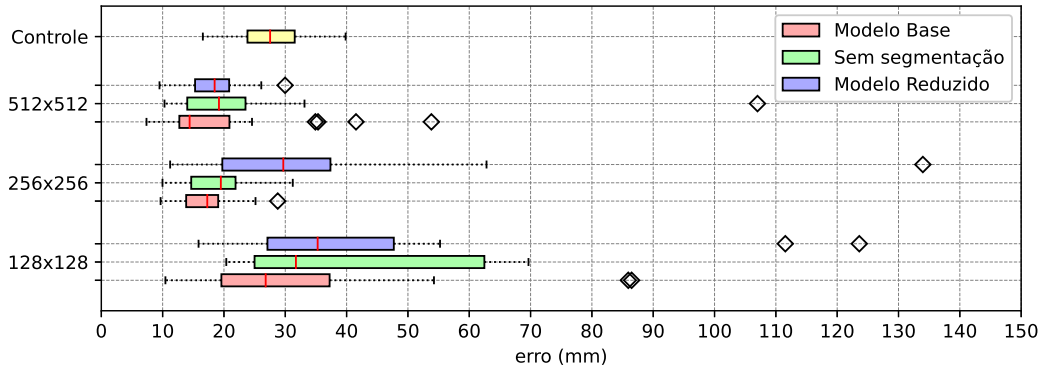

Figura 5.13: Erro absoluto médio observado para cada configuração com câmera inferior no Campo 5

uma maior sensibilidade no posicionamento da linha na imagem, possivelmente derivado da proximidade da câmera com o solo. Durante as simulações foi observado que essa sensibilidade demanda ganhos menores para o controlador, e, consequentemente, a completa recalibração do controlador utilizado pode gerar resultados melhores para cada câmera.

O tempo de processamento também se mostrou um componente importante nos testes realizados. Devido ao alto tempo de processamento observado, o desempenho da linha controle foi prejudicado. Isso indica uma situação na qual que modelos melhores na previsão da linha podem obter resultados inferiores no posicionamento do robô e um bom balanço entre tempo de processamento e qualidade da previsão é necessário.

5.2.3 Comparação entre câmeras superior e inferior

A perturbação na atuação inserida é um dos fatores que também afetam o desempenho de todos os resultados extraídos a partir da posição do robô. Para eliminar a variável do comportamento do controlador e do sistema, o erro da previsão da linha pode ser observado. Este erro pode se manifestar como erros no deslocamento X ou ângulo θ enviados ao controlador.

No campo 1, com plantas no menor tamanho, o erro do deslocamento observado é comparável entre a resoluções de 256 e 512 *pixels*, com e sem segmentação, conforme Figura 5.14. A evolução do desempenho da rede reduzida com o aumento da resolução é novamente notado. Nesta situação, o desempenho inferior do modelo base na resolução de 512 *pixels*, observado na Figura 5.9, pode ser atribuído ao maior tempo de processamento.

Dificuldades no modelo reduzido são vistos na resolução de 128 *pixels*, na câmera superior, dada sua grande variabilidade, fruto de trocas de linha e falhas

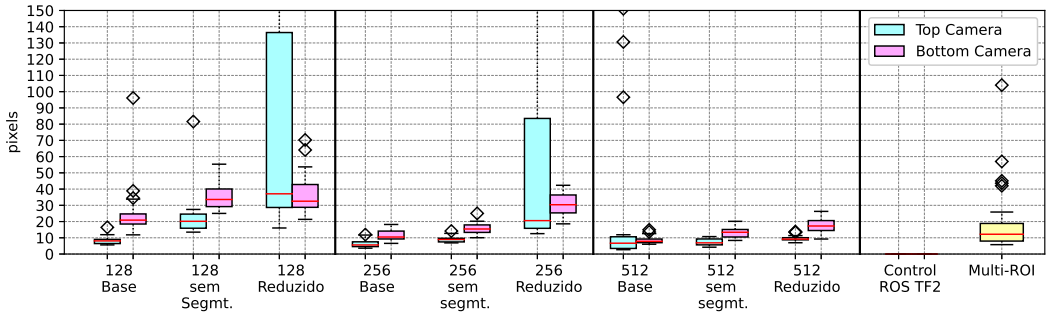

Figura 5.14: Erro absoluto médio observado para o deslocamento da linha no campo 1

gerais, estes podem ser observados na Figura 5.15. O mesmo não acontece na resolução de 256 *pixels* do modelo base, também da câmera superior, onde todas as travessias foram bem sucedidas, mantendo-se ao centro da fileira, como visto na Figura 5.16.

O segundo e terceiro campos, com um estágio de crescimento intermediário, apresentam desempenho similar ao primeiro. Nestes campos, o caminho a ser seguido pode ser mais facilmente detectado e a influência de plantas faltantes ou variação na posição é reduzida. Isso leva a uma boa precisão geral do modelo. Importante notar, também, que o desempenho do algoritmo *Multi-ROI* é comparável aos propostos, conforme observado na Figura 5.17, o que confirma sua eficácia nas corretas condições para utilização.

O quinto campo marca a condição onde a visão superior oferece inferências inadequadas. Todos os modelos de câmera superior oferecem situações em que o erro ultrapassa 150 *pixels*, segundo Figura 5.18, o que é suficiente para causar o desempenho inferior observado na Figura 5.8.

Diante das configurações avaliadas, o modelo reduzido se mostrou mais

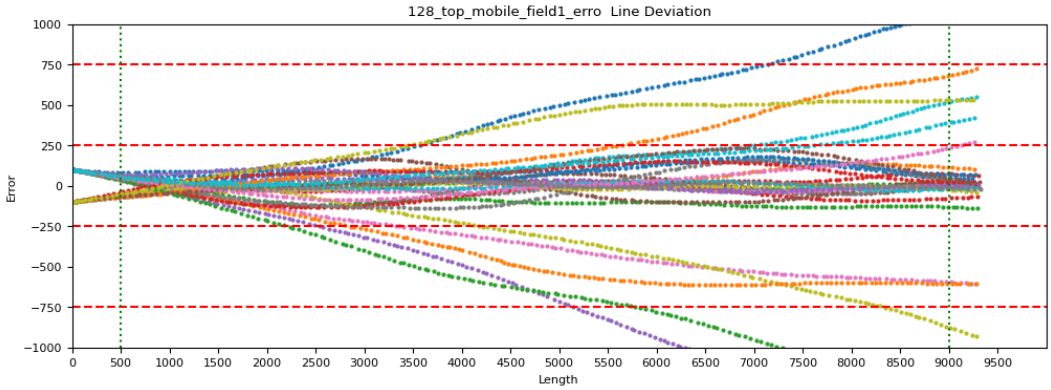

Figura 5.15: Travessias observadas com modelo reduzido e resolução de 128 *pixels* no Campo 1 com câmera superior

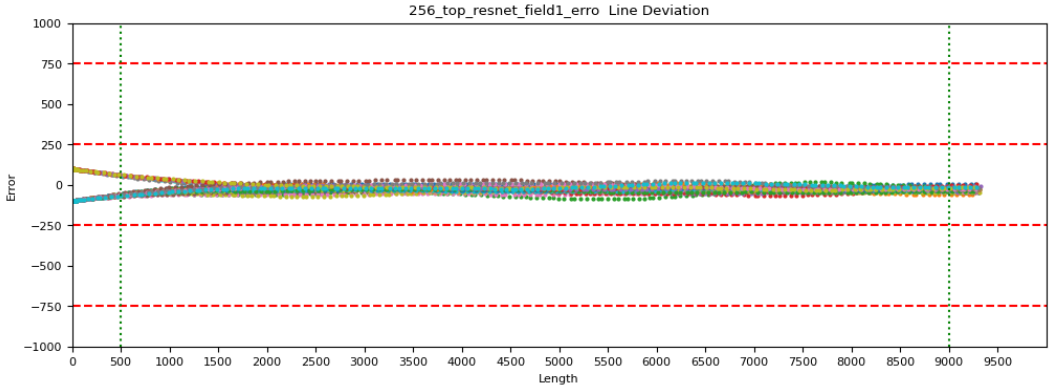

Figura 5.16: Travessias observadas com modelo base e resolução de 256 *pixels* no Campo 1

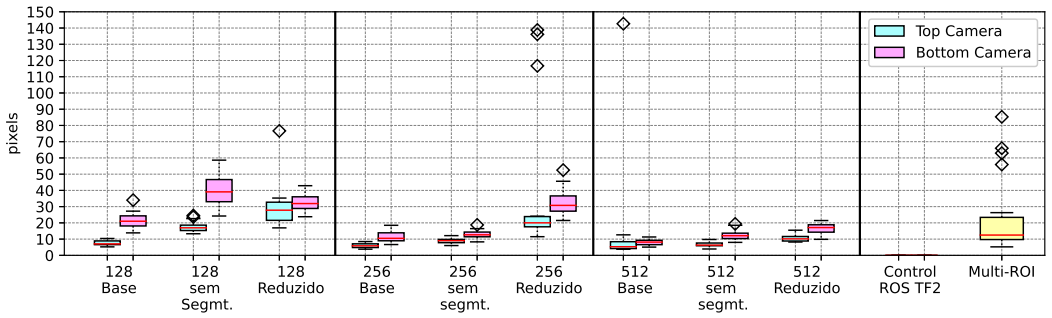

Figura 5.17: Erro absoluto médio observado para o deslocamento da linha no Campo 2

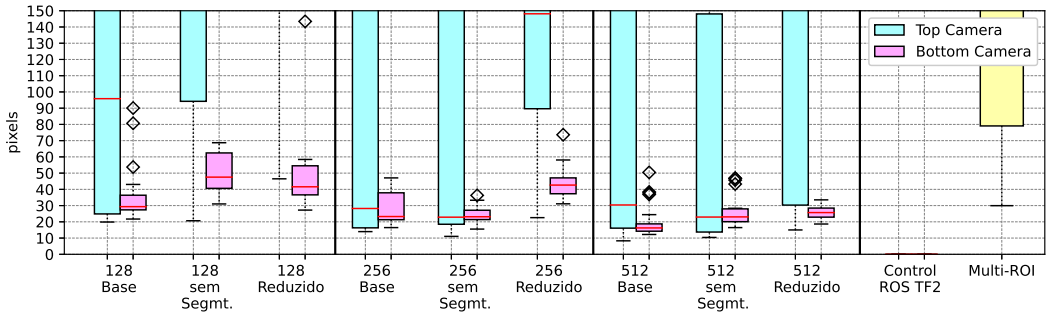

Figura 5.18: Erro absoluto médio observado para o deslocamento da linha no Campo 5

limitado no desempenho geral e qualidade da inferência, porém capaz de operar de maneira satisfatória em diversas circunstâncias, além de possuir um custo computacional menor. Em ambientes onde obter uma máscara de segmentação seja conveniente a outras tarefas, este modelo pode ser uma opção.

O modelo sem a tarefa auxiliar mostrou desempenho competitivo, apresentou baixo tempo de inferência e é de fácil implementação, devido a facilidade em se gerar dados necessários ao treinamento em um novo ambiente. O desempenho ainda é superior, normalmente, ao modelo reduzido e de qualidade similar ao modelo com segmentação de resolução inferior.

O modelo base completo se mostrou o mais capaz ao realizar a previsão da linha, porém com elevado custo computacional. Nos ambientes simulados testados, o tempo de processamento elevado pode tornar o desempenho geral do sistema inferior, por isso o modelo base com 256 *pixels* de resolução se mostrou uma boa alternativa. Esta configuração é capaz de oferecer desempenho satisfatório com tempo de processamento menos custoso em relação a resolução de 512 *pixels*.

5.3 Comportamento em ambientes controlados

O objetivo deste conjunto de testes é validar a integração do nó construído com os demais presentes no robô. Durante esta fase foram ajustados ganhos do controlador, para melhor operação com o robô real, e configurações gerais do sistema. Em situações que o algoritmo proposto precisou ser executado no laptop previamente descrito, as configurações de rede foram ajustadas e testes de transmissão de dados foram realizados para garantir um correto funcionamento.

5.3.1 Soybot-II

Para os testes iniciais com esse robô, o modelo base com resolução de 256 *pixels* foi escolhido. Nas simulações este modelo obteve bons resultados e apresentou funcionamento adequado com este ambiente real, como visto na Figura 5.19. Entretanto, as sombras projetadas se mostraram um desafio, causando, em momentos, erros de segmentação.

Uma boa correlação entre o posicionamento da linha e a máscara gerada pode ser observado durante estes testes. Em momentos no qual a rede não era capaz de realizar uma boa segmentação a linha prevista também não era adequada, este comportamento reforça a conexão esperada entre a tarefa

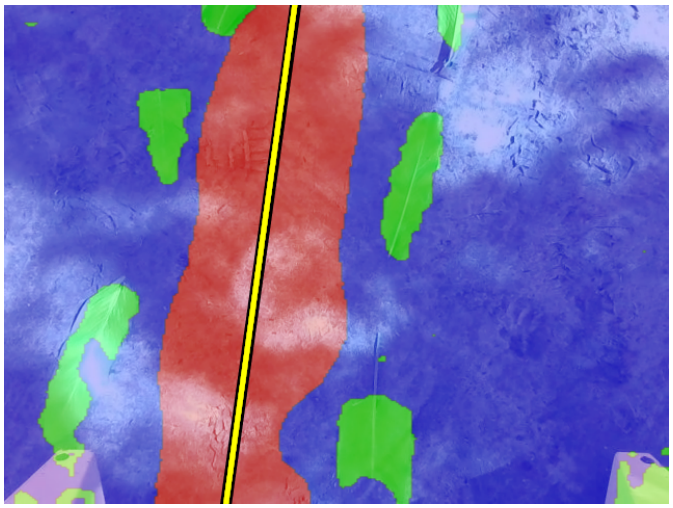

Figura 5.19: Exemplo da saída obtida durante testes preliminares com Soybot-II

auxiliar proposta e a tarefa principal. A saída observada durante os testes está disponível em vídeo[1].

O robô foi capaz de realizar diversas travessias neste ambiente de testes[2]. A previsão do modelo foi próxima da linha manual proposta posteriormente, tanto na distância X, vide Figura 5.20, como no ângulo θ, na Figura 5.21.

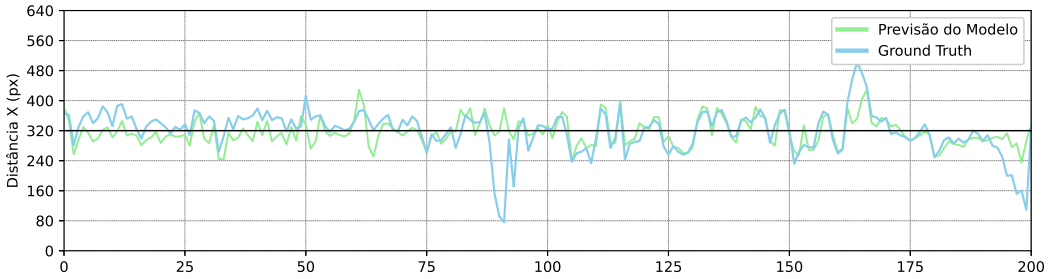

Figura 5.20: Comparação entre a previsão da rede e os valores esperados pela segmentação manual para a distância X

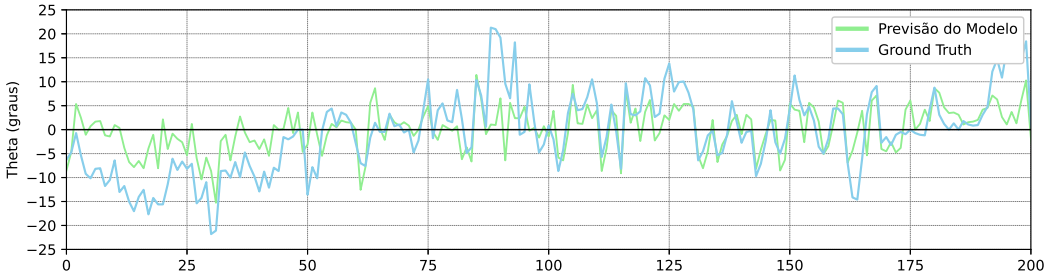

Figura 5.21: Comparação entre a previsão da rede e os valores esperados pela segmentação manual para o ângulo θ

[1]https://youtu.be/Ew3AAtIdKm8
[2]https://youtu.be/ycHnxuPXTbc

5.3.2 Thorvald

A mesma rede utilizada previamente com o robô Soybot-II, proveniente das simulações, foi empregada nos testes iniciais com o robô Thorvald-II. A rede continuou gerando resultados adequados quando apresentada a condições de iluminação e solo similares as anteriores, como visto na Figura 5.22. Importante destacar que ambos os robôs falharam ao serem testados em solo pavimentado, a diferença na coloração e demais características do solo demandam que dados relacionados a esta configuração sejam adicionados ao treino.

O robô também foi capaz de realizar diversas travessias, validando as configurações e ajustes realizados. É importante destacar que executar o nó em outro computador conectado por rede *wireless* adiciona latência na transmissão de dados, em casos de sobrecarga de banda na rede utilizada os resultados são comprometidos. Durante os testes realizados isto foi observado ao não aplicar compressão nas imagens.

Após segmentação manual dos resultados obtidos, a Figura 5.23 denota a distância X da linha prevista enquanto a Figura 5.24 mostra a comparação entre os ângulos. Ambos os valores se comportaram de maneira similar com a rede demonstrando previsões conservadoras, isto é, mais próximas do centro. A travessia também pode ser observada em video[3].

[3]https://youtube.com/shorts/fu7DyqUyCQg

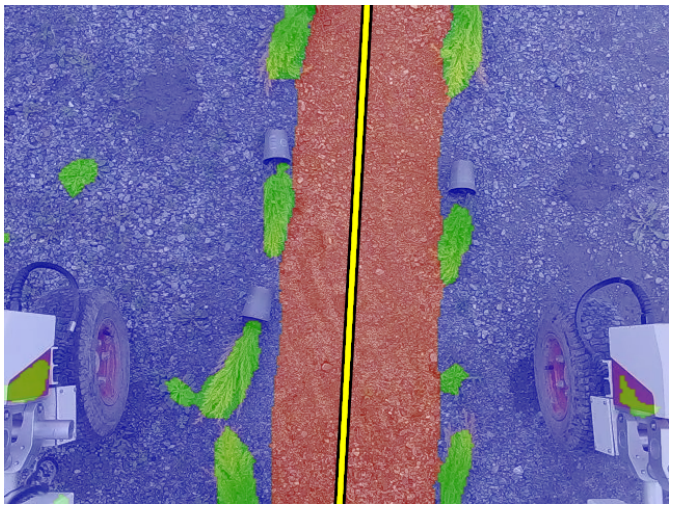

Figura 5.22: Exemplo da saída obtida durante testes preliminares com Thorvald-II

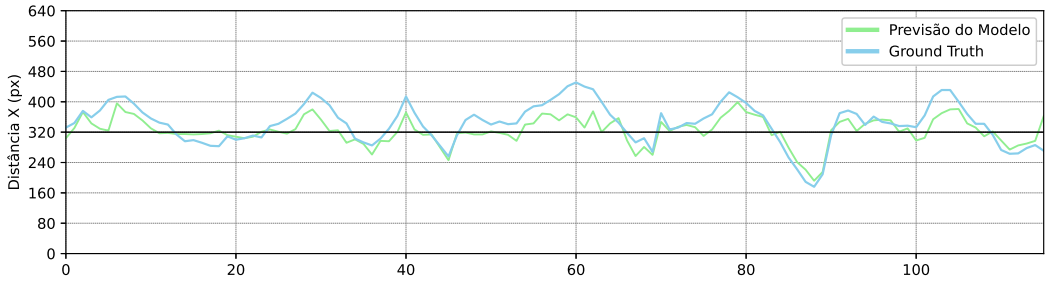

Figura 5.23: Comparação entre a previsão da rede e os valores esperados pela segmentação manual para a distância X

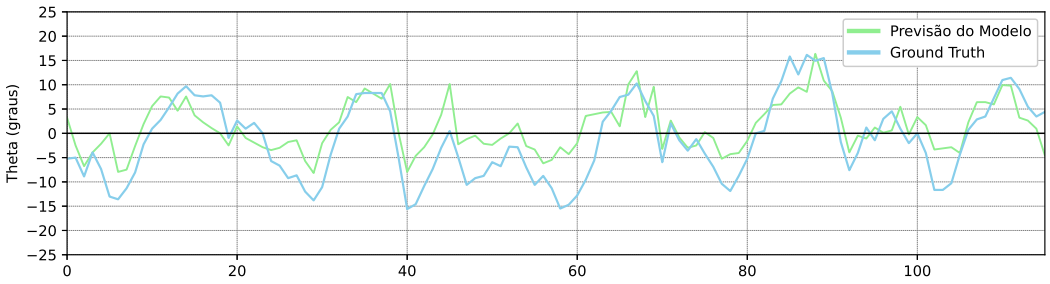

Figura 5.24: Comparação entre a previsão da rede e os valores esperados pela segmentação manual para o ângulo θ

5.4 Resultados em ambientes reais

5.4.1 Resultados plantação de morangos

Para este cenário, o modelo base com resolução de 512 *pixels* foi escolhido para a utilização, por ser uma das melhores configurações observadas durante a simulação e a segmentação para este cenário ser de fácil execução. O robô foi capaz de percorrer toda a fileira de plantação, entretanto um deslocamento lateral pode ser observado durante o processo[4], conforme Figura 5.25.

Isso pode, também, ser percebido ao comparar a previsão do modelo com uma segmentação manual posterior, realizada antes de observar a previsão. Na Figura 5.26, o modelo apresenta uma tendência em manter a linha mais próxima ao centro da imagem, aumentando a previsão do deslocamento da linha apenas em condições mais extremas. O comportamento angular apresentou características similares, porém apresentou alta variabilidade, conforme Figura 5.27.

[4]https://youtu.be/Khc0aHjeb-8

Figura 5.25: Deslocamento lateral na posição do robô observado durante as travessias

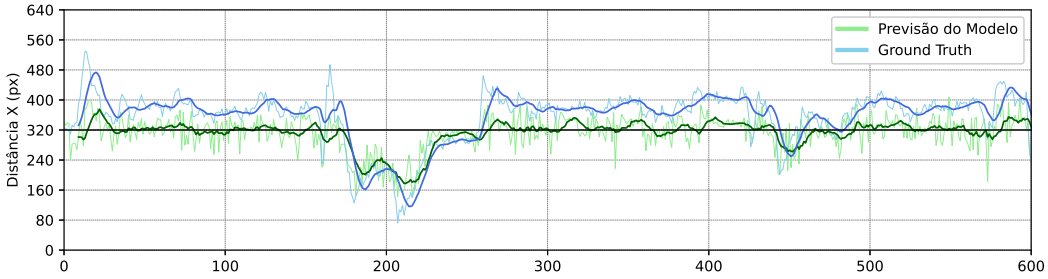

Figura 5.26: Comparação do deslocamento X da linha segmentada e produzida pelo modelo - Linha escura auxiliar adicionada para visualização facilitada representando a média deslizante ao longo do tempo com janela de 10 medidas

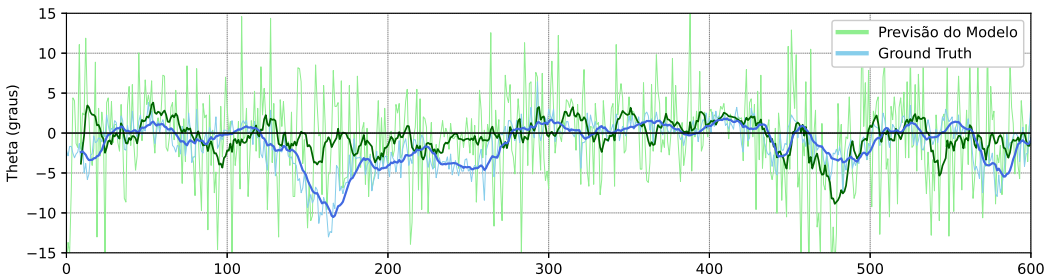

Figura 5.27: Comparação do ângulo θ da linha segmentada e produzida pelo modelo - Linha escura auxiliar adicionada para visualização facilitada representando a média deslizante ao longo do tempo com janela de 10 medidas

Embora a tendência observada prejudique a previsão da linha, ao manter resultados conservadores, porém em fase, isto é, mesma direção apesar de menor valor, permitiu que o robô completasse a travessia com sucesso, já que seria equivalente a utilizar um sinal de controle atenuado. A Figura 5.28 exemplifica esta divergência observada nos resultados.

É importante destacar que o modelo foi capaz de operar a partir de dados de treinamento registrados com câmera consideravelmente diferente da utilizada durante a travessia. Desta forma, um treino posterior com dados extraídos da travessia realizada com o modelo inicial poderia levar a resultados superiores, o que sugere flexibilidade quanto a câmera utilizada para o processo.

5.4.2 Resultados em campo de plantio acadêmico misto

O modelo sem segmentação foi escolhido para os testes no cenário de campo de plantio acadêmico misto, por requirir um *dataset* simples (não segmentado), usando a resolução de 512 pixels. O robô foi capaz de percorrer diferentes fileiras em ambos os sentidos com sucesso[5]. Algumas situações se mostraram desafiadoras, com o robô exibindo maiores dificuldades em fileiras largas com grandes espaços vazios, conforme pode ser visto na Figura 5.29.

O modelo treinado apresentou capacidade adequada de posicionar a linha na maioria dos cenários, entretanto se mostrou incapaz de posicionar a linha nas extremidades laterais da imagem conforme Figura 5.30(b). Esta

[5]https://youtu.be/DwrEfACJno8

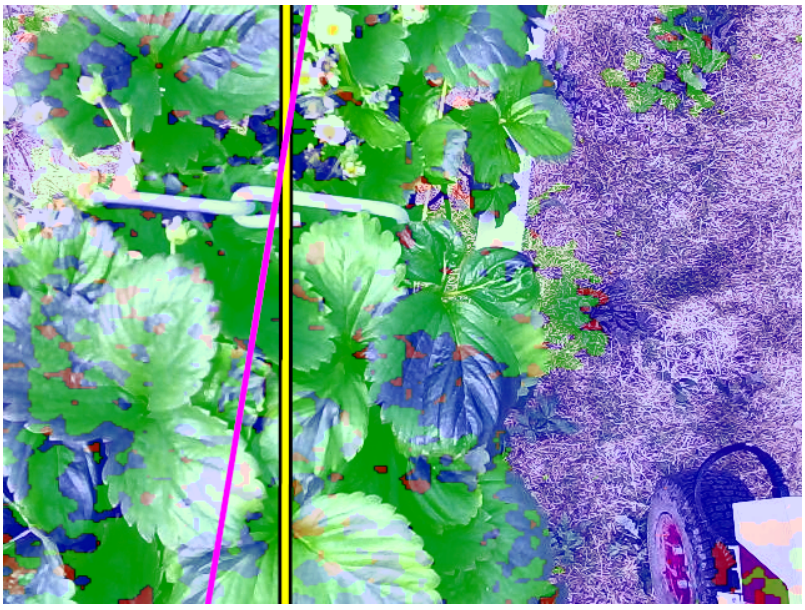

Figura 5.28: Exemplo da segmentação realizada pelo modelo em comparação da linha de referência utilizada (em rosa)

Figura 5.29: Vista superior do campo de plantio, fileiras com diversas configurações são presentes

característica não impediu o algoritmo de operar, mas a saída se comporta de forma atenuada nestes casos o que pode causar que o robô não se corrija em tempo hábil.

As Figuras 5.31 e 5.32 comparam a saída da rede com a linha marcada de forma manual. Pode ser observado, principalmente no ângulo θ, que há uma leve tendência na previsão, porém acaba sendo compensado pelo deslocamento X, assim, o modelo tende a manter a linha levemente angulada conforme Figura 5.30(c).

Neste campo, foi realizado a troca da câmera utilizada para verificar o comportamento do modelo. A câmera Intel D415i foi substituída temporariamente pela Lenovo FHDWC510, que possui campo de visão maior. A nova câmera, devido ao grande campo de visão, exacerbou o comportamento observado da linha, onde o modelo tende a manter a previsão da linha em posição mais central da imagem. Isto fez com que o robô operasse normalmente em condições padrão, porém ter menos capacidade de recuperação ao estar fora de centro.

Figura 5.30: Exemplos de linhas geradas pelo modelo (em rosa): (a) Linha correta; (b) Pequeno desvio angular; (c) Grande desvios observados nas extremidades

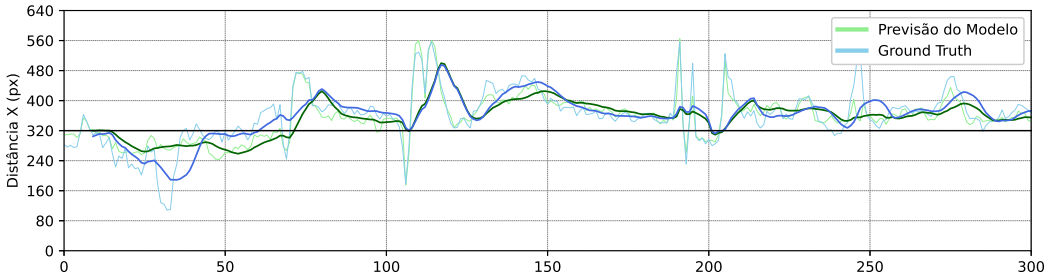

Figura 5.31: Comparação do deslocamento X da linha segmentada e produzida pelo modelo - Linha escura auxiliar adicionada para visualização facilitada representando a média deslizante ao longo do tempo com janela de 10 medidas

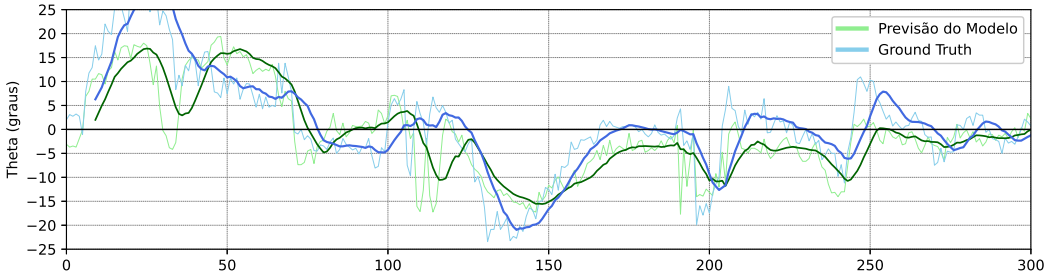

Figura 5.32: Comparação do ângulo θ da linha segmentada e produzida pelo modelo - Linha escura auxiliar adicionada para visualização facilitada representando a média deslizante ao longo do tempo com janela de 10 medidas

5.4.3 Comparação dos resultados de campo

A Figura 5.33 traz o valor do erro absoluto do deslocamento X e angulo θ médio para os cenários discutidos nas seções anteriores. Um erro angular médio de menos de 10° foi observado ao longo de todos os testes obtidos, onde apenas o resultado do túnel de morangos apresentou erro no deslocamento maior que 50 *pixels* de forma consistente.

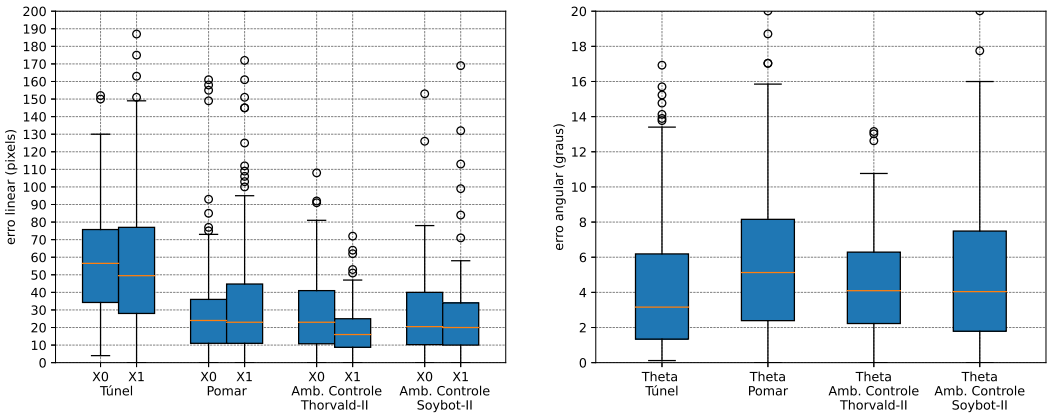

Figura 5.33: Comparação do erro das saídas da rede X_0 e X_1 para os testes de campo realizados (esquerda) e do ângulo θ derivado da linha para os testes de campo realizados (direita)

Este erro maior, neste cenário, pode ser atribuído ao fato dos dados coletados para treinamento não terem sido realizados com o robô. Ambos os valores X_0 e X_1 utilizados para produzir a linha alcançaram valores similares para o erro médio e o erro angular se mostrou consistente entre os modelos observados.

Considerando que as melhores configurações das simulações foram utilizadas, estas obtiveram resultados semelhantes entre si, de forma similar ao observado anteriormente nas simulações. Entretanto, o modelo com segmentação se mostrou mais robusto na utilização geral durante o trabalho em campo, assim seu uso é recomendado, novamente, frente ao sem segmentação.

5.4.4 Comparação cruzada entre câmeras

Os *datasets* foram produzidos em momentos distintos e em fileiras diversas dos pomares. Desta forma, podem conter diferenças e características particulares, além da diferença de reprodução de imagem por parte de cada câmera. Cada modelo foi treinado em apenas um *dataset* com os mesmos parâmetros descritos nos capítulos 3 e 4, utilizando a resolução de 256 *pixels* sem máscara de segmentação.

Para cada câmera avaliada, nesta comparação, um *dataset* contendo 580 imagens, com linha de trajetória marcada, foi utilizado em uma divisão de 70% treino, 20% validação e 10% teste. Em seguida, os três *datasets* foram combinados em um modelo misto. Cada modelo foi avaliado com o conjunto teste de todos os outros *datasets* e no *dataset* misto, com resultados segundo tabela 5.3.

Dos conjuntos de dados elaborados, o misto apresenta os melhores resultados, demonstrando que a junção de todos os dados apresenta ao modelo maiores condições de aprendizagem apesar da variabilidade apresentada.

Entretanto, os demais modelos apresentaram erros maiores e com mais variabilidade quando apresentados a conjuntos teste dos demais *datasets*. Porém, não é possível discernir uma tendência entre melhor ou pior *dataset*

Tabela 5.3: Erro médio e desvio padrão observado para os modelos treinados com diferentes datasets advindos de diferentes câmeras

Modelo	Intel D415i Dataset	Lenovo Cam Dataset	Smartphone Cam Dataset	Dataset Misto
Intel D415i	19,0 (17,7)	58,8 (59,4)	80,2 (78,5)	52,6 (63,0)
Lenovo Cam	59,5 (59,7)	38,0 (38,3)	77,0 (62,5)	58,2 (56,9)
SmartP. Cam	46,6 (38,5)	61,2 (57,2)	48,5 (48,4)	52,1 (49,0)
Misto	20,7 (13,7)	25,4 (30,6)	36,7 (35,7)	27,7 (29,0)

ou modelo, levando a acreditar que o impacto de mudança da câmera existe mas não é grande o suficiente para inviabilizar o modelo.

Disto, pode ser concluído que dados obtidos de outras fontes diferentes do robô podem ser utilizados para uma execução inicial. Porém, dados obtidos com a câmera a ser utilizada durante a operação são melhores e manter *datasets* anteriores, quando relevantes, podem contribuir positivamente para o desempenho do modelo. Então, o modelo misto obtém resultados melhores de maneira consistente, superando, muitas vezes, os resultados observados por um modelo treinado e exposto a apenas um *dataset*.

Assim, ao longo do capítulo 5, podemos observar o desempenho do sistema proposto sob múltiplas óticas e situações. O desempenho com todas as variações de parâmetros propostas foi observado no ambiente simulado e algumas com melhor desempenho avaliadas nos ambientes reais, o que permite a discussão dos resultados aqui obtidos e as considerações finais realizadas no capítulo 6.

6
Conclusão e Trabalhos Futuros

Este trabalho teve por objetivo desenvolver um sistema de navegação autônomo autônomo acessível e de simples implantação para fileiras de plantação. Simulações de diversas configurações possíveis e execução de testes com o robô Thorvald-II em campos foram conduzidos. Este capítulo visa sintetizar e discutir os resultados obtidos e, ao final, apontar possíveis áreas onde trabalhos futuros possam ser explorados.

6.1
Conclusão

O sistema de navegação autônoma proposto foi capaz, nos testes e simulações realizadas, de percorrer os campos de plantações propostos. O sistema foi construído com o *framework* ROS, na sua versão 1, possuindo três variações: modelo base, reduzido e sem tarefa auxiliar.

O nó ROS desenvolvido é capaz de operar de forma remota, através de conexão *wireless*. O nó utiliza tópicos de imagem com compressão para permitir menor uso de dados e envia ao robô, com uso de controle previamente proposto, velocidades linear e angular desejadas para o deslocamento.

Para a correta operação do nó, é necessária apenas uma câmera RGB capaz de operar adequadamente sob possivelmente intensa luz solar. Embora não requerido, o nó desenvolvido também é capaz de controlar o robô com o uso de um controle de console compatível com o sistema Linux. Além disso, uma GPU compatível com instruções CUDA é recomendável.

Para realização das simulações, um ambiente gazebo foi elaborado contendo uma variedade de plantas dispostas em fileiras contendo cinco diferentes estágios de crescimento. Desta simulação, imagens já segmentadas e uma linha referência ideal podem ser extraídas para fácil implementação e testes de arquiteturas de rede.

Uma ferramenta para a criação de dados de treinamento a partir de imagens de campo, já existente, foi aprimorada. Esta ferramenta é capaz de produzir dados para treinamento do modelo sem tarefa auxiliar de maneira rápida e é, também, capaz de gerar máscaras de segmentação em condições onde as plantas podem ser facilmente distinguidas do ambiente a sua volta.

A ferramenta de criação de dados foi utilizada para gerar um *dataset* para a plantação de morangos, onde, posteriormente, o robô Thorvald-II foi capaz de se deslocar ao longo de toda a plantação. A ferramenta foi, inclusive, utilizada para gerar linhas referência sem máscaras para o treinamento do modelo sem tarefa auxiliar, utilizado na plantação de árvores frutíferas mistas.

Das três variações propostas, o modelo com tarefa auxiliar demonstra os melhores resultados de previsão da linha, porém possui alto custo computacional e maior dificuldade de obtenção de novos dados, devido à segmentação.

O modelo reduzido se mostra uma opção válida para casos onde a segmentação pode ser aproveitada para outras tarefas e há pouco poder computacional disponível. Para a configuração base é recomentado uma GPU para a realização do processamento, entretanto o modelo reduzido é possível com uma CPU moderna.

Por fim, o modelo sem tarefa auxiliar de segmentação se mostrou eficaz nos cenários avaliados com desempenho igual ou superior ao modelo reduzido. Também possui baixo tempo de processamento e os dados necessários para o treinamento são de simples obtenção. Foi observado que o tempo de processamento é um fator de extrema importância, pois pode causar instabilidade no sistema de controle utilizado.

Baseado nos testes de ambiente controlado e de campo, os resultados obtidos foram promissores. A abordagem *end-to-end* utilizada para a determinação da linha a ser seguida para o robô foi capaz de operar com sucesso com diferentes câmeras montadas em diferentes posições em diferentes robôs. Desta forma, observou-se flexibilidade e capacidade de generalização das redes neurais convolucionais.

Os testes de campo apontaram a importância do posicionamento correto da câmera no robô, que mesmo sob pequenos desalinhamentos podem inserir erros no deslocamento. Também foi observado que o algoritmo proposto para a câmera inferior funciona melhor em fileiras de plantação com menores dimensões. Ao utilizar câmeras de grande abertura em fileiras largas, a anotação dos dados pode se tornar ambígua e propensa a erros, onde o desempenho da rede é comprometido.

Por fim, o sistema proposto se mostrou bem sucedido na tarefa alvo com as limitações e características citadas. O sistema se mostrou eficiente, mesmo em frente a trocas de câmeras, e possui simples aquisição de dados, podendo ser uma alternativa adicional para a navegação em fileiras de plantação. O pacote desenvolvido ao longo deste trabalho pode ser obtido no GitHub.[1]

[1]https://github.com/ifcosta/cnn_line

6.2 Trabalhos Futuros

Durante a execução do presente trabalho, limitações e oportunidades de desenvolvimento futuro puderam ser vislumbradas. Primeiramente, as alterações propostas na arquitetura DeepLabV3+ são iniciais. Um compreensivo estudo com diferentes topologias de rede pode ser desenvolvido a fim de refinar o modelo e configuração utilizada.

A utilização das duas posições de câmera propostas de forma conjunta também pode ser explorado, assim como um modelo que realize tal inferência de forma combinada. A utilização de câmeras em diferentes posições nos robôs utilizados também é de importante avaliação, principalmente sob a possibilidade de combinar diferentes detecções.

A integração de relevo no ambiente simulado é desejável para permitir a avaliação do impacto de diferentes estratégias de controle. Separar a detecção da linha e obtenção da velocidade desejada em diferentes nós também se mostra um outro passo para a criação de um sistema de navegação flexível e customizável, no qual diferentes controladoras podem assumir em situações características, como na entrada de fileira.

De maneira similar, a integração de sistemas de localização clássicos, como LiDAR, para detecção de obstáculos, IMU, técnicas de odometria visual ou mesmo detecção do ambiente, contribuem para melhor localização espacial do robô. Assim, o sistema proposto pode atuar em conjunto com rotas planejadas e possuir redundância para situações de falha.

Técnicas de filtragem da linha obtida também podem ser exploradas para retirar *outliers* e tornar o modelo mais robusto, uma vez que o robô terá menos chances de reagir incorretamente. Em implementações futuras, a câmera, com a abordagem aqui proposta, pode se tornar um sensor em um conjunto de técnicas paralelas, oferecendo aumento de precisão a um baixo custo de implementação.

Com o incentivo à adoção e o amadurecimento de ROS 2, ja que ROS 1 está em sua última versão, todo o sistema aqui proposto e a máquina de estados do Soybot-II tem uma oportunidade de ser retrabalhada e atualizada para a nova versão do *framework*.

Referências bibliográficas

[1] COSTA, R. R. D.. **Ipea - instituto de pesquisa econômica aplicada**, Feb 2023.

[2] BERGERMAN, M.; BILLINGSLEY, J.; REID, J. ; VAN HENTEN, E.. **Robotics in agriculture and forestry**. In: SPRINGER HANDBOOK OF ROBOTICS, p. 1463–1492. Springer International Publishing, 2016.

[3] BARBOSA, G. B. P. **Robust vision-based autonomous crop row navigation for wheeled mobile robots in sloped and rough terrains**. Dissertação de mestrado em engenharia elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, 2022.

[4] EDAN, Y.; HAN, S. ; KONDO, N.. **Automation in agriculture**. In: SPRINGER HANDBOOK OF AUTOMATION, p. 1095–1128. Springer Berlin Heidelberg, 2009.

[5] BECHAR, A.; VIGNEAULT, C.. **Agricultural robots for field operations: Concepts and components**. Biosystems Engineering, 149:94–111, Sept. 2016.

[6] GRIMSTAD, L.; FROM, P. J.. **Thorvald II - a modular and reconfigurable agricultural robot**. IFAC-PapersOnLine, 50(1):4588–4593, July 2017.

[7] LE, T. D.; PONNAMBALAM, V. R.; GJEVESTAD, J. G. O. ; FROM, P. J.. **A low-cost and efficient autonomous row-following robot for food production in polytunnels**. Journal of Field Robotics, 37(2):309–321, June 2019.

[8] BARBOSA, W. S.; OLIVEIRA, A. I. S.; BARBOSA, G. B. P.; LEITE, A. C.; FIGUEIREDO, K. T.; VELLASCO, M. M. B. R. ; CAARLS, W.. **Design and development of an autonomous mobile robot for inspection of soy and cotton crops**. In: 2019 12TH INTERNATIONAL CONFERENCE ON DEVELOPMENTS IN eSystems ENGINEERING (DeSE). IEEE, Oct. 2019.

[9] PONNAMBALAM, V. R.; BAKKEN, M.; MOORE, R. J. D.; GJEVESTAD, J. G. O. ; FROM, P. J.. **Autonomous crop row guidance using adaptive multi-ROI in strawberry fields**. Sensors, 20(18):5249, Sept. 2020.

[10] MARTINS, F. F. **Navigation and pest identification systems for the soybot pest monitoring robot**. Dissertação de mestrado em engenharia elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, 2020.

[11] AHMADI, A.; NARDI, L.; CHEBROLU, N. ; STACHNISS, C.. **Visual servoing-based navigation for monitoring row-crop fields**. In: 2020 IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA). IEEE, May 2020.

[12] GOODFELLOW, I.; BENGIO, Y. ; COURVILLE, A.. **Deep Learning**. MIT Press, 2016. `http://www.deeplearningbook.org`.

[13] MARTINS, F. F.; CARVALHO, T. M.; CELECIA, A.; OLIVEIRA, A. I. S.; BARBOSA, G. B. P.; VELLASCO, M. M. B.; CAARLS, W.; FIGUEIREDO, K. ; LEITE, A. C.. **Sistema de navegação autônoma para o robô agrícola soybot**. In: PROCEDINGS DO XV SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE. SBA Sociedade Brasileira de Automática, 2021.

[14] AHMADI, A.; HALSTEAD, M. ; MCCOOL, C.. **Towards autonomous visual navigation in arable fields**. 2021.

[15] RASCHKA, S.; MIRJALILI, V. **Python Machine Learning**. Packt Publishing Ltd, Birmingham, United Kingdom, 3nd edition, 2019.

[16] MAHESH, B.. **Machine learning algorithms - a review**. International Journal of Science and Research, 1:382–386, 2020.

[17] SUMIT, S. **A comprehensive guide to convolutional neural networks**. A Comprehensive Guide to Convolutional Neural Networks, 2018. Acesso em: Abril de 2023.

[18] HE, K.; ZHANG, X.; REN, S. ; SUN, J.. **Deep residual learning for image recognition**. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), p. 382–386, 2016.

[19] HOWARD, A. G.; ZHU, M.; CHEN, B.; KALENICHENKO, D.; WANG, W.; WEYAND, T.; ANDREETTO, M. ; ADAM, H.. **Mobilenets: Efficient**

convolutional neural networks for mobile vision applications, 2017.

[20] SANDLER, M.; HOWARD, A.; ZHU, M.; ZHMOGINOV, A. ; CHEN, L.-C.. **MobileNetV2: Inverted residuals and linear bottlenecks**. In: 2018 IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION. IEEE, June 2018.

[21] CHEN, L.-C.; PAPANDREOU, G.; KOKKINOS, I.; MURPHY, K. ; YUILLE, A.. **Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs**. IEEE Transactions on Pattern Analysis and Machine Intelligence, PP, 06 2016.

[22] CHEN, L.-C.; PAPANDREOU, G.; SCHROFF, F. ; ADAM, H.. **Rethinking atrous convolution for semantic image segmentation**, 2017.

[23] CHEN, L.-C.; ZHU, Y.; PAPANDREOU, G.; SCHROFF, F. ; ADAM, H.. **Encoder-decoder with atrous separable convolution for semantic image segmentation**, 2018.

[24] TORREY, L.; SHAVLIK, J.. **Transfer learning**. In: HANDBOOK OF RESEARCH ON MACHINE LEARNING APPLICATIONS, p. 242–264, Valencia, Spain, 2009. IGI Global.

[25] DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K. ; FEI-FEI, L.. **ImageNet: A large-scale hierarchical image database**. In: 2009 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 248–255. IEEE, June 2009.

[26] DENG, L.. **The mnist database of handwritten digit images for machine learning research**. IEEE Signal Processing Magazine, 29(6):141–142, 2012.

[27] KRIZHEVSKY, A.. **Learning multiple layers of features from tiny images**. University of Toronto, 05 2012.

[28] LIEBEL, L.; KÖRNER, M.. **Auxiliary tasks in multi-task learning**, 2018.

[29] CIPOLLA, R.; GAL, Y. ; KENDALL, A.. **Multi-task learning using uncertainty to weigh losses for scene geometry and semantics**, 2018.

[30] KUSUMAM, K.; KRAJNÍK, T.; PEARSON, S.; DUCKETT, T. ; CIELNIAK, G.. **3d-vision based detection, localization, and sizing of broccoli heads in the field**. Journal of Field Robotics, 34(8):1505–1518, June 2017.

[31] NAKARMI, A. D.; TANG, L.. **Within-row spacing sensing of maize plants using 3d computer vision**. Biosystems Engineering, 125:54–64, Sept. 2014.

[32] MCCOOL, C. S.; BEATTIE, J.; FIRN, J.; LEHNERT, C.; KULK, J.; BAWDEN, O.; RUSSELL, R. ; PEREZ, T.. **Efficacy of mechanical weeding tools: a study into alternative weed management strategies enabled by robotics**. IEEE Robotics and Automation Letters, p. 1–1, 2018.

[33] WU, X.; ARAVECCHIA, S. ; PRADALIER, C.. **Design and implementation of computer vision based in-row weeding system**. In: 2019 INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA). IEEE, May 2019.

[34] ENGLISH, A.; ROSS, P.; BALL, D.; UPCROFT, B. ; CORKE, P.. **Learning crop models for vision-based guidance of agricultural robots**. In: 2015 IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS). IEEE, Sept. 2015.

[36] VITOR AKIHIRO HISANO HIGUTI. **2d lidar-based perception for under canopy autonomous of small ground robots within narrow lanes of agricultural fields**. Dissertação de mestrado em engenharia mecânica, Universidade de São Paulo, São Carlos, SP, 2021.

[37] SHALAL, N.; LOW, T.; MCCARTHY, C. ; HANCOCK, N.. **Orchard mapping and mobile robot localisation using on-board camera and laser scanner data fusion – part a: Tree detection**. Computers and Electronics in Agriculture, 119:254–266, Nov. 2015.

[38] BERGERMAN, M.; MAETA, S. M.; ZHANG, J.; FREITAS, G. M.; HAMNER, B.; SINGH, S. ; KANTOR, G.. **Robot farmers: Autonomous orchard vehicles help tree fruit production**. IEEE Robotics Automation Magazine, 22(1):54–63, Mar. 2015.

[39] ZHANG, J.; CHAMBERS, A.; MAETA, S.; BERGERMAN, M. ; SINGH, S.. **3d perception for accurate row following: Methodology and results**. In: 2013 IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS. IEEE, Nov. 2013.

[40] BALDWIN, I.; NEWMAN, P.. **Road vehicle localization with 2d push-broom LIDAR and 3d priors**. In: 2012 IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION. IEEE, May 2012.

[41] DE SILVA, R.; CIELNIAK, G.; WANG, G. ; GAO, J.. **Deep learning-based crop row following for infield navigation of agri-robots**, 2022.

[42] XAUD, M. F. S.; LEITE, A. C. ; FROM, P. J.. **Thermal image based navigation system for skid-steering mobile robots in sugarcane crops**. In: 2019 INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA). IEEE, May 2019.

[43] CHERUBINI, A.; CHAUMETTE, F. ; ORIOLO, G.. **An image-based visual servoing scheme for following paths with nonholonomic mobile robots**. In: 2008 10TH INTERNATIONAL CONFERENCE ON CONTROL, AUTOMATION, ROBOTICS AND VISION. IEEE, Dec. 2008.