

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

**ALESSANDRO MASSAYUKI NAKATANI
JOSÉ MÁRIO NISHIHARA DE ALBUQUERQUE**

**ANÁLISE DO USO DE INFORMAÇÕES DE PROFUNDIDADE PARA
CLASSIFICAÇÃO DE OBJETOS EM SISTEMAS ELÉTRICOS DE POTÊNCIA
COM UM SENSOR MULTIMODAL**

CURITIBA

2025

**ALESSANDRO MASSAYUKI NAKATANI
JOSÉ MÁRIO NISHIHARA DE ALBUQUERQUE**

**ANÁLISE DO USO DE INFORMAÇÕES DE PROFUNDIDADE PARA
CLASSIFICAÇÃO DE OBJETOS EM SISTEMAS ELÉTRICOS DE POTÊNCIA
COM UM SENSOR MULTIMODAL**

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Eletrônica do Curso de Bacharelado em Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná.

Orientador(a): Prof. Dr. Ronnier Frates Rohrich

Coorientador(a): Prof. Dr. André Schneider de Oliveira

CURITIBA

2025



[4.0 Internacional](#)

Esta licença permite remixe, adaptação e criação a partir do trabalho, para fins não comerciais, desde que sejam atribuídos créditos ao(s) autor(es). Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

**ALESSANDRO MASSAYUKI NAKATANI
JOSÉ MÁRIO NISHIHARA DE ALBUQUERQUE**

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Eletrônica do Curso de Bacharelado em Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná.

Data de aprovação: dia / mês / ano

Nome completo e por extenso do Membro 1 (de acordo com o Currículo Lattes)

Titulação (Especialização, Mestrado, Doutorado)

Nome completo e por extenso da instituição a qual possui vínculo

Nome completo e por extenso do Membro 2 (de acordo com o Currículo Lattes)

Titulação (Especialização, Mestrado, Doutorado)

Nome completo e por extenso da instituição a qual possui vínculo

Nome completo e por extenso do Membro 3 (de acordo com o Currículo Lattes)

Título (Titulação (Especialização, Mestrado, Doutorado))

Nome completo e por extenso da instituição a qual possui vínculo

CURITIBA

2025

Dedico este trabalho à minha família, pelos
momentos de ausência.

AGRADECIMENTOS

Certamente estes parágrafos não irão atender a todas as pessoas que fizeram parte dessa importante fase de minha vida. Portanto, desde já peço desculpas àquelas que não estão presentes entre essas palavras, mas elas podem estar certas que fazem parte do meu pensamento e de minha gratidão.

Agradeço ao(a) meu(minha) orientador(a) Prof.(a) Dr.(a) Nome Completo, pela sabedoria com que me guiou nesta trajetória.

Aos meus colegas de sala.

A Secretaria do Curso, pela cooperação.

Gostaria de deixar registrado também, o meu reconhecimento à minha família, pois acredito que sem o apoio deles seria muito difícil vencer esse desafio.

Enfim, a todos os que por algum motivo contribuíram para a realização desta pesquisa.

RESUMO

Este trabalho apresenta o desenvolvimento e avaliação de modelos de aprendizado de máquina aplicados à classificação de objetos em linhas de transmissão, utilizando dados de profundidade capturados por uma câmera *RealSense D415*, um sensor LiDAR *RPLIDAR A1* da Slamtec e a combinação de ambos. Foram testados cinco modelos - *k-Nearest Neighbors*, Árvore de Decisões, *Naive Bayes*, Rede Neural e Floresta Aleatória - com dados simulados e reais. Os resultados indicaram que o uso de apenas um dos sensores já permite uma classificação satisfatória, mas a fusão sensorial torna o sistema mais robusto. Os modelos mais leves apresentaram desempenho competitivo, evidenciando seu potencial para futura implementação embarcada. O estudo também explora técnicas de pré-processamento de imagens e dados para a criação de *features* utilizadas no treinamento dos modelos. Destaca-se, assim, a viabilidade do uso de sensores de profundidade na inspeção autônoma de linhas de transmissão, com ganhos em segurança e redução de custos operacionais.

Palavras-chave: aprendizado de máquina; sensores de profundidade; inspeção autônoma; classificação de objetos; linhas de transmissão.

ABSTRACT

Seguir o mesmo padrão do resumo, com a tradução do texto do resumo e referência, se houver, para a língua estrangeira (língua inglesa).

Keywords: machine learning; deep sensors; autonomous inspection; power lines.

LISTA DE FIGURAS

Figura 1 – Objetos de interesse no ambiente de simulação: (a) Amortecedor, (b) Isolador e (c) Sinalizador.	30
Figura 2 – Objetos de interesse no ambiente real: (a) Amortecedor, (b) Isolador e (c) Sinalizador.	31
Figura 3 – Vistas do protótipo do robô principal no ambiente de simulação: (a) Vista frontal, (b) Vista de perfil e (c) Vista superior.	48
Figura 4 – Vistas do protótipo do primeiro robô secundário no ambiente de simulação: (a) Vista frontal, (b) Vista de perfil e (c) Vista superior.	50
Figura 5 – Vistas do protótipo do segundo robô secundário no ambiente de simulação: (a) Vista frontal, (b) Vista de perfil e (c) Vista superior.	50
Figura 6 – Detalhes do sensor multimodal desenvolvido: (a) Suporte com sua dimensão horizontal e (b) Robô principal simulado indicando as distâncias dos sensores em relação ao cabo.	51
Figura 7 – Leituras do sensor LiDAR para diferentes classes no ambiente simulado, com indicação do início e término da detecção do objeto: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).	58
Figura 8 – Leituras do sensor LiDAR para diferentes classes no ambiente real, com indicação do início e término da detecção do objeto: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).	59
Figura 9 – Exemplos de imagens brutas da câmera de profundidade para diferentes classes no ambiente simulado: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).	60
Figura 10 – Exemplos de imagens brutas da câmera de profundidade (histograma equalizado) para diferentes classes no ambiente real: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).	61
Figura 11 – Exemplos de imagens da câmera de profundidade no ambiente real após aplicação do filtro de limite de profundidade: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).	62
Figura 12 – Exemplos do resultado da aplicação do filtro Laplaciano em imagens da câmera de profundidade para diferentes classes no ambiente simulado: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).	63
Figura 13 – Exemplos do resultado da aplicação do filtro Laplaciano em imagens da câmera de profundidade para diferentes classes no ambiente real: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).	64
Figura 14 – Exemplos de imagens binarizadas da câmera de profundidade para diferentes classes no ambiente simulado: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).	65
Figura 15 – Exemplos de imagens binarizadas da câmera de profundidade para diferentes classes no ambiente real: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).	66
Figura 16 – Matriz de confusão do k-Vizinhos mais próximos na dobra 5 com dados brutos do LiDAR (simulação).	91
Figura 17 – Árvore de decisão gerada para a dobra 1 utilizando dados brutos do LiDAR (simulação).	92

Figura 18 – Matriz de confusão do Naive Bayes na dobra 5 com dados brutos do LiDAR (simulação).	93
Figura 19 – Matriz de confusão da Rede Neural na dobra 2 com dados brutos do LiDAR (simulação).	94
Figura 20 – Matriz de confusão do k-Vizinhos mais próximos na dobra 5 com features extraídas do LiDAR (simulação).	97
Figura 21 – Matriz de confusão do Naive Bayes na dobra 5 com features extraídas do LiDAR (simulação).	98
Figura 22 – Matriz de confusão da Rede Neural na dobra 5 com features extraídas do LiDAR (simulação).	99
Figura 23 – Matriz de confusão do k-Vizinhos mais próximos na dobra 1 com dados brutos do LiDAR (real).	102
Figura 24 – Matriz de confusão do Naive Bayes na dobra 2 com dados brutos do LiDAR (real).	103
Figura 25 – Matriz de confusão da Rede Neural na dobra 5 com dados brutos do LiDAR (real).	104
Figura 26 – Matriz de confusão da Rede Neural na dobra 4 com features extraídas das imagens (real).	106
Figura 27 – Matriz de confusão do k-Vizinhos mais próximos na dobra 5 com features extraídas do LiDAR (real).	107
Figura 28 – Matriz de confusão do Naive Bayes na dobra 5 com features extraídas do LiDAR (real).	108
Figura 29 – Matriz de confusão da Rede Neural na dobra 5 com features extraídas do LiDAR (real).	109
Figura 30 – Matriz de confusão da dobra 3 para a Rede Neural com features combinadas (real).	111
Figura 31 – Matriz de confusão do modelo SqueezeNet utilizando imagens brutas (primeiro robô das topologias secundárias).	113
Figura 32 – Matriz de confusão para o k-Vizinhos mais próximos com dados brutos do LiDAR (primeiro robô das topologias secundárias).	114
Figura 33 – Matriz de confusão para a Árvore de Decisão com dados brutos do LiDAR (primeiro robô das topologias secundárias).	114
Figura 34 – Matriz de confusão para o Naive Bayes com dados brutos do LiDAR (primeiro robô das topologias secundárias).	115
Figura 35 – Matriz de confusão para a Rede Neural com dados brutos do LiDAR (primeiro robô das topologias secundárias).	115
Figura 36 – Matriz de confusão para a Floresta Aleatória com dados brutos do LiDAR (primeiro robô das topologias secundárias).	116
Figura 37 – Matriz de confusão para o k-Vizinhos mais próximos com features extraídas das imagens (primeiro robô das topologias secundárias).	117
Figura 38 – Matriz de confusão para a Árvore de Decisão com features extraídas das imagens (primeiro robô das topologias secundárias).	117
Figura 39 – Matriz de confusão para o Naive Bayes com features extraídas das imagens (primeiro robô das topologias secundárias).	118
Figura 40 – Matriz de confusão para a Rede Neural com features extraídas das imagens (primeiro robô das topologias secundárias).	118
Figura 41 – Matriz de confusão para a Floresta Aleatória com features extraídas das imagens (primeiro robô das topologias secundárias).	119
Figura 42 – Matriz de confusão para o k-Vizinhos mais próximos com features extraídas do LiDAR (primeiro robô das topologias secundárias).	120

Figura 43 – Matriz de confusão para a Árvore de Decisão com features extraídas do LiDAR (primeiro robô das topologias secundárias).....	120
Figura 44 – Matriz de confusão para o Naive Bayes com features extraídas do LiDAR (primeiro robô das topologias secundárias).....	121
Figura 45 – Matriz de confusão para a Rede Neural com features extraídas do LiDAR (primeiro robô das topologias secundárias).....	121
Figura 46 – Matriz de confusão para a Floresta Aleatória com features extraídas do LiDAR (primeiro robô das topologias secundárias).....	122
Figura 47 – Matriz de confusão para o k-Vizinhos mais próximos com features combinadas (primeiro robô das topologias secundárias).....	123
Figura 48 – Matriz de confusão para a Árvore de Decisão com features combinadas (primeiro robô das topologias secundárias).....	123
Figura 49 – Matriz de confusão para o Naive Bayes com features combinadas (primeiro robô das topologias secundárias).....	124
Figura 50 – Matriz de confusão para a Floresta Aleatória com features combinadas (primeiro robô das topologias secundárias).....	124
Figura 51 – Matriz de confusão para o k-Vizinhos mais próximos com dados brutos do LiDAR (segundo robô das topologias secundárias)	126
Figura 52 – Matriz de confusão para a Árvore de Decisão com dados brutos do LiDAR (segundo robô das topologias secundárias).....	126
Figura 53 – Matriz de confusão para o Naive Bayes com dados brutos do LiDAR (segundo robô das topologias secundárias)	127
Figura 54 – Matriz de confusão para a Rede Neural com dados brutos do LiDAR (segundo robô das topologias secundárias)	127
Figura 55 – Matriz de confusão para a Floresta Aleatória com dados brutos do LiDAR (segundo robô das topologias secundárias).....	128
Figura 56 – Matriz de confusão para o k-Vizinhos mais próximos com features extraídas das imagens (segundo robô das topologias secundárias)	129
Figura 57 – Matriz de confusão para a Árvore de Decisão com features extraídas das imagens (segundo robô das topologias secundárias)	129
Figura 58 – Matriz de confusão para o Naive Bayes com features extraídas das imagens (segundo robô das topologias secundárias)	130
Figura 59 – Matriz de confusão para a Rede Neural com features extraídas das imagens (segundo robô das topologias secundárias)	130
Figura 60 – Matriz de confusão para a Floresta Aleatória com features extraídas das imagens (segundo robô das topologias secundárias)	131
Figura 61 – Matriz de confusão para o k-Vizinhos mais próximos com features extraídas do LiDAR (segundo robô das topologias secundárias)	132
Figura 62 – Matriz de confusão para a Árvore de Decisão com features extraídas do LiDAR (segundo robô das topologias secundárias)	132
Figura 63 – Matriz de confusão para o Naive Bayes com features extraídas do LiDAR (segundo robô das topologias secundárias)	133
Figura 64 – Matriz de confusão para a Rede Neural com features extraídas do LiDAR (segundo robô das topologias secundárias)	133
Figura 65 – Matriz de confusão para a Floresta Aleatória com features extraídas do LiDAR (segundo robô das topologias secundárias)	134
Figura 66 – Matriz de confusão para o k-Vizinhos mais próximos com features combinadas (segundo robô das topologias secundárias)	135
Figura 67 – Matriz de confusão para a Árvore de Decisão com features combinadas (segundo robô das topologias secundárias)	135

Figura 68 – Matriz de confusão para o Naive Bayes com features combinadas (segundo robô das topologias secundárias).	136
Figura 69 – Matriz de confusão para a Rede Neural com features combinadas (segundo robô das topologias secundárias).	136
Figura 70 – Matriz de confusão para a Floresta Aleatória com features combinadas (segundo robô das topologias secundárias).	137

LISTA DE FOTOGRAFIAS

Fotografia 1 – Fotografia do protótipo do robô principal real: Vista frontal.	48
Fotografia 2 – Fotografia do protótipo do robô principal real: Vista de perfil.	49
Fotografia 3 – Câmera de profundidade Intel RealSense D415 utilizada no projeto...	53
Fotografia 4 – Sensor LiDAR RPLIDAR A1M8 utilizado no projeto.	54

LISTA DE TABELAS

Tabela 1 – Vantagens e Desvantagens dos Sensores para Inspeção de Linhas de Transmissão.....	24
Tabela 2 – Resumo da Acurácia Média e Tempo de Validação Médio por Modelo e Cenário.....	72
Tabela 3 – Acurácia Média e Tempo de Validação Médio por Modelo e Tipo de Dado (Robô Principal - Simulado).....	74
Tabela 4 – Acurácia Média e Tempo de Validação Médio por Modelo e Tipo de Dado (Robô Principal - Real).....	76
Tabela 5 – Acurácia e Tempo de Validação por Modelo e Tipo de Dado (Primeiro Robô Secundário - Multimodal).....	77
Tabela 6 – Acurácia e Tempo de Validação por Modelo e Tipo de Dado (Segundo Robô Secundário - Multimodal).....	78
Tabela 7 – Acurácia Média por Cenário e Tipo de Dado.....	80
Tabela 8 – Desempenho da SqueezeNet com imagens brutas (simulação).....	90
Tabela 9 – Desempenho do k-Vizinhos mais próximos com dados brutos do LiDAR (simulação)	90
Tabela 10 – Desempenho da Árvore de Decisão com dados brutos do LiDAR (simulação)	91
Tabela 11 – Desempenho do Naive Bayes com dados brutos do LiDAR (simulação) ..	92
Tabela 12 – Desempenho da Rede Neural com dados brutos do LiDAR (simulação) ..	93
Tabela 13 – Desempenho da Floresta Aleatória com dados brutos do LiDAR (simulação)	94
Tabela 14 – Desempenho do k-Vizinhos mais próximos com features extraídas das imagens (simulação).....	95
Tabela 15 – Desempenho da Árvore de Decisão com features extraídas das imagens (simulação)	95
Tabela 16 – Desempenho do Naive Bayes com features extraídas das imagens (simulação)	95
Tabela 17 – Desempenho da Rede Neural com features extraídas das imagens (simulação)	96
Tabela 18 – Desempenho da Floresta Aleatória com features extraídas das imagens (simulação)	96
Tabela 19 – Desempenho do k-Vizinhos mais próximos com features extraídas do LiDAR (simulação)	96
Tabela 20 – Desempenho da Árvore de Decisão com features extraídas do LiDAR (simulação)	97
Tabela 21 – Desempenho do Naive Bayes com features extraídas do LiDAR (simulação)	97
Tabela 22 – Desempenho da Rede Neural com features extraídas do LiDAR (simulação)	98
Tabela 23 – Desempenho da Floresta Aleatória com features extraídas do LiDAR (simulação)	99
Tabela 24 – Desempenho do k-Vizinhos mais próximos com features combinadas (simulação)	100
Tabela 25 – Desempenho da Árvore de Decisão com features combinadas (simulação).....	100
Tabela 26 – Desempenho do Naive Bayes com features combinadas (simulação)	100
Tabela 27 – Desempenho da Rede Neural com features combinadas (simulação)	101

Tabela 28 – Desempenho da Floresta Aleatória com features combinadas (simulação)	101
Tabela 29 – Desempenho da SqueezeNet com imagens brutas (real)	101
Tabela 30 – Desempenho do k-Vizinhos mais próximos com dados brutos do LiDAR (real)	102
Tabela 31 – Desempenho do Árvore de Decisão com dados brutos do LiDAR (real)....	103
Tabela 32 – Desempenho do Naive Bayes com dados brutos do LiDAR (real)	103
Tabela 33 – Desempenho da Rede Neural com dados brutos do LiDAR (real).....	104
Tabela 34 – Desempenho da Floresta Aleatória com dados brutos do LiDAR (real)....	104
Tabela 35 – Desempenho do k-Vizinhos mais próximos com features extraídas das imagens (real).	105
Tabela 36 – Desempenho da Árvore de Decisão com features extraídas das imagens (real).....	105
Tabela 37 – Desempenho do Naive Bayes com features extraídas das imagens (real). 105	105
Tabela 38 – Desempenho da Rede Neural com features extraídas das imagens (real). 106	106
Tabela 39 – Desempenho da Floresta Aleatória com features extraídas das imagens (real).....	106
Tabela 40 – Desempenho do k-Vizinhos mais próximos com features extraídas do LiDAR (real).....	107
Tabela 41 – Desempenho da Árvore de Decisão com features extraídas do LiDAR (real).....	108
Tabela 42 – Desempenho do Naive Bayes com features extraídas do LiDAR (real)....	108
Tabela 43 – Desempenho da Rede Neural com features extraídas do LiDAR (real)....	109
Tabela 44 – Desempenho da Floresta Aleatória com features extraídas do LiDAR (real).....	109
Tabela 45 – Desempenho do k-Vizinhos mais próximos com features combinadas (real).....	110
Tabela 46 – Desempenho do Naive Bayes com features combinadas (real).....	110
Tabela 47 – Desempenho do Naive Bayes com features combinadas (real).....	110
Tabela 48 – Desempenho da Rede Neural com features combinadas (real).....	111
Tabela 49 – Desempenho da Floresta Aleatória com features combinadas (real).....	111
Tabela 50 – Desempenho da SqueezeNet com imagens brutas (robô 1 das topologias secundárias).....	112
Tabela 51 – Comparativo de desempenho entre diferentes modelos.....	113
Tabela 52 – Desempenho dos modelos com features extraídas das imagens (primeiro robô das topologias secundárias).....	116
Tabela 53 – Desempenho dos modelos com features extraídas do LiDAR (primeiro robô das topologias secundárias).....	119
Tabela 54 – Desempenho dos modelos com features combinadas (primeiro robô das topologias secundárias)	122
Tabela 55 – Desempenho da SqueezeNet com imagens brutas (robô 1 das topologias secundárias).....	125
Tabela 56 – Desempenho dos modelos com dados brutos do LiDAR (segundo robô das topologias secundárias)	125
Tabela 57 – Desempenho dos modelos com features extraídas das imagens (segundo robô das topologias secundárias).....	128
Tabela 58 – Desempenho dos modelos com features extraídas do LiDAR (segundo robô das topologias secundárias).....	131
Tabela 59 – Desempenho dos modelos com features combinadas (segundo robô das topologias secundárias)	134

SUMÁRIO

1	INTRODUÇÃO	19
1.1	Inspeções e Classificação de Objetos em Linhas de Transmissão	19
1.2	Justificativa.....	20
1.3	Objetivos.....	20
1.3.1	Objetivos Gerais	20
1.3.2	Objetivos Específicos	20
2	REVISÃO BIBLIOGRÁFICA	22
3	METODOLOGIA	26
3.1	Ambiente de Desenvolvimento e Simulação	26
3.1.1	Robot Operating System.....	26
3.1.1.1	Ferramentas do ROS.....	27
3.1.2	CoppeliaSim	28
3.1.3	Ambiente Híbrido	29
3.2	Aquisição de Dados	29
3.2.1	Classes de Objetos a Serem Detectadas	30
3.2.2	Sensores utilizados	30
3.2.3	Planejamento da Coleta de Dados	31
3.2.4	Organização e Formato dos Dados Coletados	32
3.2.5	Ferramentas e Scripts de Coleta.....	32
3.2.6	Pré-processamento Inicial	33
3.2.7	Considerações Éticas e Reprodutibilidade.....	33
3.3	Construção das Cenas Simuladas	34
3.3.1	Ambiente Virtual Criado no CoppeliaSim	34
3.3.2	Robô Simulado.....	35
3.3.3	Integração com ROS	36
3.3.4	Protocolo de Coleta na Simulação.....	36
3.3.5	Limitações e Considerações sobre a Simulação	37
3.4	Construção das Cenas no Laboratório.....	37
3.4.1	Objetivos da Validação em Ambiente Real	38
3.4.2	Infraestrutura do Laboratório	38
3.4.3	Coleta com Robô em Cabo Suspens	38
3.4.3.1	Estrutura do cabo suspenso.....	39
3.4.4	Integração com ROS	39
3.4.5	Protocolo de Coleta no Laboratório.....	40
3.4.6	Comparação com a Simulação.....	41
3.5	Registro e Gestão dos Dados	41
3.6	Treinamento e Avaliação dos Modelos	42
3.6.1	Objetivo da Etapa de Treinamento	42
3.6.2	Arquiteturas e Algoritmos Testados	42
3.6.2.1	k-Vizinhos mais proximos	43
3.6.2.2	Árvore de Decisões	43
3.6.2.3	Naive Bayes	43
3.6.2.4	Floresta Aleatória	43
3.6.2.5	Perceptron Multicamadas	44
3.6.2.6	Rede Neural Convolucional – SqueezeNet	44
3.6.3	Treinamento dos Modelos	44
3.6.4	Avaliação de Desempenho	45

3.6.5	Limitações da Metodologia	45
4	DESENVOLVIMENTO.....	47
4.1	Hardware	47
4.1.1	Estrutura Mecânica dos Robôs	47
4.1.2	Sensor Multimodal	49
4.1.3	Sensores Utilizados	52
4.1.3.1	Intel RealSense D415	52
4.1.3.2	RPLIDAR A1M8.....	52
4.2	Firmware e Software.....	53
4.2.1	Controle do Robô via ROS	54
4.2.2	Configuração dos Sensores	55
4.2.3	Script de Coleta	55
4.3	Coleta dos Dados	56
4.4	Processamento de Dados	56
4.4.1	Dados do LiDAR	57
4.4.2	Dados da RealSense.....	59
4.4.2.1	Limite de profundidade.....	60
4.4.2.2	Preparação das imagens para a rede SqueezeNet.....	61
4.4.2.3	Contorno dos objetos	62
4.4.2.4	Operações morfológicas	63
4.5	Treinamento e Validação	65
4.5.1	Configuração dos Modelos	66
4.5.1.1	Árvore de Decisão	67
4.5.1.2	<u>k-Vizinhos mais próximos</u>	67
4.5.1.3	Naive Bayes	67
4.5.1.4	Random Forest.....	67
4.5.1.5	Rede Neural	68
4.5.1.6	SqueezeNet	68
4.5.2	Validação	68
5	RESULTADOS E DISCUSSÕES	70
5.1	Comparativo Geral do Desempenho dos Modelos.....	71
5.1.1	Análise Detalhada no Cenário Principal Simulado	74
5.1.2	Análise Detalhada no Cenário Principal Real	75
5.1.3	Análise dos Testes de Portabilidade no Primeiro Robô Secundário	76
5.1.4	Análise dos Testes de Portabilidade no Segundo Robô Secundário	78
5.2	Análise da Influência do Tipo de Dado	79
5.3	Análise das Matrizes de Confusão	81
6	CONCLUSÕES	84
	REFERÊNCIAS.....	85
	APÊNDICE A RESULTADOS	89
	A.1 Resultados com o robô principal	89
A.1.1	Dados simulados	89
A.1.1.1	Imagens brutas	90
A.1.1.2	Dados brutos do LiDAR	90
A.1.1.2.1	<i>k-Vizinhos mais próximos</i>	90
A.1.1.2.2	<i>Árvore de Decisão</i>	91
A.1.1.2.3	<i>Naive Bayes</i>	92
A.1.1.2.4	<i>Rede Neural</i>	93

A.1.1.2.5	<i>Floresta Aleatória</i>	94
A.1.1.3	Features extraídas das imagens	94
A.1.1.3.1	<i>k-Vizinhos mais próximos</i>	95
A.1.1.3.2	<i>Árvore de Decisão</i>	95
A.1.1.3.3	<i>Naive Bayes</i>	95
A.1.1.3.4	<i>Rede Neural</i>	96
A.1.1.3.5	<i>Floresta Aleatória</i>	96
A.1.1.4	Features extraídas do LiDAR	96
A.1.1.4.1	<i>k-Vizinhos mais próximos</i>	96
A.1.1.4.2	<i>Árvore de Decisão</i>	97
A.1.1.4.3	<i>Naive Bayes</i>	97
A.1.1.4.4	<i>Rede Neural</i>	98
A.1.1.4.5	<i>Floresta Aleatória</i>	99
A.1.1.5	Features combinadas	99
A.1.1.5.1	<i>k-Vizinhos mais próximos</i>	100
A.1.1.5.2	<i>Árvore de Decisão</i>	100
A.1.1.5.3	<i>Naive Bayes</i>	100
A.1.1.5.4	<i>Rede Neural</i>	101
A.1.1.5.5	<i>Floresta Aleatória</i>	101
A.1.2	Dados reais	101
A.1.2.1	Imagens brutas	101
A.1.2.2	Dados brutos do LiDAR	102
A.1.2.2.1	<i>k-Vizinhos mais próximos</i>	102
A.1.2.2.2	<i>Árvore de Decisão</i>	103
A.1.2.2.3	<i>Naive Bayes</i>	103
A.1.2.2.4	<i>Rede Neural</i>	104
A.1.2.2.5	<i>Floresta Aleatória</i>	104
A.1.2.3	Features extraídas das imagens	105
A.1.2.3.1	<i>k-Vizinhos mais próximos</i>	105
A.1.2.3.2	<i>Árvore de Decisão</i>	105
A.1.2.3.3	<i>Naive Bayes</i>	105
A.1.2.3.4	<i>Rede Neural</i>	106
A.1.2.3.5	<i>Floresta Aleatória</i>	106
A.1.2.4	Features extraídas do LiDAR	107
A.1.2.4.1	<i>k-Vizinhos mais próximos</i>	107
A.1.2.4.2	<i>Árvore de Decisão</i>	108
A.1.2.4.3	<i>Naive Bayes</i>	108
A.1.2.4.4	<i>Rede Neural</i>	109
A.1.2.4.5	<i>Floresta Aleatória</i>	109
A.1.2.5	Features combinadas	110
A.1.2.5.1	<i>k-Vizinhos mais próximos</i>	110
A.1.2.5.2	<i>Árvore de Decisão</i>	110
A.1.2.5.3	<i>Naive Bayes</i>	110
A.1.2.5.4	<i>Rede Neural</i>	111
A.1.2.5.5	<i>Floresta Aleatória</i>	111
	A.2 Resultados com os robôs secundários	112
A.2.1	Dados do primeiro robô	112
A.2.1.1	Imagens brutas	112
A.2.1.2	Dados brutos do LiDAR	113
A.2.1.3	Features extraídas das imagens	116

A.2.1.4	Features extraídas do LiDAR	119
A.2.1.5	Features combinadas	122
A.2.2	Dados do segundo robô	124
A.2.2.1	Imagens brutas	125
A.2.2.2	Dados brutos do LiDAR	125
A.2.2.3	Features extraídas das imagens	128
A.2.2.4	Features extraídas do LiDAR	131
A.2.2.5	Features combinadas	134

1 INTRODUÇÃO

A crescente demanda por eficiência e segurança na manutenção de sistemas elétricos de potência impulsiona a busca por soluções inovadoras para a inspeção de linhas de transmissão. Tradicionalmente, tais inspeções envolvem métodos manuais ou o uso de drones, que apresentam limitações significativas em termos de custo, risco, autonomia e precisão. Neste contexto, sistemas robóticos embarcados, capazes de percorrer os cabos e realizar inspeções detalhadas de forma autônoma, surgem como uma alternativa promissora. A utilização de sensores de profundidade, como câmeras de profundidade e LiDARs, acoplados a esses robôs, permite a captura de informações tridimensionais ricas do ambiente, essenciais para a identificação e classificação de componentes e anomalias.

Este trabalho de conclusão de curso se dedica a investigar a aplicação de técnicas de aprendizado de máquina para a classificação de objetos comumente encontrados em linhas de transmissão — especificamente sinalizadores, amortecedores e isoladores — utilizando dados provenientes de uma câmera de profundidade RealSense D415 e um sensor LiDAR RPLIDAR A1. A pesquisa explora diferentes abordagens de processamento de dados, incluindo o uso de informações brutas dos sensores, a extração de *features* e a combinação de dados de ambas as fontes, com foco na viabilidade de implementação em sistemas embarcados. A análise comparativa do desempenho de diversos modelos de classificação em cenários simulados e reais visa fornecer *insights* valiosos sobre a eficácia e robustez desta abordagem para a automação da inspeção em linhas de transmissão.

Este capítulo introdutório estabelece o panorama da inspeção e classificação de objetos em linhas de transmissão, detalha a justificativa para a pesquisa, e define os objetivos gerais e específicos que nortearam o desenvolvimento do projeto.

1.1 Inspeções e Classificação de Objetos em Linhas de Transmissão

As inspeções periódicas em linhas de transmissão de alta tensão, pertencentes ao Sistema Elétrico de Potência (SEP), são essenciais para a manutenção da confiabilidade e segurança de todo o sistema, evitando acidentes e interrupções no fornecimento de energia elétrica. Essas estruturas operam em ambientes adversos e estão sujeitas a danos causados por condições climáticas extremas e objetos estranhos na fiação.

Existem diferentes métodos com finalidades específicas para a realização da inspeção, seja de modo preventivo ou corretivo. Tradicionalmente, o serviço é executado por equipes que escalam torres ou utilizam drones. Embora drones cubram grandes áreas com agilidade e baixo custo, enfrentam limitações como autonomia reduzida, dependência de visibilidade e menor precisão em detalhes complexos. Já a inspeção manual, apesar de precisa, apresenta altos custos, riscos operacionais, riscos de vida e baixa escalabilidade.

Com o avanço das tecnologias de inspeção, os métodos tradicionais aplicados em instalações de energia têm se mostrado insuficientes para atender às demandas atuais do setor,

principalmente devido à baixa eficiência das inspeções manuais, aos altos riscos envolvidos e à detecção limitada de falhas potenciais. Nesse cenário, observa-se um crescente interesse na automação dessas tarefas por meio de sistemas robóticos embarcados, capazes de percorrer fisicamente os cabos das linhas de transmissão e realizar inspeções de forma autônoma e contínua. Tais sistemas podem ser equipados com sensores capazes de capturar informações detalhadas dos componentes da linha, mesmo em condições adversas. Entre os sensores, destacam-se as câmeras de profundidade, como a RealSense D415, e os sensores LiDAR, como o RPLIDAR A1, que permitem a obtenção de dados tridimensionais da cena com alta precisão.

Este trabalho investiga o uso de modelos de aprendizado de máquina para a classificação de objetos em linhas de transmissão, com dados de sensores em ambientes simulados e reais. A análise considera dados da RealSense, do LiDAR e a combinação dos dois, com foco na eficiência, no processamento embarcado e em técnicas de pré-processamento e extração de *features*. Os objetos de interesse foram alguns dos mais populares presentes nas linhas: sinalizadores, amortecedores e isoladores.

1.2 Justificativa

Para que robôs autônomos de inspeção possam realizar rotinas de superação de obstáculos, existe a necessidade de classificação dos objetos presentes em linhas de transmissão. A identificação assertiva possibilita que o sistema opere de forma mais adequada àquela situação, uma vez que cada objeto possui diferentes dimensões, pontos de fixação e funcionalidades. Este trabalho consiste na interpretação de dados recebidos pelos sensores e seus respectivos tratamentos via modelos de aprendizado de máquina, buscando comparações de resultados e desempenho.

1.3 Objetivos

1.3.1 Objetivos Gerais

Avaliar o uso de sensores LiDAR 360º e RealSense para classificação de objetos em linhas de transmissão e avaliar os resultados com base no desempenho medido por métricas como acurácia, tempo de processamento e quantidade de informações utilizadas.

1.3.2 Objetivos Específicos

Desenvolvimento de Códigos de Configuração e Acionamento dos Sensores: Criar os scripts necessários para configurar e acionar os sensores, utilizando Python para controlar a

captura dos dados. O processo envolverá o desenvolvimento de códigos para coletar imagens de profundidade e dados do LiDAR, com armazenamento em arquivos CSV e imagens.

Sensor multimodal: Desenvolver o projeto com a flexibilidade necessária para permitir sua integração em diferentes topologias, incluindo robôs terrestres, aéreos e aqueles que se locomovem diretamente no cabo de linhas de transmissão.

Montagem de Cenas Simuladas no CoppeliaSim: Criar cenas simuladas no Coppelia-Sim, onde os sensores serão acoplados ao robô, que percorrerá a linha de transmissão. O robô será posicionado entre duas torres, com um cabo rígido esticado entre elas, e os sensores de profundidade e LiDAR 360º serão usados para coletar dados enquanto o robô se move ao longo do cabo. As cenas levarão em consideração o movimento do robô, com acionamento dos motores para simular a coleta de dados em tempo real.

Montagem de Ambientes Controlados no Laboratório: Construir ambientes controlados no laboratório, com os sensores presos a mecanismos que simulem diferentes ângulos e distâncias, onde os sensores não estarão no robô. Além disso, será montada uma pequena cena com um cabo pendurado, contendo objetos encontrados em linhas de alta tensão onde o robô será preso ao cabo, com os sensores fixados a ele, e percorrerá o trajeto enquanto coleta dados sobre os objetos presentes no cabo.

Treinamento de Modelos de Aprendizado de Máquina: Utilizar os dados coletados para treinar diferentes modelos de aprendizado de máquina, com o objetivo de avaliar a capacidade dos modelos em classificar os objetos presentes nas linhas de transmissão. Para isso, serão montados diferentes datasets, considerando separadamente as informações do LiDAR, da câmera de profundidade e a combinação de ambos, permitindo a análise do impacto de cada sensor na performance dos modelos.

Avaliação de Desempenho: Avaliar o desempenho dos modelos treinados, utilizando métricas de performance como acurácia, precisão, recall e tempo de processamento para comparar os resultados obtidos pelos diferentes sensores e modelos.

2 REVISÃO BIBLIOGRÁFICA

A inspeção de linhas de transmissão de alta tensão tem sido um campo de pesquisa ativo, com diversos trabalhos focando no desenvolvimento de sistemas robóticos capazes de realizar essa tarefa de forma mais eficiente e segura do que os métodos tradicionais. Esta revisão aborda a evolução desses robôs, desde os modelos operados remotamente até as abordagens mais recentes que incorporam diferentes tipos de sensores e técnicas de aprendizado de máquina para a detecção e classificação de objetos.

Os primeiros robôs projetados para a inspeção de linhas de transmissão dependiam fortemente de operadores humanos. Um exemplo inicial, proposto por Fonseca, Abdo e Alberto (2012), explorou inicialmente um motor a combustão, rapidamente descartado pela inviabilidade de operação segura em redes energizadas. A versão subsequente, com motores elétricos, possuía dois braços, sem mecanismos de trava ao cabo ou de superação de obstáculos, sendo controlada via rádio e equipada com uma câmera RGB para inspeção visual.

A necessidade de superar obstáculos impulsionou o desenvolvimento de mecanismos mais sofisticados. Um conceito recente abordando a superação de obstáculos foi proposto por Xiao *et al.* (2024), no qual um robô equipado com dois conjuntos de rodas e dois braços mecânicos permitia a travessia de obstáculos verticais curtos. Já o robô apresentado por Yue, Wang e Jiang (2017) foi projetado para escalar *jumpers* com inclinações de até 80°, utilizando um mecanismo de locomoção similar a uma lagarta e uma garra de suporte, embora ainda dependesse de navegação manual. Outro robô com capacidade de superar obstáculos foi descrito por Qing *et al.* (2016), possuindo múltiplas unidades mecânicas para locomoção, fixação e rotação; contudo, cada etapa do deslocamento e da travessia de obstáculos exigia controle manual, limitando sua eficiência operacional. Alguns robôs também foram desenvolvidos para tarefas específicas, como o modelo de Jiang *et al.* (2019) com dois braços para substituição de amortecedores, mas sua operação permanecia dependente de controle humano.

A incorporação de múltiplos sensores visou aumentar a eficiência e autonomia dos robôs. Fan *et al.* (2018) apresentaram um sistema com sensores RGB, infravermelho e LiDAR para coletar dados e superar obstáculos, utilizando os sensores ópticos para análise estrutural e o LiDAR para mapeamento ambiental e de vegetação. Apesar do potencial autônomo, o robô ainda requeria uma equipe para operação. Qin *et al.* (2017) desenvolveram um robô com LiDAR para criar nuvens de pontos 3D da linha, permitindo modelagem detalhada e detecção de deformações e objetos.

Paralelamente ao desenvolvimento de robôs, diversas pesquisas focaram na classificação de objetos utilizando os dados sensoriais. Peters, Ahn e Borkowski (2002) apresentaram um robô rastejador que utilizava um conjunto simplificado de quatro sensores de proximidade (derivados de 34 sensores distribuídos) para detectar obstáculos e ativar rotinas de superação. O trabalho propôs o uso de redes neurais para lidar com interferências eletromagnéticas e manter a precisão das medições. Lima, Bomfim e Mourão (2018), no projeto POLIBOT, propuseram

um robô leve que utilizava um sensor LiDAR e um sensor ultrassônico para detectar obstáculos e calcular ângulos de atuação para navegação.

O trabalho de Pouliot, Richard e Montambault (2012) caracterizou o desempenho do sensor LiDAR UTM-30LX, posicionado na parte inferior de um robô e varrendo a linha em um ângulo de 45° para detectar objetos, identificando suas bordas para estimar diâmetros e distâncias, com metologia similar a aplicada nesse projeto para o sensor. Esta abordagem foi posteriormente implementada e testada em linha real (Richard; Pouliot; Montambault, 2014), servindo de inspiração para o posicionamento do sensor LiDAR e extração de características geométricas no presente trabalho. Contudo, o estudo não implementou um modelo para classificação automática.

Outra técnica utilizando LiDAR envolve a geração de nuvens de pontos é apresentada no trabalho de Qin *et al.* (2018), onde usaram este método para classificar objetos com base na distribuição espacial dos pontos e segmentação por crescimento de região 3D, alcançando 90,6% de acurácia. O robô utilizado é o mesmo de Qin *et al.* (2017), Qin *et al.* (2018). Wang *et al.* (2018) compararam métodos de classificação de linhas de transmissão a partir de nuvens de pontos de LiDAR (ALS e MLS), onde Random Forest e Redes Neurais obtiveram os melhores resultados, com até 98,5% de taxa de qualidade.

Sensores RGB também são comuns nesse contexto. Song *et al.* (2015) propuseram um método de visão computacional para detectar espaçadores quebrados, utilizando transformações morfológicas e análise de conexões na imagem. Zhu, Wang e Xu (2016) usaram visão computacional em um robô para detectar amortecedores, espaçadores e braçadeiras, aplicando transformações e técnicas morfológicas, e classificando com uma SVM estrutural, atingindo acurárias entre 92,67% e 96% para diferentes objetos.

A Tabela 1 resume as principais vantagens e desvantagens dos tipos de sensores abordados neste contexto.

Tabela 1 – Vantagens e Desvantagens dos Sensores para Inspeção de Linhas de Transmissão.

SENSOR	VANTAGENS	DESVANTAGENS
RGB	Capacidade de diferenciar objetos com base em cores e texturas (Zhu; Wang; Xu, 2016; Fan <i>et al.</i> , 2018). Boa resolução espacial (Zhu; Wang; Xu, 2016). Fácil integração com modelos de visão computacional (Zhu; Wang; Xu, 2016).	Sensível a variações de iluminação (Zhu <i>et al.</i> , 2024). Dificuldade em diferenciar objetos com cores semelhantes. Não fornece informações de profundidade diretamente (Peters; Ahn; Borkowski, 2002).
LiDAR com nuvem de pontos	Alta precisão na determinação da posição e forma (Qin <i>et al.</i> , 2018; Qin <i>et al.</i> , 2017). Funciona independentemente da iluminação (Pouliot; Richard; Montambault, 2012). Permite reconstrução 3D detalhada (Qin <i>et al.</i> , 2017). Facilita identificação de obstáculos e variações estruturais (Qin <i>et al.</i> , 2017).	Processamento pode ser computacionalmente intensivo (Wang <i>et al.</i> , 2018). Pode exigir pré-processamento complexo para ruído e segmentação (Wang <i>et al.</i> , 2018).
LiDAR sem nuvem de pontos (varredura)	Menor consumo de memória e processamento (Pouliot; Richard; Montambault, 2012). Boa precisão para detectar presença e distância (Pouliot; Richard; Montambault, 2012; Lima; Bomfim; Mourão, 2018). Funciona independentemente da iluminação (Pouliot; Richard; Montambault, 2012).	Pode ser insuficiente para diferenciar objetos com formatos semelhantes (Pouliot; Richard; Montambault, 2012). Detecção limitada a poucos pontos, dificultando identificação de detalhes finos (Pouliot; Richard; Montambault, 2012).
Câmera de Profundidade	Oferece visualização 3D sem processamento pesado de nuvem de pontos LiDAR. Funciona bem em baixa iluminação (modelos ativos).	Qualidade da profundidade afetada por superfícies refletivas ou muito escuras. Alcance geralmente menor que LiDAR. Menor precisão que nuvem de pontos LiDAR para detalhes finos.

Fonte: Baseado em (Zhu; Wang; Xu, 2016; Fan *et al.*, 2018; Zhu *et al.*, 2024; Peters; Ahn; Borkowski, 2002; Qin *et al.*, 2018; Qin *et al.*, 2017; Pouliot; Richard; Montambault, 2012; Lima; Bomfim; Mourão, 2018; Wang *et al.*, 2018) e análise do autor.

Trabalhos mais recentes continuam a explorar novas topologias robóticas e métodos de sensoriamento. O *RaccoonBot* (Mendez-flores; Pourshahidi; Egerstedt, 2025) é um robô autônomo para monitoramento ambiental, alimentado por energia solar e projetado para se apoiar em dois cabos, visando maior estabilidade. Outro estudo recente foca na análise da estabilidade de movimento de robôs do tipo *straddle* para manutenção em linhas de distribuição, testando cenários operacionais com estratégias de equilíbrio e aderência (Cheng *et al.*, 2025). Em relação à classificação, o trabalho de Zhu *et al.* (2024) propõe um robô com sistema visual em plataforma móvel de dois eixos, utilizando YOLOv3 para classificar componentes, alcançando 94,26% de acurácia sob luz visível e analisando o impacto da iluminação e velocidade. Xiao *et al.* (2024) desenvolveram um robô de inspeção de segurança para campus baseado no algoritmo YOLO. Domingues *et al.* (2024) apresentam o desenvolvimento de uma garra robótica

para inspeção confiável de linhas de transmissão, detalhando desde a simulação até a materialização do protótipo real equipado com câmera e LiDAR.

3 METODOLOGIA

Este capítulo apresenta a metodologia adotada para o desenvolvimento e validação do sistema proposto neste trabalho. O estudo foi estruturado em duas etapas principais: simulação e implementação real. A simulação foi realizada utilizando o ambiente CoppeliaSim integrado ao Robot Operating System (ROS), permitindo a criação de cenários controlados para a coleta inicial de dados e testes com diferentes configurações robóticas. Na etapa real, os dados foram coletados em um ambiente físico montado no laboratório, onde o sistema foi submetido a condições práticas semelhantes às encontradas em linhas de transmissão reais. Além disso, são descritos os processos de aquisição, armazenamento e pré-processamento dos dados provenientes de sensores LiDAR e câmera de profundidade, bem como a construção das cenas simuladas e físicas, os modelos e algoritmos de aprendizado de máquina utilizados no treinamento e avaliação do sistema. Essa abordagem híbrida entre simulação e experimentação física visa garantir robustez, reproduzibilidade e aplicabilidade prática dos resultados obtidos ao longo do projeto.

3.1 Ambiente de Desenvolvimento e Simulação

Nesta seção, são descritas as ferramentas e tecnologias utilizadas para a construção do ambiente de simulação e desenvolvimento do projeto. O foco está na criação de uma infraestrutura que permita tanto a comunicação entre sensores e o robô quanto a simulação realista das operações. A subseção está dividida em três partes: introdução ao ROS (Robot Operating System), introdução ao CoppeliaSim, e justificativa do uso de um ambiente híbrido.

3.1.1 Robot Operating System

O *Robot Operating System* é um middleware de código aberto amplamente utilizado no desenvolvimento de sistemas robóticos. Longe de ser um sistema operacional tradicional, o ROS funciona como uma camada de abstração que oferece um conjunto de bibliotecas e ferramentas para simplificar a criação de softwares robóticos complexos. Ele fornece funcionalidades essenciais como gerenciamento de pacotes, comunicação entre processos (nós), manipulação de sensores e atuadores, visualização de dados e integração com ambientes de simulação (Quigley *et al.*, 2009). Neste projeto foi adotada a distribuição **ROS Noetic Ninjemys**, que é a última versão de longo suporte (LTS) da linha ROS 1 e é compatível com o sistema operacional **Ubuntu 20.04**. A necessidade de uma versão específica do sistema operacional se dá pelo fato de que o ROS é altamente dependente do ambiente Linux e das versões específicas das bibliotecas de sistema, que variam entre distribuições e versões do Ubuntu. Isso garante estabilidade e compatibilidade com os pacotes oficiais mantidos pela comunidade. A escolha do ROS Noetic também se justifica pelo alinhamento com as plataformas robóticas consideradas no projeto, as quais já utilizam o ROS como base para controle de seus elementos, permitindo uma integração

mais fluida e reduzindo o esforço de configuração e implementação de funcionalidades. Além disso, o ecossistema do ROS conta com uma vasta gama de bibliotecas e drivers desenvolvidos e mantidos pela comunidade, oferecendo suporte a diversos sensores, como o LiDAR (Light Detection and Ranging) e câmeras de profundidade, utilizados neste trabalho. Essa modularidade e flexibilidade tornam-o uma escolha ideal para o desenvolvimento de aplicações autônomas e simulações realistas em ambientes robóticos.

No ROS, a arquitetura é baseada em uma estrutura distribuída composta por nós (*nodes*) que se comunicam entre si por meio de tópicos (*topics*). Cada nó pode atuar como publicador (*publisher*) ou assinante (*subscriber*), permitindo a troca de mensagens de forma assíncrona. Essa estrutura possibilita uma comunicação modular e flexível entre diferentes componentes do sistema robótico. No contexto deste projeto, foram definidos nós específicos para realizar o controle, aquisição, armazenamento e simulação dos dados. Um dos tópicos foi dedicado à configuração dos sensores, no qual os próprios sensores atuam como assinantes e recebem parâmetros de operação — como limites de leitura e frequência de varredura — a partir de um nó publicador controlado pelo usuário. Paralelamente, os sensores também atuam como publicadores em seus respectivos tópicos de dados, transmitindo continuamente as informações captadas. Esses tópicos são monitorados por nós assinantes responsáveis por armazenar os dados em arquivos locais, para posterior análise e uso no treinamento dos modelos de classificação. Essa organização garante a separação de responsabilidades e facilita a manutenção e expansão do sistema.

3.1.1.1 Ferramentas do ROS

O funcionamento do ROS depende da execução do *roscore*, um serviço mestre responsável por gerenciar a comunicação entre os nós. O *roscore* atua como um servidor central que permite o registro e a descoberta de nós, tópicos e serviços, garantindo que os nós publicadores e assinantes consigam localizar-se mutuamente em tempo de execução. Após sua inicialização, os demais componentes do sistema podem ser executados de forma distribuída. A estrutura de pacotes do ROS organiza os scripts e arquivos de configuração em unidades reutilizáveis. Para executar scripts localizados em pacotes, utiliza-se o comando *rosrun*, que permite a execução direta de nós sem necessidade de navegar até seus diretórios. Para monitorar, depurar e inspecionar os dados que circulam entre os nós, a ferramenta *rostopic* é amplamente utilizada. Com ela, é possível listar tópicos ativos, verificar suas taxas de publicação e até mesmo visualizar em tempo real o conteúdo das mensagens publicadas.

O armazenamento dos dados publicados em tópicos é feito com o uso do *rosbag*, uma ferramenta que grava todas as mensagens trafegadas por tópicos especificados, criando arquivos reutilizáveis em análises futuras ou no treinamento de modelos. Para fins de visualização, o ambiente gráfico *RViz* é empregado, permitindo representar em tempo real a leitura de sensores como câmeras de profundidade e LiDARs, além da posição e orientação do robô no ambiente. O *RViz* se comunica diretamente com os tópicos do ROS, proporcionando uma visualização

clara e precisa dos dados sensoriais e do estado geral do sistema. Embora o *rosbag* seja uma ferramenta poderosa para o registro direto das mensagens trocadas nos tópicos, ele não foi utilizado neste projeto devido à necessidade de pré-processamento imediato dos dados brutos obtidos pelos sensores. Para isso, foram desenvolvidos nós dedicados que se inscrevem nos tópicos dos sensores e executam rotinas personalizadas de filtragem, formatação e armazenamento dos dados. Isso permitiu maior controle sobre a estrutura e organização das informações coletadas, facilitando a etapa posterior de treinamento dos modelos de aprendizado de máquina e possibilitando a integração com bibliotecas externas de processamento.

3.1.2 CoppeliaSim

O CoppeliaSim, anteriormente conhecido como V-REP (Virtual Robot Experimentation Platform), é um ambiente de simulação 3D amplamente utilizado na área de robótica para o desenvolvimento e teste de sistemas autônomos. Segundo Rohmer, Singh e Freese (2013), o CoppeliaSim se destaca por sua flexibilidade, escalabilidade e suporte a múltiplos paradigmas de controle, permitindo que scripts sejam executados de forma embutida ou remota, e que interações com softwares externos ocorram de maneira transparente por meio de interfaces como o ROS. A escolha do CoppeliaSim neste projeto se justifica por diversas vantagens. Entre elas, destaca-se a ampla biblioteca de modelos de robôs e sensores disponibilizados pela comunidade, o que facilita a adaptação e acelera o processo de prototipagem. Além disso, já existem projetos específicos relacionados à inspeção de linhas de transmissão de alta tensão desenvolvidos no simulador, possibilitando avaliar a portabilidade do sensor multimodal deste trabalho em diferentes topologias robóticas. Dessa forma, o CoppeliaSim proporciona um ambiente ideal tanto para a simulação realista do comportamento dos sensores quanto para testes de integração com diferentes configurações robóticas.

Para permitir a comunicação entre o ambiente de simulação CoppeliaSim e o middleware ROS, foi utilizado o plugin ROS Interface que fornece uma ponte bidirecional entre o simulador e o sistema operacional robótico, possibilitando que entidades simuladas no CoppeliaSim (como sensores, atuadores e o próprio robô) publiquem e assinem tópicos ROS em tempo real. Essa integração é essencial para manter a consistência entre os ambientes de simulação e implementação real, pois permite que os mesmos scripts e algoritmos de controle e aquisição de dados sejam reaproveitados em ambos os contextos. A comunicação entre o CoppeliaSim e o ROS é configurada por meio de scripts escritos em linguagem Lua, executados diretamente no simulador. Esses scripts utilizam funções específicas da API simROS para publicar mensagens em tópicos ROS, como dados de sensores LiDAR ou câmeras de profundidade, além de receber comandos de controle vindos de nós externos no ROS. Através dessa abordagem, foi possível simular o comportamento dos sensores e do robô com precisão, enquanto os dados gerados na simulação eram processados e armazenados por scripts no ROS, exatamente como será feito na aplicação real.

3.1.3 Ambiente Híbrido

O uso inicial da simulação se mostrou fundamental para a validação preliminar do sistema proposto, especialmente no que diz respeito às configurações e posicionamento dos sensores. Através do ambiente virtual, foi possível testar diferentes posições e suportes para os sensores LiDAR e câmera de profundidade, observando sua viabilidade prática e identificando possíveis impedimentos impostos pela geometria e topologia dos robôs utilizados. Além disso, a simulação oferece uma série de benefícios importantes: a **repetibilidade** garante que os experimentos possam ser reproduzidos com precisão, facilitando a comparação de resultados entre diferentes configurações; o **controle de variáveis** permite isolar fatores específicos, como distância entre objetos e velocidade de deslocamento do robô, assegurando uma avaliação mais precisa do desempenho dos sensores e algoritmos; por fim, a **redução de riscos** evita danos ao hardware e acidentes durante a fase inicial de testes, que poderiam comprometer o cronograma e os recursos do projeto. Dessa forma, o ambiente simulado atua como uma etapa crucial de validação antes da implementação em cenários reais.

Um dos principais benefícios de utilizar o ROS como plataforma central do projeto é a facilidade de transição entre os ambientes simulado e real. Devido à estrutura modular e ao uso de tópicos padronizados, a maioria dos scripts desenvolvidos para configuração dos sensores, controle do robô e processamento dos dados pôde ser reaproveitada sem modificações significativas. Essa portabilidade se deve ao fato de que, tanto na simulação quanto na aplicação real, os sensores e atuadores interagem com o sistema através de interfaces ROS bem definidas. Dessa forma, o mesmo código que coleta e processa dados de sensores simulados no *CoppeliaSim* pôde ser utilizado para lidar com dados de sensores reais, bastando apenas ajustar os parâmetros específicos de cada dispositivo ou ambiente. Esse reaproveitamento agiliza o desenvolvimento, reduz a ocorrência de erros e reforça a confiabilidade do sistema, pois permite testar previamente todos os módulos em um ambiente seguro e controlado.

3.2 Aquisição de Dados

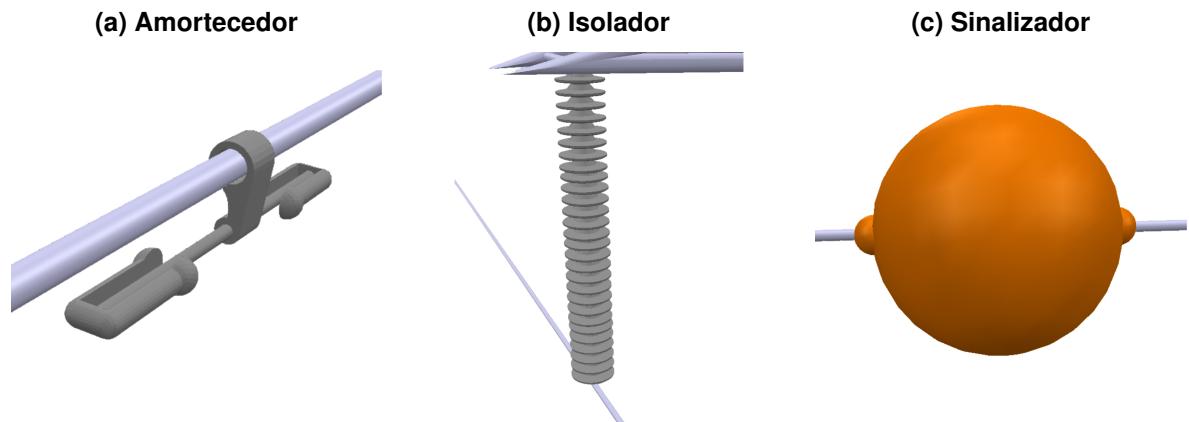
A aquisição de dados constitui uma etapa fundamental deste trabalho, pois fornece as informações brutas que serão utilizadas no treinamento e validação dos modelos de classificação. Esta seção descreve a estratégia adotada para a coleta de dados, os tipos de sensores utilizados, a forma como os dados foram organizados e armazenados, além das ferramentas e scripts empregados no processo. Também são discutidas práticas de pré-processamento inicial, com o objetivo de garantir a consistência e a qualidade dos dados coletados. Por fim, são abordadas considerações relacionadas à reproduzibilidade do processo de coleta e à avaliação crítica da base de dados gerada. A seção está dividida em sete partes principais: objetos de interesse, sensores utilizados, estratégia geral de coleta, estrutura de armazenamento, ferramentas utilizadas, pré-processamento e considerações éticas e técnicas sobre a coleta.

3.2.1 Classes de Objetos a Serem Detectadas

No contexto deste projeto, as classes de objetos a serem detectadas e classificadas são componentes e situações comumente encontradas em inspeções de linhas de transmissão. As Figuras 1 e 2 ilustram os objetos de interesse, tanto em suas representações simuladas quanto reais, respectivamente. As classes incluem:

- **Isoladores Poliméricos:** Dispositivos utilizados para isolar condutores em linhas de alta tensão, essenciais para garantir a segurança e a eficiência das linhas elétricas. Um exemplo de isolador simulado é visto na Figura 1b e um real na Figura 2b.
- **Sinalizadores de Linhas de Alta Tensão:** Marcadores visuais colocados em linhas de alta tensão para aumentar a visibilidade e evitar acidentes (Figuras 1c e 2c).
- **Amortecedores:** Dispositivos que ajudam a reduzir vibrações e choques em sistemas de transmissão e suportes de linhas de alta tensão (Figuras 1a e 2a).
- **Ausência de Obstáculos** (classe "Nada"): Situações em que não há nenhum dos objetos de interesse no campo de visão direto do sensor ou à frente do robô, o que é uma condição importante a ser monitorada para a navegação segura.

Figura 1 – Objetos de interesse no ambiente de simulação: (a) Amortecedor, (b) Isolador e (c) Sinalizador.

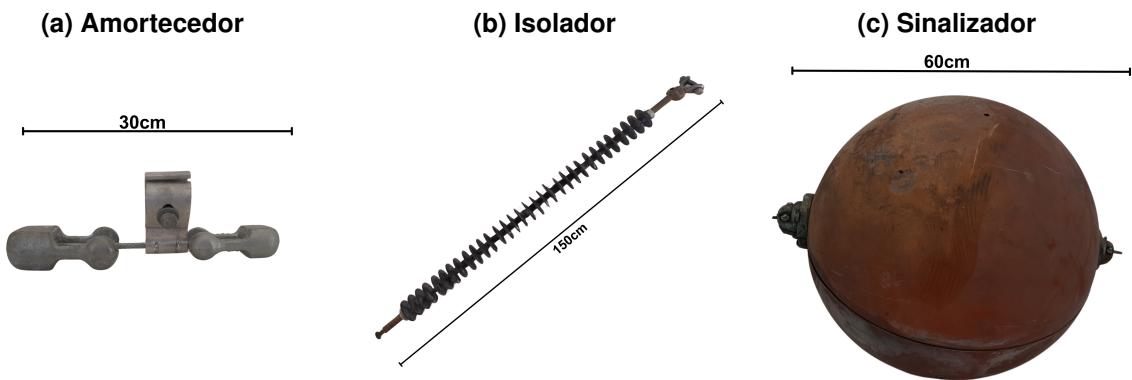


Fonte: Autoria própria (2025).

3.2.2 Sensores utilizados

Neste projeto, foram utilizados dois tipos de sensores: o LiDAR e a câmera de profundidade. O LiDAR funciona emitindo feixes de luz laser que são refletidos pelos objetos ao seu redor. A partir do tempo que o feixe leva para retornar ao sensor, é possível calcular a distância até o objeto com alta precisão. Ele gera um valor de distância para cada ângulo de varredura,

Figura 2 – Objetos de interesse no ambiente real: (a) Amortecedor, (b) Isolador e (c) Sinalizador.



Fonte: Autoria própria (2025).

cobrindo um campo de visão de 360º, o que permite mapear a estrutura tridimensional do ambiente de forma eficiente. Já a câmera de profundidade utiliza luz infravermelha para gerar um mapa denso de profundidade, capturando a distância de cada ponto em sua linha de visão. Diferentemente do LiDAR, que mede distâncias pontuais a partir de um feixe, a câmera de profundidade fornece uma imagem completa com a informação de distância de cada pixel.

3.2.3 Planejamento da Coleta de Dados

A coleta de dados neste trabalho foi planejada com o objetivo de gerar uma base diversificada e representativa de amostras que permitisse o treinamento eficaz de modelos de aprendizado de máquina para a classificação de objetos em linhas de transmissão. Para isso, buscou-se capturar diferentes características espaciais dos objetos de interesse, considerando variações de distância. Foram utilizados dois tipos de sensores de profundidade, de modo a enriquecer a base com diferentes perspectivas e formatos de dados. A coleta foi realizada em dois cenários distintos, um simulado e outro real, seguindo um protocolo padronizado de captura de múltiplas amostras por objeto. Em ambos os cenários, foi coletado um total de 150 instâncias de dados válidos para cada tipo de objeto, totalizando um número significativo de amostras que ajudam na robustez do modelo de aprendizado. Cada instância corresponde a uma varredura de 360º realizada pelo LiDAR, bem como uma imagem capturada pela câmera de profundidade, com cada amostra variando em relação à distância do robô aos objetos, e consequentemente, a distância entre os sensores e os mesmos. Os dados foram organizados e rotulados de forma a permitir sua posterior divisão em conjuntos de treinamento, validação e teste.

A base de dados coletada neste trabalho é crucial para o treinamento dos modelos de aprendizado de máquina, pois sua riqueza em informações determina diretamente a eficácia e precisão da classificação dos objetos. Quanto mais diversificada e representativa for a base de dados, mais robusto e preciso será o modelo treinado, ainda que o treinamento se torne mais intenso e demorado devido ao volume de dados. O objetivo principal da coleta é capturar as características espaciais dos objetos de interesse. Espera-se que os sensores de profundidade

possam extrair informações detalhadas sobre a geometria dos objetos, suas dimensões, e suas posições relativas dentro do ambiente de inspeção. Esses dados são essenciais para permitir uma classificação precisa e eficiente dos diferentes tipos de objetos encontrados durante a inspeção das linhas de transmissão.

3.2.4 Organização e Formato dos Dados Coletados

Para possibilitar o uso eficiente dos dados capturados no treinamento e validação dos modelos de aprendizado de máquina, adotou-se uma estrutura de armazenamento clara e padronizada. Os dados do sensor LiDAR foram armazenados em arquivos *CSV* (Comma-Separated Values), e os dados da câmera de profundidade foram salvos em imagens no formato *PNG* (Portable Network Graphics), ambos amplamente compatíveis com bibliotecas de processamento. Além disso, cada amostra foi acompanhada por metadados como rótulo da classe, origem (simulado ou real) e timestamp, assegurando organização e rastreabilidade.

No caso do LiDAR, cada arquivo CSV contém 150 leituras, correspondendo a 150 amostras capturadas em diferentes distâncias entre o robô e o objeto. Cada linha representa uma leitura completa, e cada coluna corresponde a um ângulo fixo dentro do campo de visão de 360°, com os valores expressando distâncias em metros (até 12 m). A estrutura tabular simplifica o processamento e permite fácil inspeção e depuração, além de possibilitar sincronização com os dados da câmera.

As imagens da câmera de profundidade foram salvas em escala de cinza, com codificação ajustada conforme a origem dos dados. Para as imagens obtidas na simulação, os valores de cada pixel foram representados por inteiros sem sinal de 8 bits, totalizando uma escala de 0 a 255. Cada valor nessa escala foi linearmente associado a uma distância entre 0 e 10 metros, permitindo a reconstrução aproximada da profundidade observada em cada ponto da imagem. Já para a câmera real, os valores de pixel foram salvos utilizando inteiros sem sinal de 16 bits, com valores de 0 a 65.535. Nesse caso, a correspondência entre o valor do pixel e a distância é direta: cada unidade representa um milímetro de distância entre o objeto e o sensor. Em ambos os casos, a resolução da câmera foi mantida em 1280 × 720 pixels, garantindo consistência na estrutura dos dados para posterior processamento.

Para assegurar a correta associação entre sensores e objetos, cada leitura foi feita com apenas um objeto presente no campo de visão, o que elimina ambiguidades na rotulagem. Os dados foram organizados em pastas hierárquicas: o diretório principal indica a origem (*real* ou *simulado*) e contém subpastas nomeadas conforme o tipo de objeto (amortecedor, isolador, sinalizador ou nenhum), o que facilita a navegação e o uso por scripts automatizados.

3.2.5 Ferramentas e Scripts de Coleta

A aquisição dos dados foi conduzida por meio de scripts desenvolvidos em Python (Python Software Foundation, 2025), integrados ao ROS para gerenciar a comunicação

entre os componentes do sistema. Esses scripts eram responsáveis por subscrever os tópicos dos sensores LiDAR e da câmera de profundidade, controlar o fluxo de aquisição e salvar os dados de forma sincronizada.

Para manipulação e organização dos dados, foram empregadas bibliotecas amplamente utilizadas na comunidade científica. O *NumPy* (Harris *et al.*, 2020) foi utilizado para processar vetores numéricos com eficiência, enquanto o *Pandas* (Mckinney *et al.*, 2010) facilitou a estruturação e exportação dos dados em arquivos *CSV*. Para as imagens de profundidade, a biblioteca *OpenCV* (Bradski, 2000) foi utilizada na conversão e salvamento das imagens no formato *PNG* em escala de cinza, conforme os padrões de codificação descritos anteriormente.

A coleta foi automatizada por meio de um loop de captura controlado, no qual cada iteração realizava a aquisição simultânea dos dados dos dois sensores. Entre as capturas, o robô era deslocado progressivamente em direção ao objeto alvo, com a distância de movimentação variando conforme o tipo de objeto analisado. Isso permitiu a obtenção de amostras com diferentes distâncias, aumentando a diversidade e representatividade da base de dados.

Para garantir a integridade e rastreabilidade dos dados, implementou-se também um sistema de registro de logs. Esse sistema armazenava informações como timestamp das amostras, falhas na leitura e inconsistências detectadas. Leituras fora do padrão esperado, como arquivos corrompidos, valores nulos ou discrepantes, foram automaticamente sinalizadas e descartadas.

Essa abordagem automatizada e monitorada proporcionou uma coleta de dados robusta, consistente e reproduzível, minimizando a intervenção manual e otimizando o tempo necessário para aquisição.

3.2.6 Pré-processamento Inicial

Devido ao funcionamento do sensor LiDAR, que realiza leituras em 360°, foi necessário filtrar os dados para considerar apenas os ângulos relevantes ao projeto, uma vez que essa configuração não pode ser alterada diretamente via hardware ou firmware. Assim, durante a captura, são selecionados apenas os valores correspondentes à faixa angular de interesse.

Já as imagens capturadas pela câmera de profundidade não passaram por pré-processamento antes do salvamento, pois os dados foram armazenados diretamente no formato esperado para as etapas subsequentes.

3.2.7 Considerações Éticas e Reprodutibilidade

O processo de coleta de dados foi planejado para garantir que outros pesquisadores possam replicar os experimentos e utilizar a base de dados gerada. Para isso, toda a metodologia, configurações dos sensores, scripts de aquisição e construção dos cenários simulados foram documentados detalhadamente. Além disso, os dados coletados e os datasets construídos estão disponíveis para acesso público no seguinte endereço: Datasets.

O objetivo é que o projeto seja totalmente replicável, ou que seus dados possam ser reutilizados em trabalhos futuros, mediante solicitação prévia para fins de colaboração ou desenvolvimento.

Quanto à qualidade dos dados, foram aplicados filtros para eliminar valores aberrantes ou muito discrepantes durante as aquisições. As rotulações foram cuidadosamente revisadas para evitar erros de classificação. A quantidade de amostras para cada tipo de objeto é considerada suficiente para o escopo deste trabalho, cobrindo variações realistas de distância entre os sensores e os objetos.

Entretanto, é importante destacar que, apesar dos cuidados adotados, algumas fontes de viés podem estar presentes nos dados. Por exemplo, a coleta realizada em ambientes controlados, tanto simulados quanto reais, pode não contemplar todas as variações e ruídos presentes em situações operacionais reais. Além disso, a escolha dos objetos e das distâncias pode influenciar a representatividade dos dados para outros cenários. Reconhecer essas limitações é fundamental para a correta interpretação dos resultados e para orientar futuros trabalhos de validação e expansão da base.

3.3 Construção das Cenas Simuladas

Nesta seção são apresentados os procedimentos adotados para a criação das cenas simuladas no ambiente virtual, essenciais para a coleta controlada dos dados utilizados no treinamento e validação dos modelos de classificação. A simulação permitiu a definição precisa da configuração espacial dos sensores e dos objetos de interesse, a experimentação ágil de diferentes cenários e parâmetros, além de garantir segurança e redução de custos ao evitar o desgaste e riscos associados à operação em ambientes reais. Os detalhes da modelagem, posicionamento dos sensores, controle do robô simulado e integração com o ROS serão detalhados nas subseções seguintes, assim como o protocolo de coleta e as limitações inerentes ao uso da simulação.

3.3.1 Ambiente Virtual Criado no CoppeliaSim

A estrutura geral da cena simulada consistia em duas torres de transmissão posicionadas a 100 metros de distância uma da outra, conectadas por um cabo cilíndrico e retilíneo, sem curvaturas, a fim de simplificar a análise e evitar instabilidades na simulação. Os objetos de interesse foram pendurados ao longo do cabo, dispostos de forma que pudessem ser capturados individualmente pelos sensores. O robô foi posicionado sobre o cabo com liberdade de movimento linear, controlado pelo usuário por meio de comandos com velocidade ajustável. Os sensores foram acoplados a um suporte fixado no robô, permitindo sua movimentação ao longo da linha.

A modelagem dos componentes utilizados na simulação foi realizada por meio de uma combinação entre o software SolidWorks e os recursos nativos do CoppeliaSim. Os modelos dos

robôs foram inicialmente criados no SolidWorks, o que possibilitou uma construção geométrica precisa. Posteriormente, foram exportados para o CoppeliaSim, onde as partes dinâmicas dos robôs foram substituídas por formas primitivas, como cubos e cilindros, a fim de garantir maior leveza e estabilidade na simulação sem comprometer a fidelidade das interações físicas.

Os objetos de interesse para classificação — amortecedores, isoladores e sinalizadores — bem como as torres, também foram modelados no SolidWorks e importados integralmente para o simulador. Diferente do robô, esses elementos mantiveram suas formas completas, pois era essencial preservar suas características físicas e visuais para uma correta detecção e classificação pelos sensores. O cabo foi representado por um cilindro simples, evitando o uso de simulação de curvatura realista, que poderia sobrecarregar o ambiente.

Para assegurar a portabilidade do sensor multimodal (LiDAR e câmera de profundidade) entre diferentes plataformas robóticas, foi desenvolvido um suporte específico no SolidWorks, projetado para manter os sensores corretamente alinhados e fixos. Esse suporte foi integrado ao robô no ambiente do CoppeliaSim, e pode ser facilmente adaptado a diferentes topologias robóticas, permitindo testes variados com o mesmo conjunto de sensores.

A complementaridade dos sensores utilizados está diretamente relacionada à sua posição no robô: a câmera de profundidade foi posicionada na parte superior, oferecendo uma visão panorâmica da porção superior do cabo e dos objetos suspensos, enquanto o LiDAR foi instalado abaixo do robô, capturando dados da região inferior. A combinação dessas perspectivas amplia a qualidade e a riqueza dos dados obtidos, fornecendo uma visão tridimensional mais completa dos componentes inspecionados nas linhas de transmissão.

3.3.2 Robô Simulado

Neste projeto, foram utilizados três protótipos de robôs simulados com o objetivo principal de testar a portabilidade e a adaptabilidade do suporte de sensores entre diferentes topologias robóticas. No entanto, apenas um desses robôs foi utilizado para a coleta dos dados de treinamento dos modelos, enquanto os outros dois serviram para validar o desempenho dos modelos em estruturas distintas.

O robô utilizado na fase de treinamento é o mais simples dos três. Sua estrutura é composta por um suporte de sensores posicionado de forma perpendicular ao cabo e alinhado ao corpo do robô. Ele possui um mecanismo de travamento que permite sua fixação ao cabo e é capaz de realizar deslocamentos lineares com velocidade ajustável, controlada via ROS. A simplicidade desse modelo favorece maior estabilidade durante a coleta de dados, reduzindo variáveis que poderiam interferir nos resultados.

Os dois robôs adicionais apresentam maior complexidade estrutural e foram projetados para superar obstáculos presentes nas linhas de transmissão. Nesses modelos, o suporte dos sensores foi instalado com uma leve inclinação em relação ao corpo principal do robô, simulando diferentes ângulos de visualização. Ambos mantêm o mesmo sistema de controle via ROS, com

velocidades lineares também ajustáveis por código, permitindo comparações consistentes com o modelo mais simples.

A construção dos sistemas robóticos foi inteiramente baseada em juntas rotacionais simples disponibilizadas pelo CoppeliaSim. Essas juntas simulam movimentos giratórios básicos e, quando combinadas, possibilitam a reprodução de movimentos mais complexos. Todas as juntas foram integradas ao ROS, o que facilitou a implementação e o controle. Essa abordagem modular e padronizada simplifica o desenvolvimento e a replicação do sistema em diferentes configurações robóticas.

3.3.3 Integração com ROS

A integração entre o ambiente de simulação CoppeliaSim e o sistema ROS foi realizada por meio de scripts escritos em Lua diretamente no simulador. Cada objeto da cena recebe um identificador único no script, o que permite controlar suas propriedades diretamente. No caso de juntas, por exemplo, é possível configurar parâmetros como torque e velocidade.

Utilizando a biblioteca integrada do ROS fornecida pelo CoppeliaSim, foram implementados publicadores e assinantes diretamente nos scripts em Lua. Essa comunicação permite que o simulador interaja em tempo real com o ROS rodando na máquina. As juntas do robô foram controladas por assinantes que recebiam comandos externos, gerados por um código fora do ambiente CoppeliaSim, sob controle do usuário. Por outro lado, os sensores, como a câmera de profundidade e o LiDAR, funcionaram como publicadores, transmitindo dados para seus respectivos tópicos definidos no script de simulação. Esses dados eram então capturados por outro código externo responsável por processá-los e armazená-los para uso posterior.

Para garantir a confiabilidade da integração e dos dados utilizados nos experimentos, foram realizados testes de validação. Objetos foram posicionados a distâncias conhecidas dos sensores, permitindo verificar a acurácia dos dados de profundidade publicados. Além disso, foi verificada a sincronização temporal e espacial entre os sensores antes do início da coleta definitiva dos dados, assegurando que as leituras fossem consistentes e alinhadas para uso no treinamento dos modelos de classificação.

3.3.4 Protocolo de Coleta na Simulação

Para realizar a coleta de dados nas cenas simuladas, os objetos foram posicionados individualmente à frente dos sensores do robô. O controle do robô foi feito manualmente pelo usuário, que o aproximava dos objetos até que o sensor LiDAR detectasse sua presença. A partir desse momento, era iniciada a gravação dos dados, enquanto o robô se deslocava em direção ao objeto com velocidade constante.

A velocidade de deslocamento foi configurada de forma distinta para cada objeto, levando em consideração suas dimensões. Objetos menores, por exemplo, exigiram velocidades

menores para garantir uma quantidade suficiente de amostras, de modo a equilibrar o volume de dados coletado em relação aos objetos maiores.

Após a coleta, os dados passaram por um processo de inspeção para a identificação e remoção de valores discrepantes ou incoerentes. Caso necessário, o processo de aquisição era repetido, a fim de garantir que os dados representassem com fidelidade o objeto analisado. Os dados finais foram então organizados e armazenados de maneira estruturada, incluindo os rótulos correspondentes a cada classe de objeto, para posterior uso no treinamento e avaliação dos modelos de aprendizado de máquina.

3.3.5 Limitações e Considerações sobre a Simulação

A simulação apresenta algumas limitações em relação ao ambiente real, principalmente no que diz respeito à fidelidade das leituras dos sensores. Nos ambientes simulados não há presença de ruídos, reflexos ou deformações nos dados capturados, o que favorece a obtenção de informações limpas e consistentes. No entanto, essa ausência de imperfeições pode comprometer a generalização dos modelos treinados exclusivamente com dados simulados, uma vez que sensores reais estão sujeitos a reflexos, interferências e variações físicas nas medições.

Como ambos os sensores utilizados operam por meio da emissão de feixes de luz (infra-vermelho), o fator que mais impacta a qualidade das medições no mundo real são as reflexões em superfícies irregulares ou brilhantes. Essa característica não é reproduzida na simulação, pois o CoppeliaSim não modela o comportamento real dos feixes de luz nem as propriedades ópticas dos materiais simulados.

Apesar dessas limitações, a simulação desempenha um papel fundamental como etapa preliminar do desenvolvimento. Ela permite gerar conjuntos de dados organizados, com classes bem definidas e isentas de ruído, que são extremamente úteis para a validação inicial dos modelos de aprendizado de máquina. Além disso, os resultados obtidos com dados simulados servem como base de comparação para os modelos treinados com dados reais, oferecendo uma referência inicial sobre o desempenho esperado e facilitando a análise das discrepâncias entre os dois contextos.

3.4 Construção das Cenas no Laboratório

Após a etapa de simulação, torna-se essencial validar o desempenho do sistema em condições reais, onde as limitações dos sensores e os desafios do ambiente físico podem impactar diretamente a eficácia dos modelos desenvolvidos. Esta seção descreve o processo de construção das cenas no laboratório, utilizado como ambiente de testes controlado para replicar, em escala reduzida, os elementos principais de uma linha de transmissão de energia. O objetivo é analisar a robustez do sistema diante de variações naturais e ruídos típicos dos sensores reais, bem como avaliar a viabilidade prática da solução proposta. Serão abordadas desde a infraestrutura física do laboratório, passando pela montagem do protótipo e do cabo suspenso, até

os métodos de coleta e gerenciamento dos dados em ambiente real. A integração com o ROS, a comunicação embarcada e a adaptação dos scripts desenvolvidos na simulação para a coleta física também são discutidas, permitindo uma comparação direta entre os dados simulados e reais.

3.4.1 Objetivos da Validação em Ambiente Real

A validação em ambiente real tem como principal finalidade confirmar a viabilidade prática do sistema desenvolvido, observando o comportamento dos sensores físicos e do robô em condições controladas, porém reais. Por meio desses testes, é possível identificar limitações operacionais dos sensores, como perdas de dados por reflexões, ruídos de leitura e falhas na aquisição, que não estão presentes na simulação. Além disso, essa etapa permite avaliar a robustez dos modelos de classificação quando expostos a variações naturais no ambiente, garantindo que o desempenho observado virtualmente possa ser replicado, ao menos em parte, no cenário físico. A coleta em laboratório representa, portanto, um elo essencial entre o ambiente idealizado da simulação e as futuras aplicações operacionais do sistema em campo.

3.4.2 Infraestrutura do Laboratório

Os testes em ambiente real foram realizados em um laboratório com infraestrutura necessária para a montagem de uma linha de transmissão em escala reduzida, bem como para a construção e operação dos protótipos robóticos utilizados no projeto. O espaço físico disponível permitia a instalação de um cabo suspenso com cerca de 5 metros de comprimento, fixado a aproximadamente 2 metros do solo, proporcionando um ambiente controlado e representativo para a realização dos experimentos.

A sustentação do cabo foi viabilizada por meio de suportes metálicos do tipo cotovelo, fixados diretamente às paredes laterais do laboratório. O cabo foi preso aos suportes utilizando abraçadeiras de aço, garantindo rigidez e leve afundamento central, simulando a curvatura natural de cabos em linhas reais. Essa configuração permitiu a instalação segura dos objetos de interesse e possibilitou o deslocamento contínuo do robô ao longo do trecho durante as coletas.

3.4.3 Coleta com Robô em Cabo Suspenso

O robô utilizado nas coletas em ambiente real é uma réplica funcional do modelo empregado nas simulações virtuais. Trata-se de um protótipo móvel desenvolvido para se deslocar linearmente sobre um cabo suspenso, com capacidade de ajuste de velocidade e mecanismos de travamento e destravamento, garantindo segurança e estabilidade durante a movimentação e a aquisição dos dados.

O sistema de locomoção é composto por dois motores de passo: um responsável pelo deslocamento linear ao longo do cabo e outro dedicado ao acionamento do mecanismo de travamento. Ambos os motores são controlados por um microcontrolador Arduino, que atua como interface entre o hardware e o ROS, permitindo a reutilização do mesmo código e lógica de controle aplicados na simulação. A estrutura completa do robô e seu funcionamento detalhado podem ser consultados em Domingues *et al.* (2024).

O controle e a alimentação do sistema são realizados por uma Raspberry Pi 5, que se comunica com o Arduino e com os sensores (LiDAR e câmera de profundidade) por meio de interfaces seriais. Toda a alimentação elétrica é provida por baterias embarcadas, tornando o sistema completamente autônomo e dispensando conexões externas durante os testes.

3.4.3.1 Estrutura do cabo suspenso

A estrutura do experimento consiste em um cabo de alumínio com alma de aço (ACSR — Aluminium Conductor Steel Reinforced), similar aos utilizados em linhas de transmissão reais. O cabo, com cerca de 5 metros de comprimento, foi suspenso a uma altura aproximada de 2 metros do solo, fixado nas extremidades por meio de suportes metálicos do tipo cotovelo ancorados nas paredes do laboratório. A fixação foi feita com abraçadeiras de aço, garantindo estabilidade e uma leve curvatura natural do cabo.

Os objetos de interesse — como isoladores, amortecedores e sinalizadores — foram fixados ao cabo individualmente durante os testes. Os componentes utilizados são peças reais, idênticas às aplicadas em linhas de transmissão de alta tensão, e foram montados utilizando os mesmos métodos de fixação empregados em campo, assegurando uma representação fidedigna do ambiente operacional.

3.4.4 Integração com ROS

Para a integração dos sensores e atuadores no ambiente real, o ROS é executado diretamente na Raspberry Pi 5 embarcada no robô. A Raspberry atuou como nó mestre, sendo responsável pela orquestração e processamento de todas as informações do sistema. No entanto, ela não possui suporte nativo aos sistemas operacionais Ubuntu 20.04 ou Debian 10, os quais são requisitos para o ROS Noetic — versão escolhida para este projeto. Para contornar essa limitação, optou-se pelo uso do Docker, uma plataforma de virtualização leve baseada em contêineres que permite a execução isolada de aplicações em qualquer sistema operacional compatível (Merkel, 2014).

O contêiner Docker foi configurado com permissões equivalentes ao usuário root do sistema, garantindo acesso irrestrito a portas seriais e arquivos do dispositivo. Dentro desse contêiner, o ambiente ROS foi completamente configurado, incluindo os nós responsáveis por publicar os dados dos sensores e comandar os motores.

O controle dos motores foi realizado por meio do pacote *rosserial*, que provê uma interface de comunicação entre dispositivos embarcados, como o Arduino, e o ROS, permitindo a publicação e subscrição de tópicos por meio de uma conexão serial (Open Source Robotics Foundation, 2025).

Para a câmera de profundidade fui utilizada a Intel RealSense D415 com o pacote *realsense-ros*, fornecido pela própria Intel. Esse pacote oferece *launch files* e configurações prontas para a integração da câmera ao ROS, facilitando a captura e publicação dos dados de profundidade e cor (Intel Corporation, 2025).

O sensor LiDAR empregado foi o RPLIDAR A1, cuja integração foi realizada por meio do pacote *rplidar_ros*, disponibilizado pela Slamtec. O pacote contém arquivos de configuração específicos para cada modelo de sensor, incluindo parâmetros de calibração e tópicos de publicação compatíveis com o ROS (Slamtec, 2025).

Com esses pacotes devidamente configurados e em execução no contêiner Docker, os dados brutos dos sensores passaram a ser publicados em tópicos ROS, estando assim disponíveis para subscrição e processamento por outros módulos do sistema.

A coleta dos dados foi realizada utilizando os mesmos scripts empregados na fase de simulação, garantindo consistência e sincronização entre os sensores. Os dados adquiridos foram armazenados localmente na Raspberry Pi durante as sessões de coleta e, posteriormente, transferidos para uma estação de trabalho dedicada, onde foram utilizados para o treinamento e validação dos modelos de classificação.

3.4.5 Protocolo de Coleta no Laboratório

A coleta de dados no ambiente real foi conduzida de forma similar à realizada na simulação. O robô foi posicionado de forma manual próximo a cada objeto até que se confirmasse a correta detecção e leitura pelos sensores. Em seguida, o robô foi configurado para se deslocar com velocidade constante ao longo do cabo, realizando a captura dos dados sensoriais.

Cada objeto foi analisado individualmente, sendo necessário ajustar a velocidade do robô para cada caso, de modo a garantir a qualidade da leitura — assim como foi feito na fase de simulação. A velocidade foi escolhida de forma que os sensores pudessem capturar uma quantidade suficiente de amostras com o objeto dentro do campo de visão.

Após a coleta, os dados foram inspecionados manualmente para remoção de valores discrepantes ou amostras com ruídos excessivos. Esse processo garantiu que o conjunto final fosse conciso, representativo e estivesse devidamente rotulado, assegurando a integridade e a confiabilidade das informações utilizadas no treinamento e validação dos modelos de aprendizado de máquina.

3.4.6 Comparação com a Simulação

Os dados coletados com os sensores reais apresentaram níveis de ruído, como era esperado, mas de forma geral mantiveram boa qualidade quando comparados aos dados da simulação, especialmente na análise de objetos com superfícies não reflexivas. Nesses casos, as leituras do LiDAR e da câmera de profundidade se mostraram consistentes e compatíveis com os dados simulados, validando a fidelidade do ambiente de simulação.

No entanto, ao lidar com objetos reflexivos, como o próprio cabo utilizado na estrutura, foram observadas distorções nas leituras. O LiDAR apresentou, em diversos momentos, medições no limite superior de alcance (cerca de 12 metros), especialmente nas primeiras e últimas amostras do objeto, quando os feixes eram emitidos em ângulos muito abertos em relação ao cabo. Tais valores são fisicamente impossíveis dentro do ambiente de teste — cujo teto estava a apenas 2 metros de altura — e indicam perda de dados devido à reflexão inadequada do feixe do sensor.

Já a câmera de profundidade demonstrou dificuldades ainda maiores nesse cenário. Durante toda a coleta, não foi possível obter leituras válidas de profundidade referentes ao cabo, mesmo em posições mais próximas. O sensor simplesmente atribuía o valor zero para esses pontos, indicando que a luz infravermelha não estava sendo refletida de forma adequada para que a profundidade pudesse ser estimada.

3.5 Registro e Gestão dos Dados

A organização dos arquivos gerados seguiu um padrão simples e funcional, dado que os dados foram coletados individualmente para cada objeto e devidamente rotulados no momento da aquisição. Isso evitou a necessidade de processos posteriores de separação ou balanceamento dos dados.

A nomenclatura adotada para os arquivos refletia diretamente o conteúdo dos dados, indicando o nome do objeto, o sensor utilizado (LiDAR ou câmera de profundidade) e a origem dos dados (simulação ou ambiente real). Essa padronização facilitou o processo de criação e manipulação dos *datasets*.

Os dados utilizados para o treinamento dos modelos foram organizados em 10 *datasets* distintos. As divisões foram feitas considerando:

- Origem dos dados: simulado ou real;
- Tipo de sensor: apenas LiDAR, apenas câmera de profundidade ou ambos combinados;
- Forma de representação: dados brutos ou extração de *features*.

Cada *dataset* era composto por arquivos no formato `.csv`, contendo 150 amostras para cada uma das quatro classes de objetos (amortecedor, isolador, sinalizador, nada), totalizando 600 amostras por *dataset*.

Além dos arquivos tabulares, foram criados dois *datasets* adicionais com imagens de profundidade:

- Um composto por imagens simuladas, obtidas diretamente do ambiente virtual;
- Outro composto por imagens reais, submetidas a técnicas de pré-processamento para remoção de ruído e aprimoramento da qualidade.

Esses conjuntos de imagens foram utilizados para o treinamento de uma rede neural convolucional, que serviu como *benchmark* para avaliar a utilidade das informações visuais nas tarefas de classificação.

3.6 Treinamento e Avaliação dos Modelos

Esta seção descreve o processo de modelagem e avaliação dos algoritmos de aprendizado de máquina utilizados para a classificação de objetos em linhas de transmissão, com base em informações de profundidade coletadas por sensores. A motivação para o uso de técnicas de inteligência artificial neste contexto está na capacidade dessas abordagens de extrair padrões complexos dos dados, superando limitações de sistemas baseados apenas em regras ou análise manual. Ao treinar modelos supervisionados com os dados previamente rotulados, busca-se criar um sistema autônomo de classificação que seja robusto a variações dentro de um cenário real ou simulado.

3.6.1 Objetivo da Etapa de Treinamento

A etapa de treinamento tem como propósito utilizar os dados coletados — tanto simulados quanto reais — para desenvolver modelos capazes de classificar automaticamente os objetos encontrados ao longo de uma linha de transmissão, como isoladores, sinalizadores, amortecedores ou a ausência de objetos (classe “nada”). A modelagem visa permitir que, a partir das informações de profundidade fornecidas pelos sensores, seja possível inferir corretamente a classe do objeto observado.

3.6.2 Arquiteturas e Algoritmos Testados

Os modelos testados neste trabalho foram k-Vizinhos Mais Próximos (k-NN), Árvore de Decisão, Naive Bayes, Floresta Aleatória, Perceptron Multicamadas (MLP) e uma Rede Neural Convolucional (SqueezeNet), todos amplamente utilizados em tarefas de classificação. A seguir, descrevemos o funcionamento de cada modelo e seus principais hiperparâmetros, que impac-

tam diretamente o desempenho e a capacidade de generalização. Para uma explicação mais aprofundada, consulte (Bishop; Nasrabadi, 2006).

3.6.2.1 k-Vizinhos mais próximos

O kNN é um modelo de aprendizado supervisionado que classifica um objeto com base na proximidade com exemplos já conhecidos no espaço de características. O princípio básico é encontrar os k exemplos mais próximos (vizinhos) e decidir a classe do novo objeto segundo as classes desses vizinhos.

Os hiperparâmetros importantes do kNN incluem o número de vizinhos considerados, que define quantos exemplos próximos serão usados para a decisão, a métrica de distância, que determina como a proximidade entre pontos é calculada e pode influenciar a sensibilidade do modelo a diferentes características, e o esquema de pesos, que pode dar maior importância a vizinhos mais próximos.

3.6.2.2 Árvore de Decisões

A árvore de decisões utiliza uma estrutura hierárquica de perguntas sobre atributos dos dados para chegar a uma classificação final. Cada nó interno representa uma condição sobre um atributo e as folhas indicam a classe predita.

Os hiperparâmetros que controlam a árvore incluem a profundidade máxima, que limita o tamanho da árvore para evitar que ela se ajuste demais aos dados de treinamento (*overfitting*), o número mínimo de amostras para dividir um nó, e o número mínimo de amostras para formar uma folha. Esses parâmetros ajudam a balancear a complexidade do modelo.

3.6.2.3 Naive Bayes

O Naive Bayes é um modelo probabilístico que assume independência condicional entre as características dado a classe. Ele calcula a probabilidade de cada classe dado o vetor de características e escolhe a mais provável.

Embora possua poucos hiperparâmetros, um deles é a escolha da distribuição probabilística usada para modelar os dados, como Gaussiana ou multinomial, o que afeta o cálculo das probabilidades condicionais e, consequentemente, o desempenho do modelo.

3.6.2.4 Floresta Aleatória

A Floresta Aleatória é um conjunto de árvores de decisão treinadas em subconjuntos aleatórios dos dados e características, cuja decisão final é feita por votação entre as árvores.

Os principais hiperparâmetros são o número de árvores no conjunto, que influencia a estabilidade e robustez da predição; a profundidade máxima das árvores; o número mínimo de amostras para dividir nós ou formar folhas; e o número máximo de características consideradas em cada divisão, que impacta a diversidade entre as árvores e ajuda a evitar *overfitting*.

3.6.2.5 Perceptron Multicamadas

O MLP é uma rede neural feedforward composta por múltiplas camadas de neurônios totalmente conectados, capaz de modelar relações não lineares complexas entre as características e as classes.

Seus hiperparâmetros incluem a arquitetura da rede, ou seja, o número de camadas e neurônios por camada, que definem a capacidade de representação do modelo; a taxa de aprendizado, que afeta a velocidade e qualidade da convergência durante o treinamento; a função de ativação, que introduz não linearidade; e o número de épocas, que determina quantas vezes o conjunto de dados é usado para atualizar os pesos.

3.6.2.6 Rede Neural Convolucional – SqueezeNet

Além dos modelos clássicos, foi utilizada uma rede neural convolucional para classificação das imagens de profundidade capturadas pela câmera. A rede escolhida foi a SqueezeNet (Iandola *et al.*, 2016), que é conhecida por atingir acurácia comparável à AlexNet com um número muito menor de parâmetros e tamanho reduzido do modelo, facilitando sua implementação em sistemas embarcados.

A SqueezeNet é composta por blocos “fire”, que combinam camadas convolucionais de filtros pequenos (1×1) e maiores (3×3) para extraír características complexas das imagens de forma eficiente. Seus hiperparâmetros envolvem a configuração do número de filtros em cada camada, o tamanho dos filtros convolucionais, além dos parâmetros típicos de treinamento, como taxa de aprendizado, tamanho do lote (batch size) e número de épocas. Esperava-se que essa rede apresentasse desempenho superior em acurácia, ainda que com tempo de processamento maior que os modelos clássicos, sendo usada como referência para validação dos dados obtidos pela câmera de profundidade.

3.6.3 Treinamento dos Modelos

Os modelos foram treinados utilizando ferramentas robustas e amplamente adotadas na comunidade de aprendizado de máquina e aprendizado profundo. As principais bibliotecas utilizadas foram o `scikit-learn` e o `PyTorch`, ambas desenvolvidas em Python.

O `scikit-learn` é uma biblioteca de aprendizado de máquina de alto nível que oferece uma ampla variedade de algoritmos clássicos, incluindo kNN, árvores de decisão, Naive Bayes e Florestas Aleatórias, além de utilitários para pré-processamento, avaliação e seleção de modelos. Sua interface simples e eficiente facilita a implementação rápida de experimentos, sendo ideal para o treinamento e validação dos modelos clássicos usados neste trabalho (Pedregosa *et al.*, 2011).

Já o `PyTorch` é uma biblioteca focada em aprendizado profundo, que proporciona um estilo imperativo de programação com alta performance e flexibilidade na construção de redes neurais, como a rede convolucional SqueezeNet utilizada neste projeto. Com suporte dinâmico

para computação em GPU e funcionalidades avançadas para otimização e manipulação de tensores, o PyTorch permite o desenvolvimento e treinamento eficiente de modelos complexos (Paszke *et al.*, 2019).

Essas ferramentas complementares permitiram o desenvolvimento dos modelos clássicos e da rede neural convolucional, garantindo uma base sólida para o processo de treinamento e avaliação.

3.6.4 Avaliação de Desempenho

A avaliação dos modelos desenvolvidos foi realizada utilizando a técnica de validação cruzada, que consiste em dividir o conjunto de dados em subconjuntos para garantir uma análise mais robusta e menos enviesada do desempenho do modelo. Na validação cruzada, os dados são segmentados em múltiplas partições, onde, em cada iteração, um subconjunto é utilizado para teste enquanto os demais servem para treinamento. Esse processo é repetido até que todos os subconjuntos tenham sido usados como teste, e os resultados são então agregados para obter uma estimativa geral da performance.

A métrica escolhida para quantificar o desempenho dos modelos foi a acurácia, definida como a proporção de classificações corretas em relação ao total de amostras avaliadas. A acurácia é uma medida intuitiva e amplamente utilizada para problemas de classificação, pois indica diretamente a eficácia do modelo em reconhecer corretamente as classes presentes nos dados.

Além da avaliação tradicional, foi realizada uma validação específica para testar a capacidade dos modelos de generalizar entre diferentes configurações robóticas. Para isso, os modelos foram treinados utilizando um dataset obtido a partir de uma topologia robótica e validados em outro conjunto de dados capturado por uma topologia distinta. Essa abordagem visa avaliar a robustez do sistema frente a variações nas topologias de robôs inspetores.

O tempo gasto no processo de validação foi também monitorado, pois é um fator importante para aplicações em sistemas embarcados e em tempo real, onde a eficiência computacional é crítica para o funcionamento prático dos classificadores.

3.6.5 Limitações da Metodologia

A metodologia adotada neste trabalho apresenta algumas limitações importantes, principalmente relacionadas à coleta de dados reais para treinamento e validação dos modelos. Primeiramente, a coleta de dados reais está sujeita a restrições de espaço físico disponível no laboratório. O que impacta diretamente na representatividade e diversidade dos dados coletados.

No ambiente simulado, apesar de fornecer uma boa aproximação do cenário real de uma linha de transmissão, os sensores não sofrem interferências externas, como ruídos e reflexões,

o que resulta em dados mais limpos e idealizados, que podem não refletir completamente as condições reais encontradas em campo.

Por outro lado, no ambiente de laboratório, os dados são capturados de sensores reais e, portanto, incluem reflexões e interferências típicas do mundo real, tornando-os mais concretos. Contudo, devido ao espaço físico limitado, o cenário representa apenas uma aproximação parcial das condições reais de uma linha de transmissão. Especificamente, o fundo de escala dos sensores nesse ambiente varia entre 2 e 5 metros, enquanto em uma estrutura real o fundo de escala pode alcançar até 12 metros, devido à ausência de objetos além da própria linha. Isso implica que as leituras obtidas no laboratório não reproduzem com fidelidade todas as características do ambiente real.

Além disso, por se tratar de um ambiente controlado, existe o risco de *overfitting* dos modelos, isto é, os modelos podem apresentar bom desempenho nos dados utilizados para treinamento, mas ter baixa capacidade de generalização para situações não contempladas, como a presença de objetos inesperados na linha de transmissão ou variações significativas na posição dos objetos.

Essas limitações indicam a necessidade de futuras coletas de dados em ambientes reais e diversificados para melhorar a robustez e aplicabilidade dos modelos desenvolvidos.

4 DESENVOLVIMENTO

Este capítulo apresenta o desenvolvimento do sistema robótico proposto, abrangendo tanto os aspectos de hardware quanto os de software. Inicialmente, são descritas as características físicas e funcionais da plataforma robótica desenvolvida, incluindo alguns detalhes sobre a estrutura mecânica, os sensores embarcados e os sistemas de controle e comunicação utilizados. Também são apresentados os dois outros protótipos empregados durante as fases de simulação, com destaque para suas diferenças estruturais e contribuições ao processo de validação. Em seguida, é detalhada a configuração dos sensores no ambiente ROS, abordando os drivers utilizados, os parâmetros operacionais ajustados e as estratégias adotadas para formatar os dados obtidos em diferentes plataformas. Posteriormente, são explorados os aspectos de software, incluindo o controle do robô via ROS, o processamento e organização dos dados adquiridos e a implementação dos modelos de aprendizado de máquina. Todo o pipeline, desde a aquisição até a inferência em tempo real, foi projetado para ser modular, adaptável e compatível tanto com simulações quanto com a operação embarcada em plataformas reais.

4.1 Hardware

Esta seção descreve os aspectos físicos e estruturais do sistema robótico desenvolvido, incluindo detalhes da construção mecânica do robô utilizado para a coleta de dados reais, os protótipos testados em simulação, e a configuração do suporte dos sensores multimodais. São apresentados também os sensores utilizados e suas principais especificações técnicas.

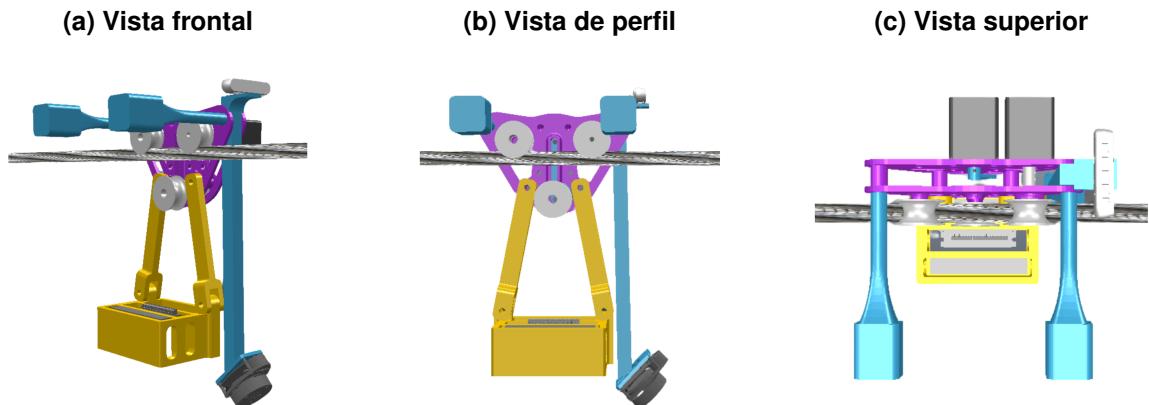
4.1.1 Estrutura Mecânica dos Robôs

O robô utilizado na coleta de dados em ambiente real foi construído com estrutura impressa em 3D, baseada no mesmo modelo utilizado nas simulações desenvolvidas no Coppelia-Sim. A Figura 3 apresenta diferentes perspectivas do modelo virtual empregado nas simulações, enquanto as Fotografias 1 e 2 exibem, respectivamente, uma vista frontal e uma vista de perfil do protótipo físico construído. Seu deslocamento ocorre sobre um cabo rígido, representando uma linha de transmissão, e utiliza dois motores de passo: um responsável pelo deslocamento linear ao longo do cabo e outro para acionar o mecanismo de travamento/destravamento. Este mecanismo, visível em detalhe na Fotografia 2, garante que o robô permaneça fixo ao cabo, impedindo quedas ou movimentos involuntários durante a coleta de dados.

Este modelo apresenta uma estrutura simplificada e não conta com sistemas para a superação de obstáculos. A escolha por este robô para a coleta de dados se deu por sua maior confiabilidade estrutural, sendo o mais estável dentre os disponíveis no laboratório. A proposta para uma versão final inclui a adição de um segundo conjunto de garra e um corpo robusto capaz de sustentar o robô com apenas uma garra acoplada. Em presença de obstáculos, a ideia é que o robô desacople uma das garras, rotacione-a para ultrapassar o obstáculo, recoloque-a no cabo

após o objeto e, em seguida, repita o processo com a outra garra, garantindo a continuidade do deslocamento.

Figura 3 – Vistas do protótipo do robô principal no ambiente de simulação: (a) Vista frontal, (b) Vista de perfil e (c) Vista superior.



Fonte: Autoria própria (2025).

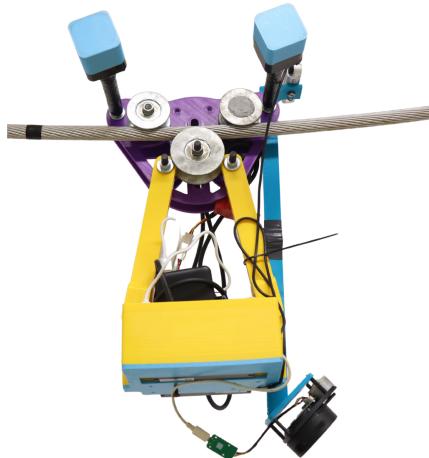
Fotografia 1 – Fotografia do protótipo do robô principal real: Vista frontal.



Fonte: Autoria própria (2025).

Além do protótipo utilizado na coleta real, dois outros robôs com mecanismos funcionais de superação de obstáculos foram projetados e simulados, visando testar a portabilidade do sensor multimodal. A Figura 4 ilustra o primeiro desses robôs secundários, que é composto por um corpo central com duas rodas laterais e uma garra central. Para este robô, o suporte do sensor multimodal foi instalado com uma inclinação adicional de 10 graus em relação à sua posição no robô principal, porém mantendo o mesmo ponto de fixação central. Quando ele detecta um obstáculo, sua garra se prende ao cabo, a roda dianteira se desacopla e desce, enquanto o corpo do robô se estende para reposicionar a roda à frente do obstáculo. Após o

Fotografia 2 – Fotografia do protótipo do robô principal real: Vista de perfil.



Fonte: Autoria própria (2025).

repositionamento, a roda retorna à altura do cabo, se fecha, o robô solta a garra, retraí o corpo e avança, repetindo o processo com a outra roda para completar a travessia.

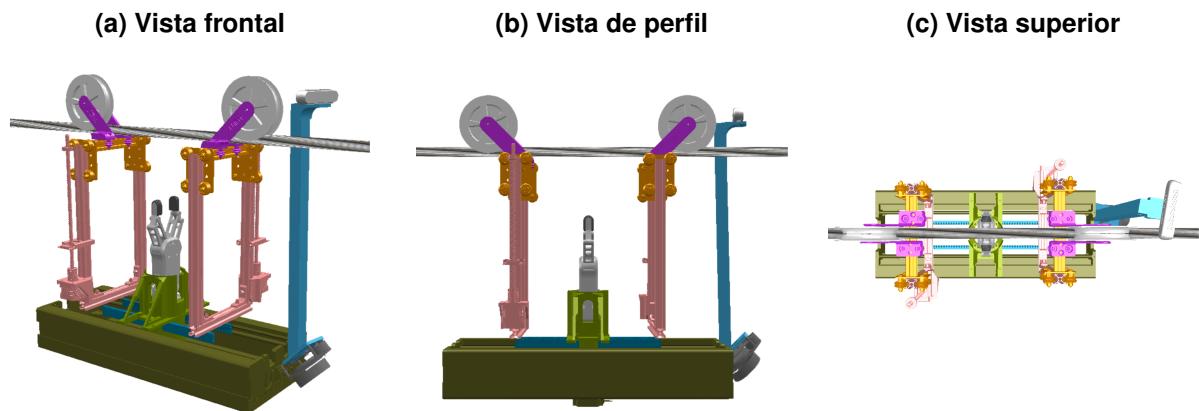
O segundo robô secundário simulado, cujas vistas frontal, de perfil e superior são apresentadas na Figura 5, possui três braços articulados, cada um equipado com rodas e um mecanismo de “cotovelo” acionável. Neste modelo, o suporte do sensor multimodal também sofreu uma alteração de inclinação de 10 graus em relação ao robô principal e, adicionalmente, foi deslocado aproximadamente 15 cm da sua posição original no centro do robô, simulando uma configuração de montagem diferente. Ao detectar um obstáculo, o primeiro braço do robô eleva seu cotovelo e ultrapassa o obstáculo. Em seguida, os outros dois braços repetem o movimento sequencialmente, permitindo que o robô supere a barreira mantendo estabilidade e tração no cabo.

Apesar da complexidade e funcionalidade teórica desses mecanismos de superação de obstáculos, a instabilidade estrutural observada nos protótipos físicos correspondentes inviabilizou sua utilização na coleta de dados em ambiente real, restringindo os testes com estas duas topologias ao ambiente simulado.

4.1.2 Sensor Multimodal

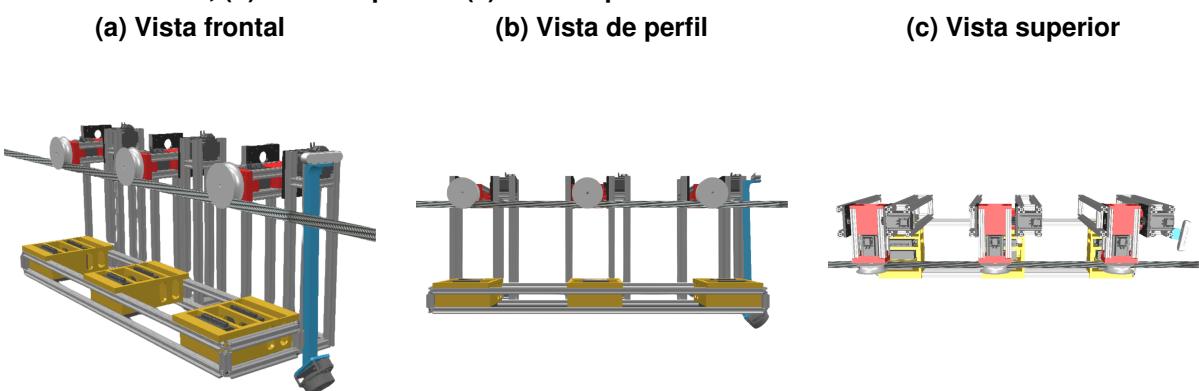
Para este projeto, foi construído um suporte físico com o objetivo de manter fixas a orientação e a posição dos sensores utilizados. O suporte foi projetado com o menor tamanho possível, desde que não comprometesse a estabilidade dos sensores. A escolha das posições relativas entre os sensores foi determinada com base em seus papéis específicos dentro do sistema robótico e nas exigências geométricas para uma coleta eficiente dos dados.

Figura 4 – Vistas do protótipo do primeiro robô secundário no ambiente de simulação: (a) Vista frontal, (b) Vista de perfil e (c) Vista superior.



Fonte: Autoria própria (2025).

Figura 5 – Vistas do protótipo do segundo robô secundário no ambiente de simulação: (a) Vista frontal, (b) Vista de perfil e (c) Vista superior.



Fonte: Autoria própria (2025).

O sensor LiDAR tem como principal função levantar informações de profundidade ao redor da linha, sendo, portanto, essencial que ele possua uma visão desobstruída do ambiente. Para maximizar a utilidade das suas leituras, o plano de varredura do sensor deve estar livre de rotações em relação ao eixo da linha. Além disso, o ângulo do plano de leitura não pode ser muito próximo ao da própria linha, pois isso limitaria a abrangência das medições. Idealmente, o plano do LiDAR estaria orientado a 90° em relação ao eixo do cabo, oferecendo uma visão lateral completa do entorno. No entanto, para que o robô seja capaz de detectar e classificar objetos à sua frente antes de se aproximar demasiadamente, é necessário inclinar o sensor em relação ao eixo do cabo. O melhor caso, nesse sentido, seria o sensor operando a 0° em relação ao eixo do cabo, mirando diretamente à frente. (Figura para demonstrar?)

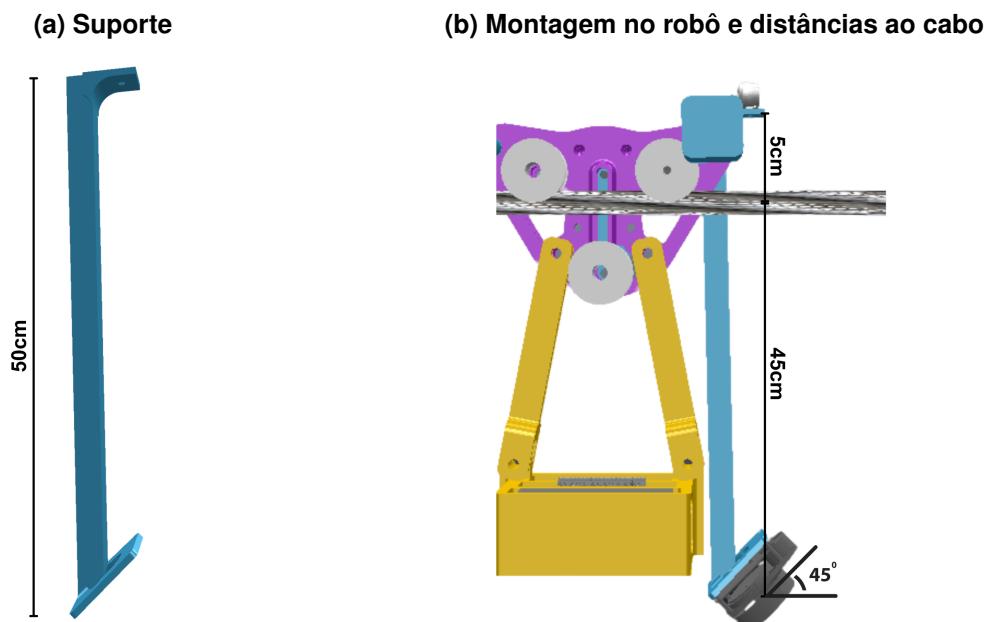
A câmera de profundidade também tem papel relevante na análise da linha e dos objetos presentes nela, sendo usada principalmente para detecção de possíveis danos físicos. Para evitar distorções e maximizar a qualidade das imagens, a câmera deve ser posicionada acima do cabo, com seu plano óptico perpendicular ao eixo da linha e voltado para frente.

Considerando os requisitos específicos de cada sensor, o suporte multimodal foi projetado para equilibrar as limitações de ambos. O ângulo de fixação do LiDAR foi definido como 45° em relação ao eixo do cabo. Essa inclinação permite a coleta tanto de dados laterais quanto frontais, desde que o sensor esteja a uma distância horizontal adequada em relação ao cabo. A câmera foi posicionada na parte superior do robô, acima do cabo, a uma distância de 5 cm a partir do centro do mesmo. (Figura com as medidas)

Foi estabelecido que a classificação dos objetos deve ser feita a partir da leitura combinada de ambos os sensores. Como a câmera de profundidade possui maior alcance visual em relação ao LiDAR (devido à sua posição elevada), definiu-se uma distância mínima de 40 cm para a detecção e classificação de objetos. Ou seja, qualquer objeto a 40 cm ou menos da frente do robô deve ser processado pelo sistema multimodal.

Para garantir essa faixa de detecção, o suporte, ilustrado na Figura 6(a), foi projetado com 50 cm de comprimento horizontal. A montagem final no robô principal, com o LiDAR posicionado a 45 cm abaixo do centro do cabo e a câmera a 5 cm acima, é demonstrada na Figura 6(b). Essa configuração assegura que o LiDAR possa detectar objetos a mais de 45 cm de distância, caso estejam abaixo do cabo, e objetos a pelo menos 40 cm se estiverem posicionados ligeiramente acima da linha. Dessa forma, o arranjo garante a eficácia do sistema de classificação sem comprometer a integridade ou a estabilidade do robô durante o deslocamento.

Figura 6 – Detalhes do sensor multimodal desenvolvido: (a) Suporte com sua dimensão horizontal e (b) Robô principal simulado indicando as distâncias dos sensores em relação ao cabo.



Fonte: Autoria própria (2025).

4.1.3 Sensores Utilizados

Para a realização deste projeto, foram selecionados dois sensores principais capazes de fornecer informações de profundidade complementares: a câmera Intel RealSense D415 e o LiDAR RPLIDAR A1M8. Ambos os sensores foram integrados ao sistema embarcado do robô para permitir a coleta de dados tridimensionais, usados para a tarefa de classificação de objetos sobre as linhas de transmissão. A escolha por sensores de profundidade com diferentes princípios de funcionamento — estereoscopia ativa e triangulação a laser — visa garantir robustez ao sistema em diferentes condições de iluminação, ângulos de incidência e posicionamentos relativos dos objetos. A seguir, são descritas as principais características técnicas de cada sensor.

4.1.3.1 Intel RealSense D415

A câmera de profundidade Intel RealSense D415, ilustrada na Fotografia 3, utiliza um sistema estereoscópico ativo para gerar mapas de profundidade com alta precisão. A tecnologia baseia-se na captura simultânea de imagens por duas câmeras infravermelhas, associadas a um projetor de padrões que melhora a detecção em superfícies com baixo contraste visual. Essa configuração permite estimar a profundidade por meio da disparidade entre os pares de imagens. O sensor é amplamente empregado em aplicações de robótica, visão computacional e automação, dada sua confiabilidade e facilidade de integração com plataformas embarcadas.

As principais especificações técnicas da RealSense D415 incluem:

- **Tecnologia de profundidade:** Estereoscopia ativa com projetor infravermelho
- **Distância mínima de operação:** Aproximadamente 0,45 m para resolução máxima
- **Precisão de profundidade:** Erro inferior a 2% a 2 m
- **Campo de visão (FOV):** $65^\circ \times 40^\circ$
- **Resolução de saída:** Até 1280×720 pixels
- **Taxa de quadros (depth):** Até 90 fps

4.1.3.2 RPLIDAR A1M8

O sensor RPLIDAR A1M8, desenvolvido pela Slamtec e apresentado na Fotografia 4, é um sensor LiDAR de baixo custo que opera por meio da técnica de triangulação a laser infravermelho. Sua principal funcionalidade está na realização de escaneamentos 2D em 360° , sendo capaz de gerar nuvens de pontos com alta densidade. Devido à sua leveza, baixo consumo energético e boa precisão, o sensor é amplamente empregado em sistemas de navegação autônoma e mapeamento em tempo real (SLAM). Um motor rotativo interno aciona o movimento contínuo do feixe de laser, permitindo medições em tempo real com ampla cobertura angular.

Fotografia 3 – Câmera de profundidade Intel RealSense D415 utilizada no projeto.



Fonte: Autoria própria (2025).

As principais especificações técnicas do RPLIDAR A1M8 são:

- **Tecnologia de medição:** Triangulação com laser infravermelho
- **Alcance de detecção:** 0,15 m a 12 m
- **Campo de visão (FOV):** 360° contínuo
- **Frequência de varredura:** 10 Hz
- **Resolução angular:** $\leq 1^\circ$
- **Frequência de amostragem:** ≥ 8000 pontos por segundo
- **Precisão de distância:** Erro inferior a 1% da distância medida
- **Interface de comunicação:** UART (TTL 3,3 V)

4.2 Firmware e Software

O funcionamento do sistema proposto depende da integração eficiente entre o hardware embarcado, os sensores e os algoritmos de controle e aquisição de dados. Para isso, foi desenvolvido um conjunto de códigos em Python e C++ utilizando o framework ROS, que facilita a comunicação entre os diversos componentes do sistema. O firmware, executado na placa Arduino, é responsável apenas pela interface direta com os drivers dos motores, enquanto o software, executado na Raspberry Pi, gerencia os sensores, o controle dos atuadores e a coleta dos dados. As seções a seguir descrevem detalhadamente a configuração dos sensores, a estrutura de controle dos robôs, tanto em ambiente simulado quanto real, e o script de coleta e armazenamento dos dados utilizados no treinamento dos modelos de classificação.

Fotografia 4 – Sensor LiDAR RPLIDAR A1M8 utilizado no projeto.



Fonte: Autoria própria (2025).

4.2.1 Controle do Robô via ROS

O controle dos motores foi realizado por meio de um microcontrolador Arduino, que funcionou como interface entre o sistema ROS e os drivers dos motores. Para essa comunicação, foi criado um tópico específico no ROS, no qual as mensagens eram do tipo `std_msgs/Int64MultiArray`. Cada posição do vetor representava um motor, e seu valor indicava, conforme a topologia do robô, a velocidade ou o número de passos a ser executado. O Arduino recebia os comandos por meio da porta serial e os repassava aos motores, sendo responsável apenas pela comunicação com os drivers. Nenhuma lógica de controle era executada no próprio Arduino, ficando essa tarefa completamente a cargo dos nós do ROS.

No robô utilizado para aquisição dos dados, existiam dois elementos principais de controle: a trava de segurança e o motor responsável pelo deslocamento linear ao longo da linha de transmissão. A trava recebia, por meio do vetor de controle, o número de passos necessários para travar e destravar o robô no cabo condutor. O motor de movimentação, por sua vez, recebia a velocidade desejada. O código de controle desenvolvido no ROS impedia que o robô iniciasse o movimento sem que a trava estivesse acionada, garantindo a segurança da estrutura e prevenindo quedas.

Para os robôs utilizados nas simulações, foram adotados os mesmos códigos utilizados no robô físico, com adaptações no tamanho do vetor de controle para refletir o número de atuadores presentes em cada topologia. O robô simulado utilizado na aquisição de dados operava de forma idêntica ao robô real, enquanto outras topologias simuladas apresentavam variações no número de motores ou atuadores, mantendo, contudo, o mesmo princípio de controle por meio de tópicos ROS e vetores de comandos.

4.2.2 Configuração dos Sensores

Na simulação, foram utilizados modelos de sensores que reproduzem as mesmas características dos sensores reais adotados na implementação prática. As configurações foram realizadas por meio das bibliotecas `rplidar_ros` e `realsense2_camera`, disponíveis no ecossistema do ROS. Para o LiDAR, não foram necessárias modificações adicionais, pois a configuração padrão já atendia aos requisitos do projeto. No caso da câmera RealSense D415, foi ajustada para operar a uma taxa de 10 Hz e com resolução de 1280×720 pixels, de modo a sincronizar sua aquisição com o LiDAR. Essa configuração resultou em uma distância mínima de leitura de aproximadamente 4,5 centímetros.

Uma vez configurados e iniciados os respectivos nós dos sensores, os dados passaram a ser publicados nos tópicos definidos pelas bibliotecas. O sensor LiDAR publica mensagens do tipo `sensor_msgs/LaserScan`, que consistem em uma estrutura padronizada para representar leituras de distância obtidas por varredura a laser. Essa mensagem contém informações como o ângulo inicial (`angle_min`) e final (`angle_max`) da varredura, o incremento angular entre medições (`angle_increment`), a distância mínima (`range_min`) e máxima (`range_max`) detectáveis, e o vetor `ranges`, que armazena as distâncias medidas, em metros, para cada ângulo. Opcionalmente, o campo `intensities` pode conter dados referentes à intensidade do sinal refletido, útil para análise de propriedades ópticas dos objetos. O campo `header`, presente em todas as mensagens ROS, fornece informações de carimbo de tempo e identificação do sistema de coordenadas (`frame_id`) associado à leitura.

Para os sensores de visão, como a câmera de profundidade RealSense, os dados são publicados em tópicos do tipo `sensor_msgs/Image`. Essa mensagem carrega as imagens ou quadros de vídeo capturados e contém metadados como a resolução da imagem (`width` e `height`), o número de bytes por linha (`step`) e os dados brutos da imagem no campo `data`. A codificação dos pixels é especificada no campo `encoding`, com formatos comuns como `"rgb8"`, `"bgr8"`, `"mono8"` ou `"yuv422"`. O campo `is_bigendian` informa a ordem dos bytes utilizada. Neste projeto, foram utilizadas imagens em escala de cinza (`grayscale`), com codificação de 8 bits por pixel para os dados simulados e de 16 bits por pixel para os dados adquiridos em ambiente real.

4.2.3 Script de Coleta

O script de coleta de dados foi desenvolvido em `Python`, com o objetivo de registrar as informações publicadas nos tópicos dos sensores utilizados. Para a câmera de profundidade, o script armazenava diretamente as imagens geradas, enquanto, para o LiDAR real, era aplicado um pré-processamento com a finalidade de restringir o campo de leitura do sensor.

Esse pré-processamento consistia em limitar o ângulo de leitura para um setor de 10 graus, centrado à frente do sensor, de modo que apenas as amostras dentro desse intervalo

fossem consideradas. Como resultado, cada varredura do LiDAR fornecia 63 leituras, correspondentes a esse setor angular reduzido.

Após iniciado, o script realizava a inscrição nos tópicos dos sensores e armazenava os dados recebidos. Como ambos os sensores estavam configurados para operar a uma frequência de 10 Hz, os dados capturados já estavam naturalmente sincronizados no tempo, dispensando a necessidade de sincronização adicional. O script permanecia em execução até atingir o número desejado de amostras, que, neste caso, foi de 160 registros por objeto observado.

4.3 Coleta dos Dados

Para a coleta dos dados, os objetos foram posicionados um a um em frente aos sensores, de modo que ficassem visíveis tanto para o LiDAR quanto para a câmera de profundidade. Em seguida, o robô era aproximado gradualmente até que o sensor LiDAR passasse a registrar leituras referentes ao objeto. Nesse momento, o script de coleta era iniciado, e o robô começava a se deslocar lentamente em direção ao objeto, conforme ilustrado na Figura

A velocidade de deslocamento era ajustada individualmente para cada objeto, com o intuito de capturar a maior variedade possível de distâncias ao longo da aproximação. A coleta era interrompida somente quando o robô alcançava a última posição em que ambos os sensores ainda conseguiam registrar o objeto de forma eficaz. Considerando que os sensores operavam a uma taxa de 10 Hz, cada sessão de coleta durava 16 segundos, o que resultava em 160 amostras por objeto.

Ao todo, foram coletadas 1.280 imagens de profundidade e 80.640 leituras de distância do LiDAR provenientes do robô principal (dados reais e simulados). Além disso, foram obtidas 640 imagens de profundidade e 40.320 leituras de distância de dados simulados de cada um dos dois robôs utilizados para validar a abordagem multimodal. Os dados foram organizados em quatro classes: amortecedor, isolador, sinalizador e nada.

A partir dessas coletas, foram criados cinco *datasets* distintos para cada origem de dado, totalizando 4 conjuntos por tipo de topologia (principal [simulado e real], secundárias [simulado]). Esses conjuntos compreendem: (i) imagens brutas da câmera de profundidade; (ii) dados brutos do LiDAR; (iii) *features* extraídas das imagens de profundidade; (iv) *features* extraídas dos dados do LiDAR; e (v) uma junção das *features* das duas modalidades. Essa organização foi essencial para a análise comparativa entre sensores e para o desenvolvimento de modelos capazes de explorar diferentes níveis de abstração e fusão de dados.

4.4 Processamento de Dados

Nesta seção são apresentadas as técnicas empregadas no processamento dos dados coletados pelos sensores, com foco na extração de *features* relevantes para a classificação dos objetos. Também são descritos os métodos utilizados para a organização e construção

dos conjuntos de dados (datasets) utilizados no treinamento dos modelos de aprendizado de máquina.

Para ambos os sensores — LiDAR e câmera de profundidade — foram aplicadas etapas de pré-processamento com o objetivo de reduzir ruídos e tornar os dados mais consistentes. Inicialmente, valores discrepantes (outliers) foram removidos, e todos os dados brutos foram truncados para conter 150 elementos, de forma a padronizar as entradas para os classificadores. Os trechos a seguir detalham os procedimentos adotados para cada sensor individualmente, abordando tanto a estrutura dos dados coletados quanto o processo de extração de características específicas (*features*) que representam propriedades dos objetos observados.

4.4.1 Dados do LiDAR

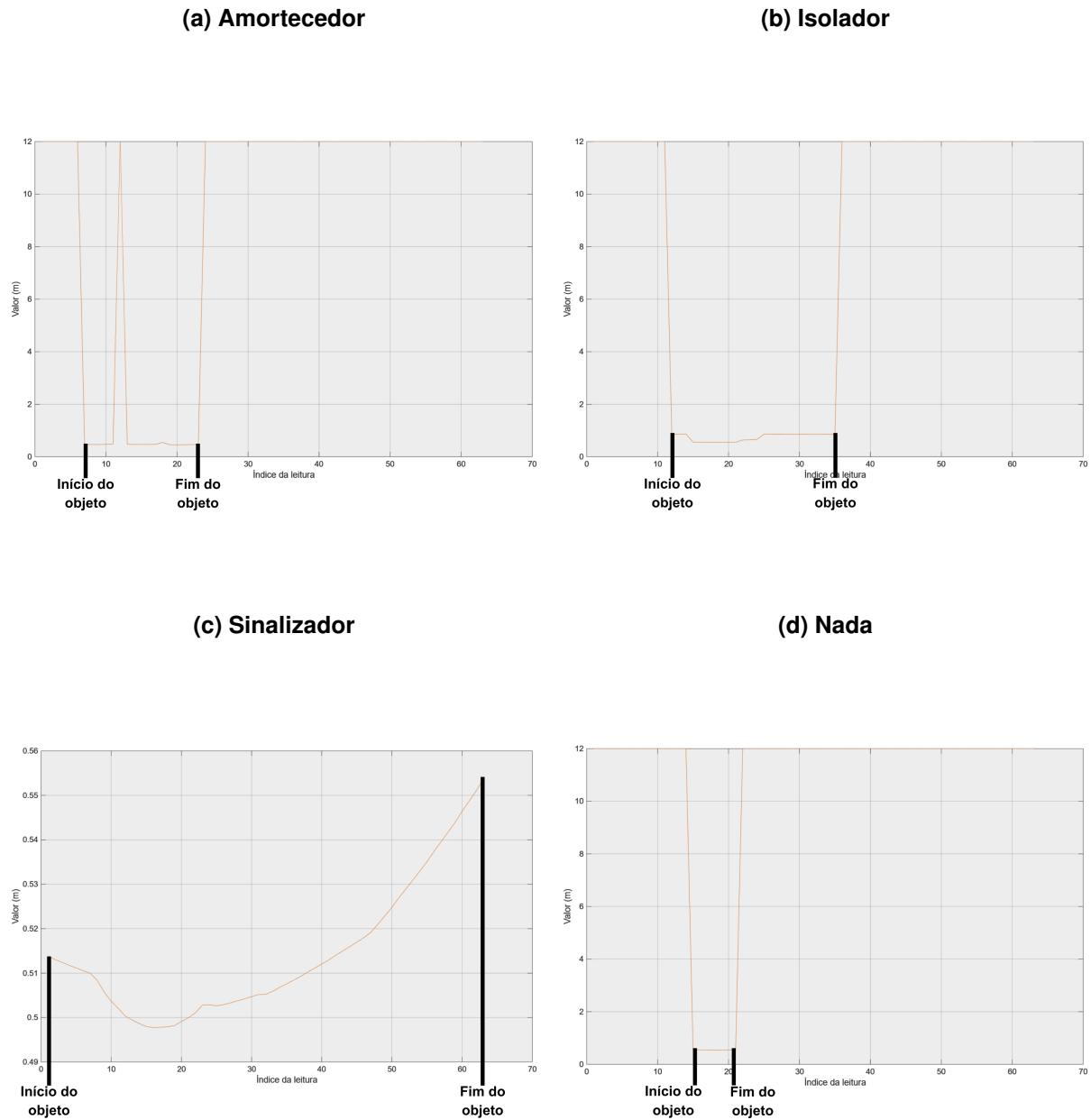
Os dados coletados pelo sensor LiDAR foram armazenados em arquivos no formato de tabela, onde cada linha representa uma varredura completa (ou *scan*) do sensor, e cada coluna corresponde a um ângulo específico da leitura. O campo de visão considerado para o processamento foi de 10° , centrado na frente do sensor, isto é, de -5° a $+5^\circ$, totalizando 63 amostras por varredura. As distâncias registradas variam entre 0 e 12 metros.

Para compor os vetores de características (*features*) utilizados nos modelos de classificação, foram extraídas duas informações principais de cada varredura do sensor:

- **Número de leituras consecutivas representando o objeto:** Essa *feature* corresponde à quantidade de amostras dentro de uma varredura que indicam a presença de um objeto. A ideia é que, quando o sensor não detecta nenhum objeto, ele retorna valores equivalentes ao "fundo de escala", ou seja, a distância máxima possível (12 metros). Isso é válido nos dados simulados, onde o ambiente é aberto, mas não se aplica diretamente ao laboratório real, onde o sensor pode registrar leituras de até 2 metros devido à proximidade de paredes e objetos. Além disso, quando o sensor real não consegue realizar a leitura, ele também retorna valores máximos. Para contornar esse problema, o algoritmo começa analisando a primeira leitura da varredura e a compara com a seguinte. Se forem iguais, assume-se que o sensor ainda não detectou um objeto. A detecção do início do objeto só é confirmada se as duas próximas leituras forem diferentes. A partir daí, o algoritmo continua até identificar o retorno ao valor inicial (indicando o fim do objeto), contando o número de amostras distintas do fundo de escala. Esse valor é proporcional ao diâmetro aparente do objeto: quanto maior o objeto, maior o número de leituras consecutivas representando sua presença.
- **Distância média do objeto:** Após identificado o intervalo das leituras correspondentes ao objeto, é calculada a média desses valores. Essa média representa a distância média entre o objeto e o sensor LiDAR durante aquela varredura. Essa *feature* é importante para distinguir objetos que, mesmo com tamanhos semelhantes, são posicionados a distâncias diferentes.

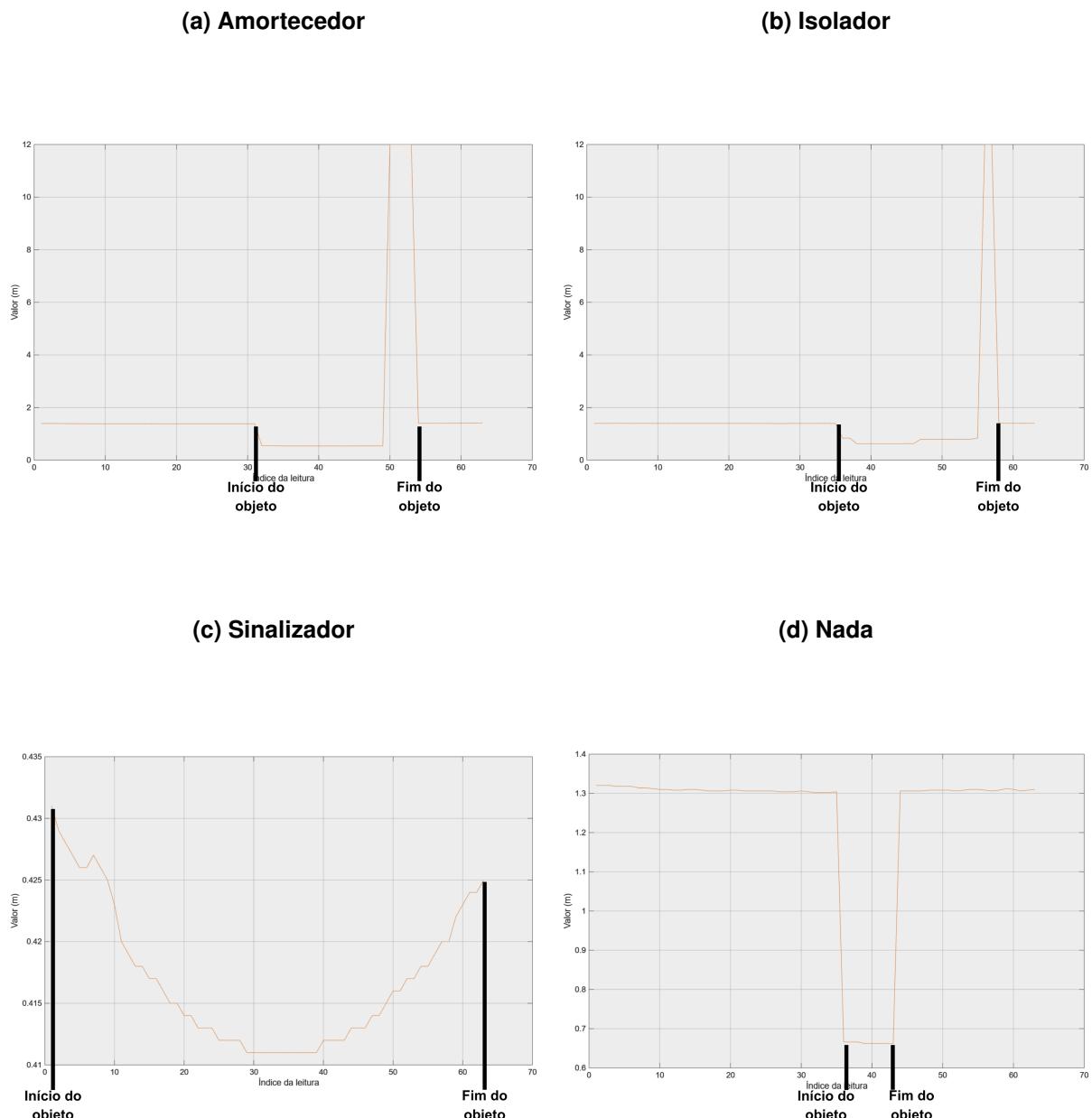
As Figuras 7 e 8 ilustram exemplos dessas leituras do sensor LiDAR, tanto para o ambiente simulado quanto para o real, respectivamente. Em cada subfigura, é possível observar o perfil de distâncias capturado para cada uma das classes de objetos (amortecedor, isolador, sinalizador e ausência de obstáculo), onde as indicações de início e término da detecção do objeto, mencionadas no algoritmo de extração da primeira *feature* estão marcadas.

Figura 7 – Leituras do sensor LiDAR para diferentes classes no ambiente simulado, com indicação do início e término da detecção do objeto: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).



Fonte: Autoria própria (2025).

Figura 8 – Leituras do sensor LiDAR para diferentes classes no ambiente real, com indicação do início e término da detecção do objeto: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).



Fonte: Autoria própria (2025).

4.4.2 Dados da RealSense

Para o processamento das imagens de profundidade obtidas pela câmera Intel RealSense D415, foram adotadas técnicas leves de extração de *features*, visando a viabilidade de uso em sistemas embarcados com recursos computacionais limitados. Dessa forma, evitou-se o uso de redes convolucionais complexas para as etapas iniciais de processamento.

Inicialmente, todas as imagens de profundidade, como as simuladas exemplificadas na Figura 9 e as reais (apresentadas na Figura 10 já com equalização de histograma aplicada,

foram redimensionadas para 460×460 pixels. Como a posição relativa do robô em relação aos objetos de interesse era fixa durante os trechos de aproximação para cada classe, essa janela foi considerada suficiente para capturar os objetos com boa visibilidade e, ao mesmo tempo, reduzir o custo de processamento nas etapas subsequentes de extração de *features*.

Figura 9 – Exemplos de imagens brutas da câmera de profundidade para diferentes classes no ambiente simulado: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).

(a) Amortecedor



(b) Isolador



(c) Sinalizador



(d) Nada



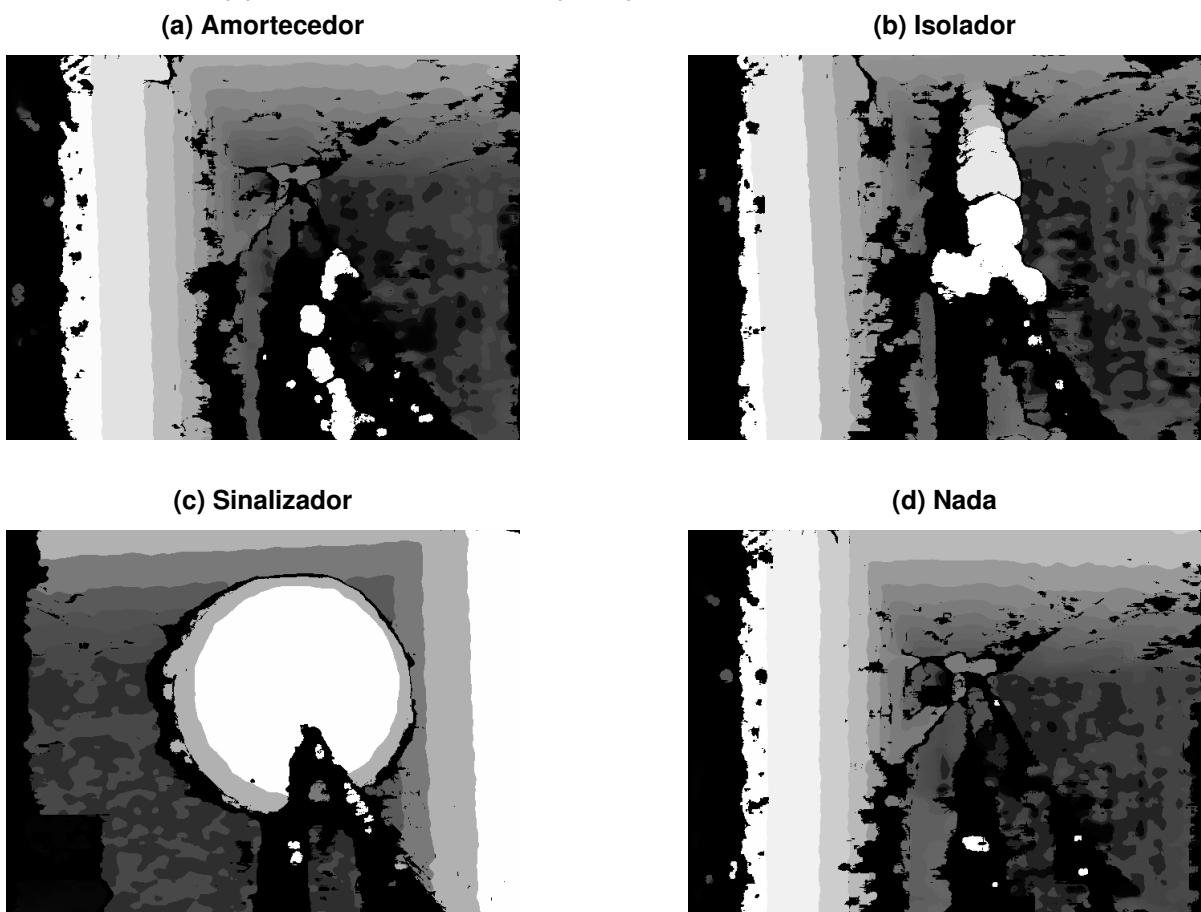
Fonte: Autoria própria (2025).

4.4.2.1 Límite de profundidade

A primeira etapa do processamento consistiu na aplicação de um limite de profundidade às imagens. Essa técnica atua como uma forma de segmentação grosseira, onde apenas objetos posicionados até uma certa distância da câmera são considerados relevantes. O limite escolhido foi de 1 metro.

No caso das imagens simuladas (com 8 bits por pixel, variando de 0 a 255 para representar 0 a 12 metros), esse corte foi feito fixando o valor de 22 como o limite superior. Assim, pixels com valores acima de 22 foram convertidos para 0, descartando regiões além de 1 metro.

Figura 10 – Exemplos de imagens brutas da câmera de profundidade (histograma equalizado) para diferentes classes no ambiente real: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).



Fonte: Autoria própria (2025).

Para as imagens reais (com 16 bits por pixel, onde cada unidade representa 1 mm), o valor limite foi definido como 1000, seguindo a mesma lógica.

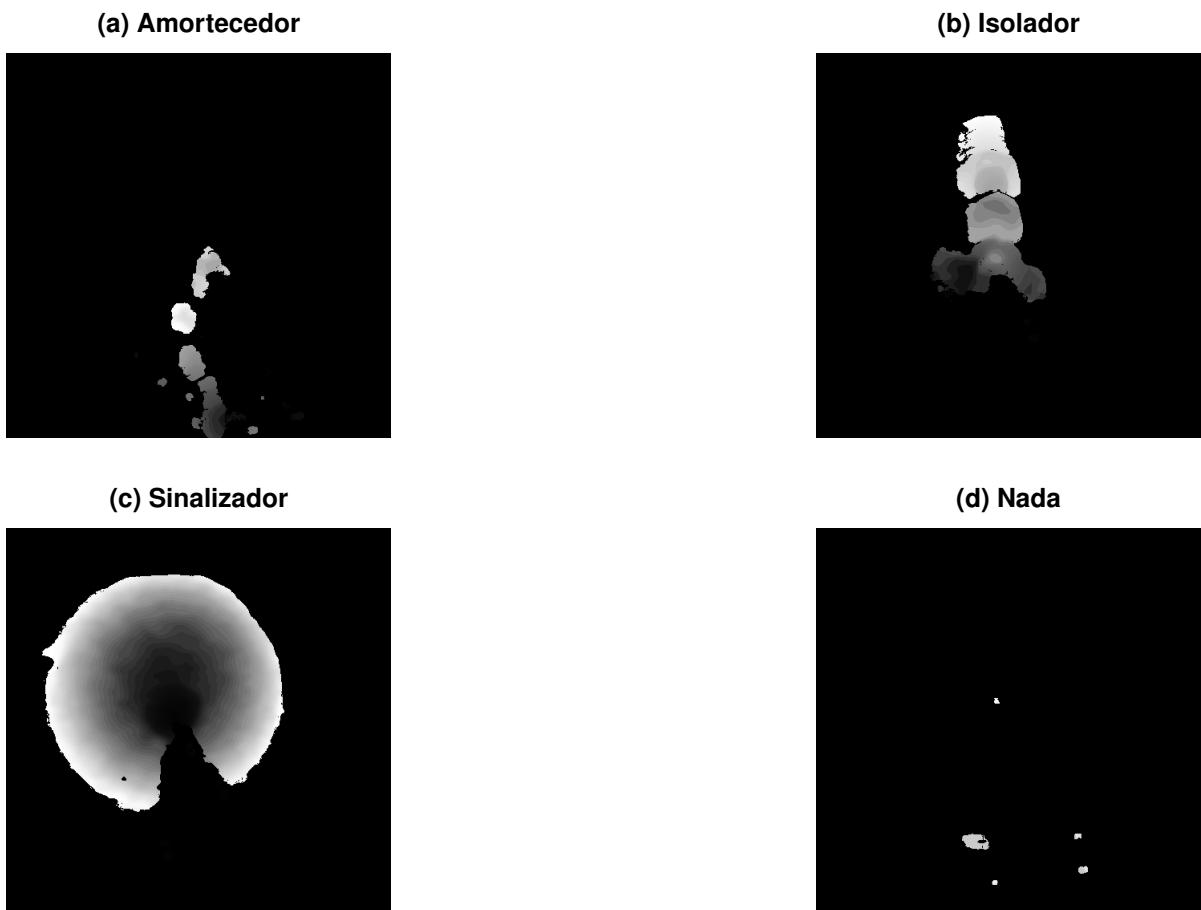
Após essa filtragem, restaram apenas os pixels associados aos objetos de interesse, facilitando a extração de características. As primeiras *features* extraídas foram a média e a variância dos valores dos pixels remanescentes, refletindo respectivamente a distância média dos objetos e a sua dispersão na imagem.

4.4.2.2 Preparação das imagens para a rede SqueezeNet

As imagens simuladas, devido à sua codificação simples, foram diretamente utilizadas para o treinamento da SqueezeNet. No entanto, as imagens reais apresentavam baixa luminosidade aparente devido à sua codificação em profundidade de 16 bits, o que dificultava sua interpretação visual e o uso direto em redes neurais.

Para resolver esse problema, foi aplicada a técnica de **equalização de histograma**, que visa melhorar o contraste da imagem redistribuindo os valores de intensidade. Essa técnica reatribui os níveis de cinza de forma que o histograma da imagem se aproxime de uma distribuição uniforme. Com isso, detalhes que antes estavam comprimidos em faixas de intensidade

Figura 11 – Exemplos de imagens da câmera de profundidade no ambiente real após aplicação do filtro de limite de profundidade: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).



Fonte: Autoria própria (2025).

estreitas passam a ser mais destacados, tornando a imagem mais nítida para o aprendizado da rede.

4.4.2.3 Contorno dos objetos

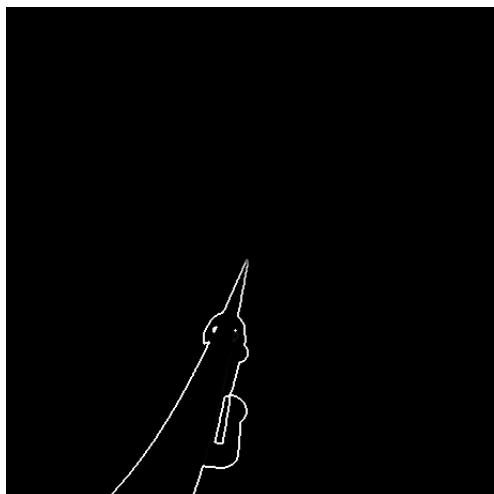
Outra *feature* extraída das imagens foi a quantidade de pixels localizados nas bordas dos objetos. Para isso, foi aplicado o **filtro Laplaciano**, um operador de detecção de bordas baseado na segunda derivada da imagem, que enfatiza regiões onde ocorrem mudanças abruptas de intensidade — ou seja, as bordas dos objetos. Foi aplicado um filtro com um **kernel** pequeno (3x3), que define a área da imagem analisada por vez na aplicação do operador (neste caso, o filtro Laplaciano). O **kernel** funciona como uma matriz que percorre a imagem, realizando operações locais — quanto menor o kernel, mais local é a detecção de variações. Como os valores dos pixels não foram multiplicados por um fator de ganho, apenas as bordas mais marcantes dos objetos foram evidenciadas, já que a imagem eram valores de profundidade as variações entre pixels era pequena se analisada em locais referentes ao mesmo objeto.

Em seguida, a imagem foi binarizada, convertendo todos os valores com variação maior que 20 centímetros para 1 e o resto para 0. O resultado desse processo, que destaca apenas os

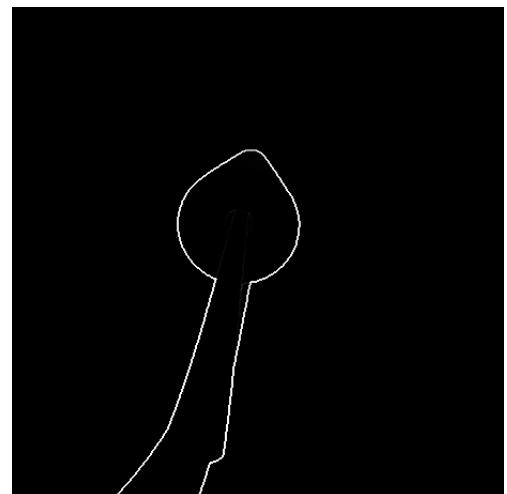
contornos principais dos objetos, pode ser visualizado para o ambiente simulado na Figura 12 e para o ambiente real na Figura 13. Em seguida, contou-se o número total de pixels com valor 1 nessa imagem binarizada, o que fornece uma estimativa do perímetro dos objetos.

Figura 12 – Exemplos do resultado da aplicação do filtro Laplaciano em imagens da câmera de profundidade para diferentes classes no ambiente simulado: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).

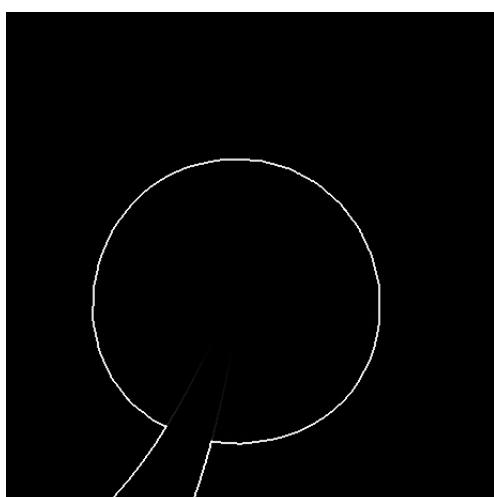
(a) Amortecedor



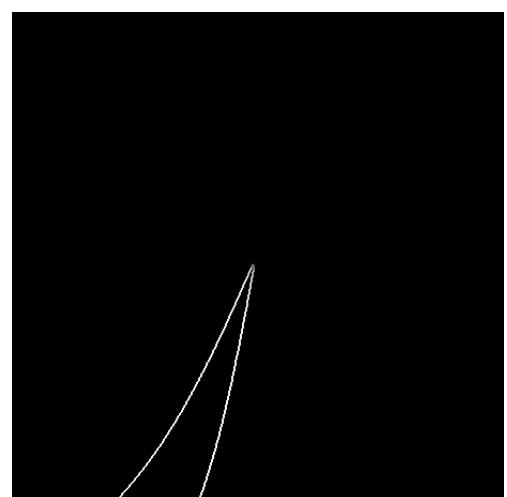
(b) Isolador



(c) Sinalizador



(d) Nada



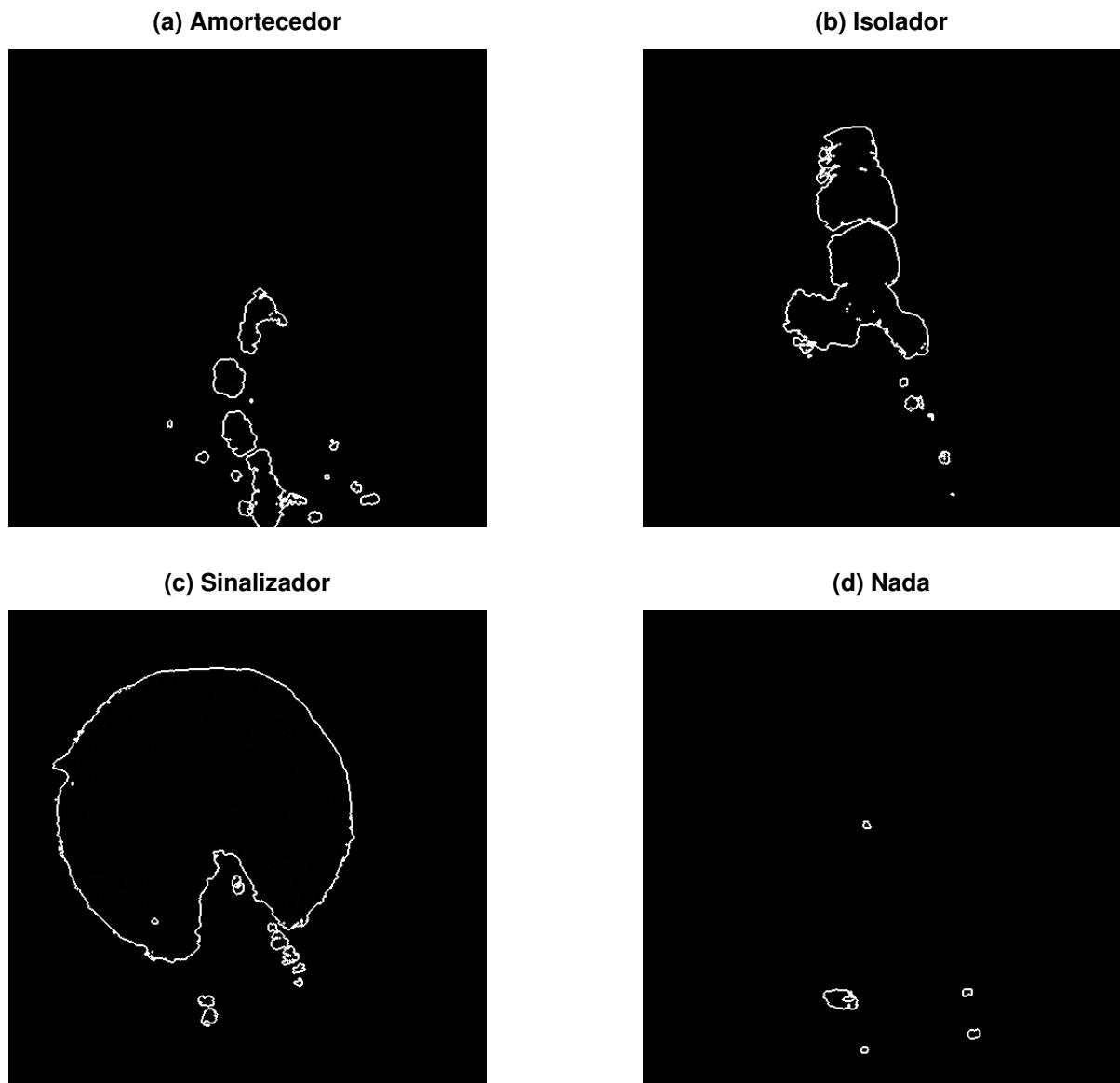
Fonte: Autoria própria (2025).

4.4.2.4 Operações morfológicas

A última *feature* extraída baseou-se em operações morfológicas aplicadas sobre a imagem com o limite de profundidade. Primeiramente, a imagem foi binarizada, com todos os pixels diferentes de zero convertidos em 1. Em seguida, foram aplicadas duas operações:

- **Fechamento:** operação composta por uma dilatação seguida de erosão. Serve para preencher pequenos buracos e eliminar pequenos vazios internos nos objetos, unificando contornos desconexos.

Figura 13 – Exemplos do resultado da aplicação do filtro Laplaciano em imagens da câmera de profundidade para diferentes classes no ambiente real: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).



Fonte: Autoria própria (2025).

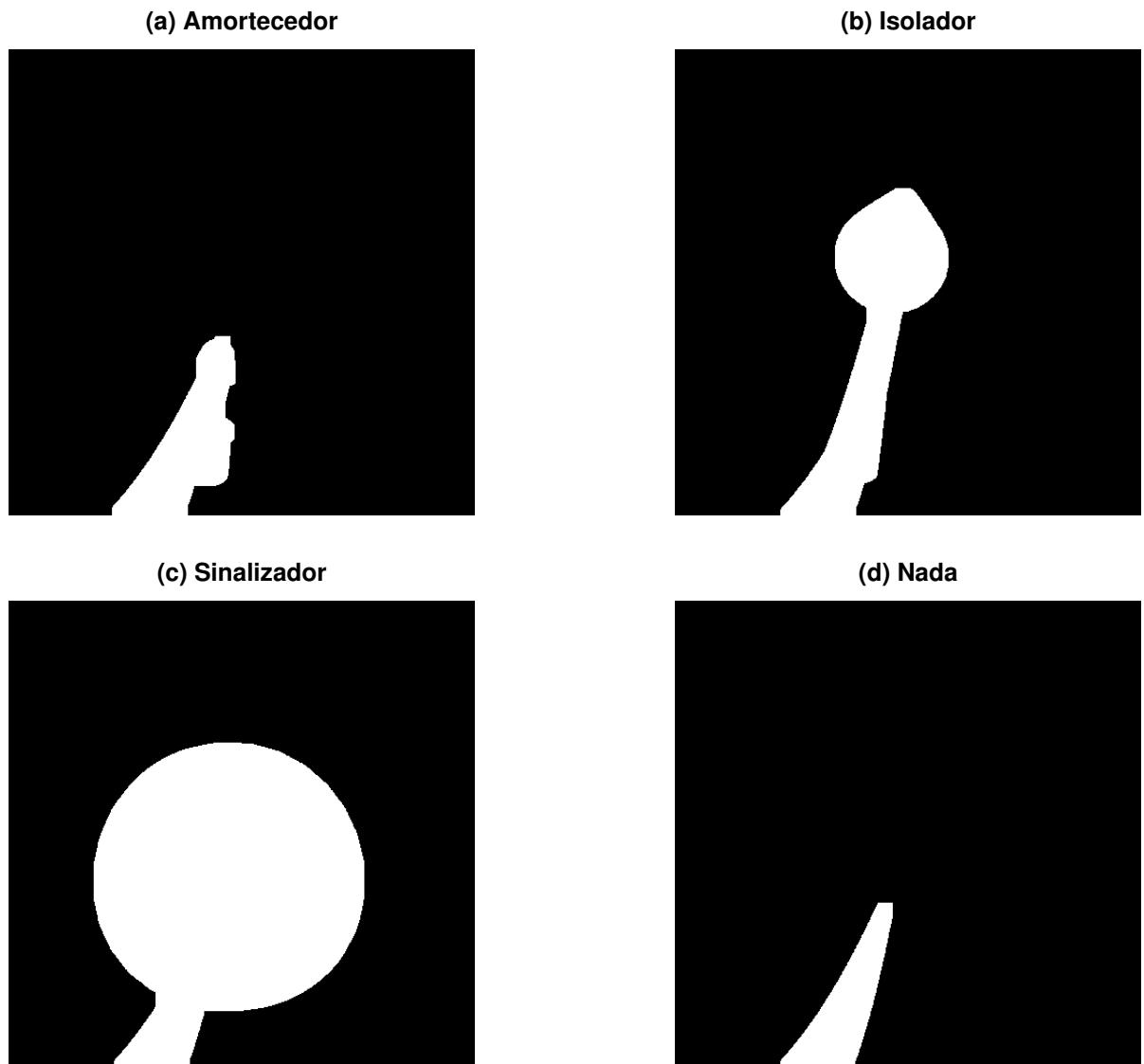
- **Abertura:** operação composta por uma erosão seguida de dilatação. Tem como função remover pequenos ruídos e detalhes irrelevantes, suavizando os contornos dos objetos.

Essas operações foram aplicadas duas vezes: uma com um **kernel grande** (15×15), o que resulta em uma suavização mais agressiva, unificando áreas extensas e eliminando ruídos significativos; e outra com um **kernel pequeno** (2×2), que atua de forma mais sutil, preservando detalhes finos.

Ao final dessas operações, a imagem resultante continha apenas a silhueta binária dos objetos principais, como exemplificado para o ambiente simulado na Figura 14 e para o ambiente

real na Figura 15. A contagem do número de pixels com valor 1 nessa imagem foi utilizada como a última *feature*, representando a área total ocupada pelo objeto visível ao sensor.

Figura 14 – Exemplos de imagens binarizadas da câmera de profundidade para diferentes classes no ambiente simulado: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).

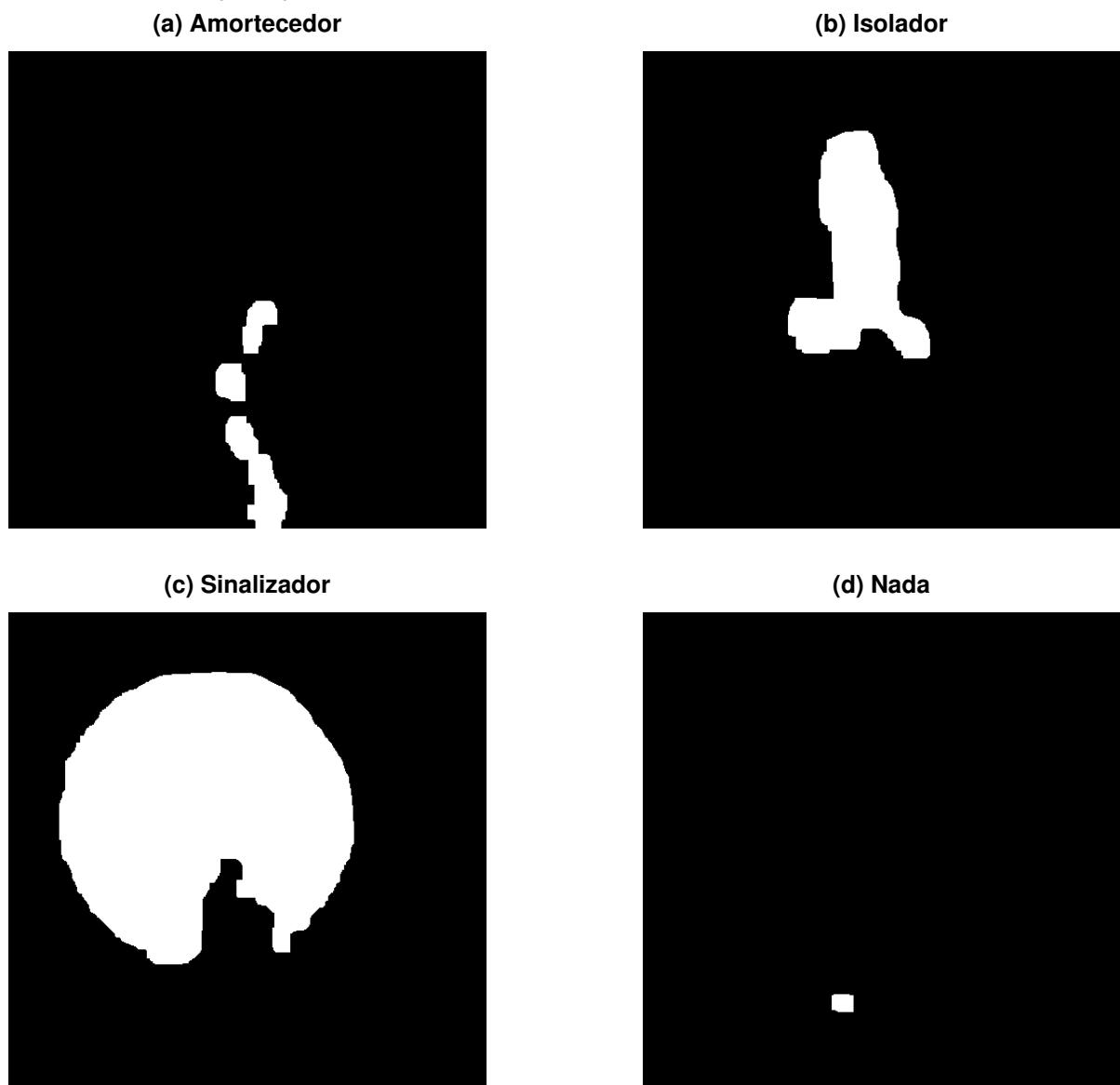


Fonte: Autoria própria (2025).

4.5 Treinamento e Validação

Nesta seção são apresentadas as configurações utilizadas para o treinamento dos modelos de aprendizado de máquina, bem como os procedimentos adotados para validação dos mesmos. Inicialmente, são detalhados os parâmetros e arquiteturas de cada um dos classificadores utilizados neste trabalho, com justificativas para as escolhas feitas em cada caso. Em seguida, são descritos os dois métodos de validação empregados: o primeiro, com validação cruzada, busca avaliar o desempenho dos modelos com os próprios dados coletados; o se-

Figura 15 – Exemplos de imagens binarizadas da câmera de profundidade para diferentes classes no ambiente real: (a) Amortecedor, (b) Isolador, (c) Sinalizador, e (d) Ausência de obstáculo (Nada).



Fonte: Autoria própria (2025).

gundo testa a capacidade dos modelos em generalizar para novas topologias de sensores, avaliando sua portabilidade em cenários distintos daquele utilizado no treinamento.

4.5.1 Configuração dos Modelos

Nesta subseção são descritas as configurações específicas adotadas para cada um dos modelos de classificação utilizados no experimento. Os modelos foram selecionados por sua ampla utilização em problemas de classificação supervisionada. Para cada modelo, são detalhados os hiperparâmetros ajustados, as estratégias de pré-processamento aplicadas e, quando relevante, as características particulares da implementação. As escolhas foram feitas com o

objetivo de equilibrar desempenho, simplicidade e viabilidade de implementação em sistemas embarcados.

4.5.1.1 Árvore de Decisão

O classificador de Árvore de Decisão foi utilizado com as configurações padrão oferecidas pela biblioteca *scikit-learn*, exceto pelo parâmetro `random_state`, definido como 42 para garantir a reproduzibilidade dos resultados. A profundidade máxima da árvore foi mantida ilimitada (`max_depth=None`), permitindo a expansão dos nós até que todas as folhas fossem puras. O número mínimo de amostras necessário para realizar uma divisão foi fixado em 2 (`min_samples_split=2`), enquanto o número mínimo de amostras exigido para formar uma folha foi mantido em 1 (`min_samples_leaf=1`). A quantidade máxima de características consideradas por divisão foi deixada como `None`, indicando que todas estavam disponíveis para análise em cada nó.

Esses parâmetros favorecem o sobreajuste aos dados de treinamento, mas esse comportamento não é problemático no contexto deste trabalho, visto que os dados seguem um padrão bem definido. O critério de divisão adotado foi o índice Gini (`criterion='gini'`), amplamente utilizado por sua simplicidade e eficiência. A estratégia de seleção das divisões foi a '`best`', escolhendo a característica e o valor de corte que maximizam a redução da impureza em cada nó.

4.5.1.2 k-Vizinhos mais próximos

Para o classificador kNN, foi adotado o valor de $k = 3$, ou seja, a classe atribuída a uma nova amostra é determinada com base nas três amostras mais próximas no conjunto de treinamento. A métrica utilizada para o cálculo das distâncias foi a distância Euclidiana, devido à sua simplicidade e adequação a espaços vetoriais contínuos. Todos os vizinhos tiveram o mesmo peso na votação (`weights='uniform'`), o que significa que cada um dos k vizinhos contribui igualmente para a decisão da classe final.

4.5.1.3 Naive Bayes

Foi utilizado o classificador GaussianNB, que assume que as características seguem uma distribuição normal (gaussiana). As probabilidades a priori das classes foram consideradas iguais, com o parâmetro `priors` definido como [0,25, 0,25, 0,25, 0,25]. O parâmetro `var_smoothing`, responsável por suavizar a variância e evitar instabilidades numéricas, foi mantido em seu valor padrão de 1×10^{-9} .

4.5.1.4 Random Forest

O classificador de Floresta Aleatória foi configurado com 100 estimadores (`n_estimators=100`), ou seja, o modelo é composto por 100 árvores de decisão independentes.

tes. O parâmetro `random_state` foi fixado em 42 para garantir reprodutibilidade. Os demais hiperparâmetros foram mantidos nos valores padrão da biblioteca, incluindo profundidade ilimitada (`max_depth=None`), divisão mínima de duas amostras (`min_samples_split=2`), critério Gini (`criterion='gini'`) e amostragem com reposição (`bootstrap=True`).

Além disso, foi incluído um estágio de normalização no pipeline utilizando o `StandardScaler`, que transforma as características para média zero e desvio padrão unitário, contribuindo para o equilíbrio das variáveis no treinamento.

4.5.1.5 Rede Neural

A arquitetura de rede neural densa utilizada foi composta por três camadas totalmente conectadas, intercaladas com funções de ativação ReLU. A primeira camada oculta contém 128 neurônios, a segunda 64, e a camada de saída possui 4 neurônios, correspondentes às quatro classes do problema. Foi utilizada a função de perda `CrossEntropyLoss`, adequada para tarefas de classificação multiclasse, e o otimizador `Adam` com taxa de aprendizado de 0,001.

O treinamento foi realizado por até 200 épocas, com tamanho de lote (`batch_size`) de 32 amostras. Implementou-se ainda uma estratégia de parada antecipada, que interrompe o treinamento caso a perda média da época fosse inferior a 0,001.

4.5.1.6 SqueezeNet

A arquitetura SqueezeNet foi adaptada para este trabalho pela modificação de sua camada final, ajustando o número de saídas para 4 (`num_classes=4`). O treinamento foi realizado com a função de perda `CrossEntropyLoss` e o otimizador `Adam`, com taxa de aprendizado de 0,001. O processo de treinamento foi conduzido por até 100 épocas, com lotes de 16 amostras (`batch_size=16`).

Foi empregada uma estratégia de parada antecipada caso a perda média por época atingisse valor inferior a 0,001. Todas as imagens utilizadas foram redimensionadas para 224×224 pixels e normalizadas com média e desvio padrão iguais a 0,5, de acordo com as recomendações usuais para esse tipo de arquitetura.

4.5.2 Validação

Foram realizados dois testes distintos para validação dos modelos de classificação. O primeiro teve como objetivo verificar se os dados coletados eram suficientemente representativos para permitir a distinção entre as classes. O segundo visou avaliar a portabilidade dos modelos treinados, bem como a capacidade do sensor multimodal de generalizar os resultados para diferentes topologias robóticas.

No primeiro teste, foi aplicada a técnica de validação cruzada com 5 dobras (*5-fold cross-validation*). O conjunto de dados foi dividido em cinco partes, mantendo a distribuição de classes equilibrada — cada uma das quatro classes apresentou 30 amostras em cada divisão, totali-

zando 120 amostras por dobra. Em cada iteração da validação cruzada, quatro partes foram utilizadas para o treinamento e a parte restante, que não participou do treinamento, foi usada para validação. Esse processo foi repetido cinco vezes, garantindo que todas as subdivisões fossem utilizadas como conjunto de validação uma vez.

Durante cada validação, foram registradas as seguintes métricas:

- **Acurácia da dobra:** proporção de classificações corretas sobre o total de amostras;
- **Perda (Loss) e época final:** quando aplicável, indicam a performance do modelo ao final do treinamento;
- **Tempo de validação:** tempo necessário para que o modelo, já treinado, classificasse as 120 amostras da dobra de validação;
- **Matriz de confusão:** registrou-se, para cada modelo treinado, a matriz de confusão resultante, com o intuito de identificar padrões de erro na classificação.

A matriz de confusão apresenta, para cada classe, a quantidade de amostras corretamente classificadas (na diagonal principal) e os erros de classificação (nas demais posições), permitindo uma análise detalhada do desempenho do modelo em cada classe individualmente.

O segundo teste de validação teve como foco a verificação da portabilidade dos modelos e da robustez do sensor multimodal frente a diferentes topologias robóticas. Os modelos foram treinados exclusivamente com os dados obtidos na simulação da primeira topologia. Em seguida, os dados das outras duas topologias foram utilizados como conjuntos de validação independentes.

As métricas registradas foram as mesmas do primeiro teste — acurácia, perda final, época final (quando aplicável) e tempo de validação. A diferença principal foi o número de amostras envolvidas: cada topologia forneceu 600 dados para validação, tornando essa avaliação mais robusta em termos estatísticos.

5 RESULTADOS E DISCUSSÕES

Todos os resultados detalhados obtidos neste projeto encontram-se compilados no **Apêndice A**. Neste apêndice, para cada modelo de aprendizado de máquina e para cada conjunto de dados avaliado, foram registrados todos os resultados de cada teste, incluindo os valores para cada dobra de validação e métricas como acurácia, tempo de validação, além da época e perda final, quando aplicáveis a cada algoritmo.

Este trabalho teve como um de seus eixos centrais a avaliação do uso de um sensor multimodal, composto por uma câmera de profundidade RealSense D415 e um sensor LiDAR RPLIDAR A1, para a classificação de objetos em linhas de transmissão. Uma vertente crucial foi a análise da portabilidade desta solução para diferentes topologias robóticas. Conforme detalhado no Apêndice A, a primeira grande seção de resultados (Seção A.1) apresenta o desempenho dos modelos quando treinados e avaliados utilizando dados do robô principal do projeto (tanto simulados quanto reais), empregando a técnica de validação cruzada. Subsequentemente, a Seção A.2 explora a robustez e a capacidade de generalização dos modelos: são apresentados os resultados de quando os modelos, treinados com os dados adquiridos pelo robô principal, foram validados com dados provenientes de duas topologias robóticas secundárias simuladas. Esta segunda análise visa determinar a viabilidade da aplicação do conceito do sensor multimodal em diferentes plataformas robóticas.

O presente capítulo, portanto, não replicará a totalidade desses dados, mas se concentrará na análise e interpretação dos principais achados. O foco será em discutir como a utilização dos dados de cada sensor individualmente (RealSense e LiDAR), das *features* extraídas de cada um, e da combinação multimodal de informações afeta o desempenho da classificação. Adicionalmente, será analisado como os diferentes modelos de aprendizado de máquina são impactados pelas mudanças nas topologias robóticas.

Para guiar esta discussão, o capítulo está estruturado da seguinte forma: inicialmente, será realizado um comparativo geral do desempenho médio dos modelos de aprendizado de máquina, considerando sua acurácia e tempo de validação em todos os testes realizados. Em seguida, será feita uma análise comparativa entre o uso de dados brutos versus *features* processadas, e o impacto da fusão sensorial. A eficácia de cada sensor (RealSense e LiDAR) para as finalidades do projeto também será discutida. Posteriormente, a análise se aprofundará por cenário experimental, começando pelos resultados do robô principal, onde se comparará o desempenho com dados simulados e reais. Na sequência, serão abordados os testes de portabilidade do sensor multimodal, analisando o comportamento dos modelos frente aos dados dos robôs secundários e discutindo a capacidade de generalização dos modelos para cada tipo de dado. Serão destacados os melhores achados, identificando as configurações ótimas para o robô principal e para os robôs secundários. Uma análise das matrizes de confusão permitirá confrontar os resultados esperados com os efetivamente observados. A discussão também abrangerá a viabilidade de embarcar os modelos propostos, considerando não apenas o tempo de validação,

mas também o uso de dados e possíveis casos de sobreajuste (*overfitting*). Finalmente, será avaliado o impacto da extração de *features* no desempenho global dos classificadores.

5.1 Comparativo Geral do Desempenho dos Modelos

A Tabela 2 a seguir apresenta um resumo consolidado do desempenho médio dos modelos de aprendizado de máquina avaliados neste trabalho. Os valores de acurácia média e tempo de validação médio foram calculados a partir dos resultados detalhados apresentados no Apêndice A, buscando oferecer uma visão comparativa entre os diferentes algoritmos e cenários de teste.

Para o cenário "Principal", que engloba os testes com dados do robô principal (tanto simulados quanto reais), a "Acurácia Média" e o "Tempo de Validação Médio" de cada modelo (k-Vizinhos mais próximos, Naive Bayes, Rede Neural e Floresta Aleatória) representam a média aritmética dos desempenhos médios obtidos em oito configurações distintas de dados (quatro tipos de datasets simulados e quatro tipos de datasets reais: dados brutos do LiDAR, features extraídas das imagens, features extraídas do LiDAR e features combinadas). A SqueezeNet foi avaliada em duas (imagens brutas simuladas e reais). Cada uma dessas configurações individuais foi, por sua vez, avaliada utilizando validação cruzada de 5 dobras, onde cada dobra de validação continha 120 amostras (30 amostras por cada uma das quatro classes).

No cenário "Multimodal", que corresponde aos testes de portabilidade para os robôs secundários, os valores apresentados para cada modelo (k-Vizinhos mais próximos, Árvore de Decisão, Naive Bayes, Rede Neural e Floresta Aleatória) também são a média aritmética dos desempenhos obtidos em oito configurações distintas: quatro tipos de datasets (dados brutos do LiDAR, features extraídas das imagens, features extraídas do LiDAR e features combinadas) para o primeiro robô secundário, e as mesmas quatro para o segundo robô secundário. Para a SqueezeNet, a média no cenário multimodal considera os resultados com imagens brutas para cada um dos dois robôs secundários.

É importante ressaltar uma diferença fundamental no cálculo e interpretação do tempo de validação entre os cenários. Enquanto no cenário "Principal", o tempo de validação médio de cada configuração de dados é derivado de testes em dobras contendo 120 amostras cada (30 para cada classe), nos testes de portabilidade do cenário "Multimodal", os modelos (treinados com dados do robô principal) foram validados utilizando o conjunto de dados completo de cada robô secundário. Cada um desses conjuntos de validação dos robôs secundários totaliza 600 amostras (150 amostras por cada uma das quatro classes). Portanto, o "Tempo de Validação Médio" reportado na Tabela 2 para o cenário "Multimodal" reflete o processamento de um volume de dados de validação cinco vezes maior em cada teste individual que compõe a média, em comparação com cada dobra do cenário "Principal". Contudo, uma relação direta entre o tempo de validação e a quantidade de amostras não pode ser estabelecida de forma simplista, já que diversos outros fatores influenciam o tempo de processamento dos modelos. Dentre eles, destacam-se a complexidade intrínseca de cada algoritmo (por exemplo, a busca por vizinhos no

k-NN versus as operações matriciais em redes neurais), a eficiência da implementação do modelo na biblioteca utilizada (scikit-learn ou PyTorch), e a própria natureza dos dados de entrada (dados brutos versus *features* processadas, que podem ter dimensionalidades e tipos diferentes). Além disso, o estado do sistema computacional no momento da execução do teste, como carga de processamento de outras tarefas ou variações no acesso a recursos, também pode introduzir pequenas variações no tempo de validação que não são diretamente proporcionais ao número de amostras. Dessa forma, a comparação direta dos valores absolutos do 'Tempo de Validação Médio' entre os cenários 'Principal' e 'Multimodal' para um mesmo modelo pode ser menos informativa do que a análise comparativa dos tempos de validação entre os diferentes modelos dentro de um mesmo cenário. A avaliação da eficiência temporal, portanto, deve ser contextualizada preferencialmente dentro de cada cenário experimental específico.

Tabela 2 – Resumo da Acurácia Média e Tempo de Validação Médio por Modelo e Cenário.

Modelo	Cenário	Acurácia Média (%)	Tempo de Validação Médio (s)
k-NN	Principal	97.15	0.003055
	Multimodal	74.25	0.013950
Árvore de Decisão	Principal	99.17	0.000128
	Multimodal	67.17	0.000174
Naive Bayes	Principal	95.52	0.000244
	Multimodal	54.08	0.000399
Rede Neural	Principal	94.59	0.003588
	Multimodal	77.00	0.014177
Floresta Aleatória	Principal	99.36	0.005717
	Multimodal	69.35	0.007775
SqueezeNet	Principal	100.00	0.659450
	Multimodal	97.67	3.356276

Fonte: Autoria própria (2025), com base nos dados do Apêndice A.

Observando a Tabela 2, que consolida o desempenho médio dos modelos, percebe-se que para o cenário "**Principal**" todos os modelos alcançaram uma acurácia média elevada, indicando que, de maneira geral, seriam capazes de realizar a classificação dos objetos de forma satisfatória para os propósitos do projeto. Este desempenho sugere também que os *datasets* construídos a partir do robô principal conseguiram representar adequadamente os objetos de interesse, permitindo aos modelos aprenderem padrões distintivos.

A diferença mais notável entre os modelos neste cenário reside no tempo médio de validação. A Árvore de Decisão consistentemente apresentou o menor tempo, o que se deve à sua simplicidade computacional após o treinamento. Uma vez construída, a classificação envolve uma série de comparações diretas de atributos com valores limiares, que são operações computacionalmente leves. As árvores geradas, em sua maioria, não necessitaram de mais do que quatro níveis para classificar os objetos, o que contribui para essa rapidez.

Em contraste, a rede convolucional SqueezeNet registrou o maior tempo de validação. Diversos fatores contribuem para isso: primeiramente, os dados de entrada consistem em imagens de 224×224 pixels, representando um volume de dados consideravelmente maior por amostra em comparação com os vetores de *features* ou dados brutos do LiDAR utilizados pelos

outros modelos. Essas imagens são então processadas por múltiplas camadas convolucionais, que realizam operações matriciais intensivas para extrair hierarquias de *features* e efetuar a classificação. A natureza custosa dessas operações é evidenciada ao comparar o tempo de validação entre os cenários "Principal" e "Multimodal" para a SqueezeNet: um aumento de cinco vezes no número de amostras de validação (de 120 para 600) resultou em um aumento quase proporcional no tempo de validação. Isso demonstra que o principal gargalo para este modelo é, de fato, a complexidade das operações realizadas em cada imagem.

Os demais modelos (k-Vizinhos mais próximos, Naive Bayes, Rede Neural MLP e Floresta Aleatória) apresentaram tempos de validação relativamente baixos no cenário "Principal". Excluindo a SqueezeNet, todos os outros modelos demonstraram tempos de validação que, teoricamente, permitiriam sua execução em taxas compatíveis com a frequência de captura dos sensores (10 Hz), o que é um indicativo positivo para uma eventual implementação embarcada.

Ao analisar os resultados do cenário "Multimodal" apresentados na Tabela 2, observa-se uma queda considerável na acurácia média da maioria dos modelos em comparação com o cenário "Principal". Com exceção da SqueezeNet, os demais classificadores (k-Vizinhos mais próximos, Árvore de Decisão, Naive Bayes, Rede Neural MLP e Floresta Aleatória) apresentaram um desempenho que, em média, poderia não ser considerado satisfatório para a aplicação robusta em diferentes topologias robóticas sem um novo treinamento ou ajuste fino.

Essa redução na performance pode ser, em parte, atribuída ao fenômeno de sobreajuste dos modelos mais simples aos dados da topologia do robô principal, utilizada para o treinamento. Como os dados de validação no cenário multimodal provêm de topologias robóticas distintas daquela usada no treinamento, os modelos, que podem ter se ajustado excessivamente às particularidades e à distribuição específica dos dados do robô principal, encontram dificuldade em generalizar para as novas configurações. Essa questão do sobreajuste e da capacidade de generalização será explorada com maior profundidade em seções posteriores deste capítulo, ao analisarmos os resultados dos robôs secundário individualmente.

É notável que a SqueezeNet não apresentou uma queda tão acentuada em sua acurácia média no cenário multimodal, mantendo um desempenho elevado. Este comportamento pode ser creditado à sua arquitetura de rede neural convolucional, especialmente se utilizada com pesos pré-treinados em grandes volumes de dados, que possui uma capacidade intrínseca superior de extrair *features* robustas e generalizáveis diretamente das imagens. Mesmo diante de variações de angulação, distância ou pequenas diferenças na perspectiva do sensor causadas pela mudança de topologia robótica, a rede consegue identificar padrões relevantes. Esse resultado, embora positivo, era de certa forma esperado, considerando o maior custo computacional associado ao processamento de imagens pela SqueezeNet, conforme discutido anteriormente.

Quanto ao tempo de validação dos modelos no cenário "Multimodal", aplicam-se as mesmas considerações sobre a complexidade relativa de cada algoritmo feitas para o cenário "Principal". Observa-se um aumento no tempo médio de validação para todos os modelos, o que é esperado devido ao aumento no número de amostras utilizadas para a validação neste cenário (600 amostras por dataset de robô secundário, em contraste com as 120 amostras por dobra da

validação cruzada no cenário "Principal"). No entanto, é interessante notar que o aumento no tempo de validação não foi estritamente proporcional ao aumento no volume de amostras para todos os modelos, sugerindo que outros fatores, como a eficiência de processamento em lote e a natureza dos dados, também influenciam a escalabilidade temporal dos algoritmos.

5.1.1 Análise Detalhada no Cenário Principal Simulado

Aprofundando a análise do desempenho dos modelos no cenário do robô principal com dados simulados, a Tabela 3 apresenta um resumo das médias de acurácia e tempo de validação para cada um dos cinco modelos de aprendizado de máquina analisados (k-Vizinhos mais próximos, Árvore de Decisão, Naive Bayes, Rede Neural e Floresta Aleatória), considerando os diferentes tipos de *datasets* utilizados: dados brutos do LiDAR, *features* extraídas das imagens, *features* extraídas do LiDAR e a combinação de *features* de ambas as fontes. Estes valores foram calculados a partir das médias de validação cruzada de 5 dobras detalhadas no Apêndice A.1.1, excluindo-se a SqueezeNet, que foi avaliada separadamente com imagens brutas. Esta análise permite observar mais de perto como cada tipo de representação dos dados impactou o desempenho de cada classificador neste ambiente controlado.

Tabela 3 – Acurácia Média e Tempo de Validação Médio por Modelo e Tipo de Dado (Robô Principal - Simulado).

Modelo	Tipo de Dado	Acurácia Média (%)	Tempo de Validação Médio (s)
k-NN	LiDAR bruto	97.33	0.003360
	Features Imagem	100.00	0.003300
	Features LiDAR	95.83	0.002800
	Features Combinadas	100.00	0.002820
Árvore de Decisão	LiDAR bruto	99.50	0.000100
	Features Imagem	100.00	0.000127
	Features LiDAR	98.17	0.000100
	Features Combinadas	100.00	0.000136
Naive Bayes	LiDAR bruto	96.60	0.000300
	Features Imagem	99.83	0.000208
	Features LiDAR	93.67	0.000260
	Features Combinadas	99.83	0.000210
Rede Neural	LiDAR bruto	96.73	0.003500
	Features Imagem	99.83	0.004200
	Features LiDAR	93.90	0.003770
	Features Combinadas	99.67	0.003600
Floresta Aleatória	LiDAR bruto	99.70	0.006000
	Features Imagem	100.00	0.005400
	Features LiDAR	97.50	0.005660
	Features Combinadas	100.00	0.005600

Fonte: Autoria própria (2025), com base nas Tabelas do Apêndice A.1.1.

Analizando a Tabela 3, que apresenta os resultados para o robô principal em ambiente simulado, observa-se que o tipo de dado utilizado nos modelos não causou um impacto drasticamente negativo nas acuráncias médias. Com exceção das *features* do LiDAR, que levaram a

acurárias ligeiramente inferiores para kNN, Naive Bayes e Rede Neural, a maioria das abordagens manteve um desempenho muito alto, frequentemente atingindo 100% ou valores próximos. Isso sugere que, para este cenário simulado e controlado, a utilização de dados de apenas um dos sensores (processados como *features* de imagem ou mesmo dados brutos do LiDAR para alguns modelos) já poderia ser suficiente para realizar a classificação dos objetos com alta eficácia.

Dentre as configurações avaliadas para os modelos neste cenário simulado, a combinação que se destacou com o melhor equilíbrio entre máxima acurácia e menor tempo de validação foi a Árvore de Decisão utilizando *features* extraídas das imagens de profundidade. Conforme detalhado na Tabela 15 do Apêndice A, este modelo alcançou 100% de acurácia em todas as cinco dobras da validação cruzada, com um tempo de validação médio de apenas 0.000127 segundos (ou 0.127 ms).

5.1.2 Análise Detalhada no Cenário Principal Real

Prosseguindo com a análise do desempenho no cenário do robô principal, esta subseção foca nos resultados obtidos com dados reais, coletados em ambiente de laboratório. A Tabela 4 sumariza as médias de acurácia e tempo de validação para os cinco modelos de aprendizado de máquina (k-Vizinhos mais próximos, Árvore de Decisão, Naive Bayes, Rede Neural e Floresta Aleatória). Estes dados são apresentados de acordo com os diferentes tipos de *datasets* utilizados: dados brutos do LiDAR, *features* extraídas das imagens, *features* extraídas do LiDAR e a combinação de *features* de ambas as fontes. Os valores compilados foram calculados a partir das médias de validação cruzada de 5 dobras, detalhadas na seção A.1.2 do Apêndice, que trata dos dados reais (excluindo-se a SqueezeNet, avaliada separadamente com imagens brutas). O objetivo desta análise é identificar como cada forma de representação dos dados, agora sob a influência das imperfeições e ruídos do mundo real, impactou o desempenho de cada classificador.

Ao analisar a Tabela 4, que detalha o desempenho com dados reais do robô principal, percebe-se que as acurárias médias dos modelos clássicos permanecem, em geral, bastante elevadas. Mesmo o pior caso observado, o modelo Naive Bayes utilizando dados brutos do LiDAR, que alcançou 81.33% de acurácia, ainda pode ser considerado um resultado relativamente decente para uma linha de base. Este panorama geral sugere que os modelos, mesmo diante das imperfeições e ruídos inerentes aos dados reais, mantêm uma boa capacidade de classificação.

Com a análise dos dados reais, um comportamento curioso emerge nos resultados da Rede Neural (MLP). Intuitivamente, seria esperado que um maior volume de informações ou a combinação de diferentes fontes de dados (como imagens e LiDAR) levasse a um desempenho superior ou, no mínimo, equivalente. No entanto, observa-se que ao utilizar as *features* extraídas das imagens da câmera de profundidade, a acurácia média do modelo (87.84%) foi inferior à obtida com as *features* do LiDAR (95.00%) e até mesmo com os dados brutos do

Tabela 4 – Acurácia Média e Tempo de Validação Médio por Modelo e Tipo de Dado (Robô Principal - Real).

Modelo	Tipo de Dado	Acurácia Média (%)	Tempo de Validação Médio (s)
k-Vizinhos mais próximos	LiDAR bruto	90.90	0.003460
	Features Imagem	100.00	0.003000
	Features LiDAR	93.17	0.002800
	Features Combinadas	100.00	0.002900
Árvore de Decisão	LiDAR bruto	99.37	0.000159
	Features Imagem	99.67	0.000137
	Features LiDAR	97.50	0.000134
	Features Combinadas	99.5	0.000134
Naive Bayes	LiDAR bruto	81.33	0.000276
	Features Imagem	99.33	0.000236
	Features LiDAR	93.90	0.000244
	Features Combinadas	99.67	0.000214
Rede Neural	LiDAR bruto	93.17	0.003855
	Features Imagem	87.84	0.003025
	Features LiDAR	95.00	0.003295
	Features Combinadas	90.60	0.003456
Floresta Aleatória	LiDAR bruto	100.00	0.005957
	Features Imagem	99.67	0.005623
	Features LiDAR	98.33	0.005730
	Features Combinadas	99.67	0.005763

Fonte: Autoria própria (2025), com base nas tabelas do Apêndice A.1.2.

LiDAR (93.17%). Isso sugere que, para a arquitetura de MLP utilizada, os dados processados da câmera podem ter introduzido ruído ou informações menos discriminatórias que os dados do LiDAR, ou que a combinação desses dados não foi otimizada pela rede da forma esperada. Este comportamento não se repetiu de forma tão evidente nos outros modelos, o que levanta a discussão sobre como a arquitetura específica da rede neural e a natureza dos dados de entrada devem ser cuidadosamente consideradas e ajustadas para evitar impactos negativos no desempenho.

Apesar do bom desempenho geral de várias configurações, se buscarmos a melhor combinação em termos de acurácia e tempo de validação com dados reais para os modelos clássicos listados na Tabela 4, a Árvore de Decisão com *features* das imagens de profundidade novamente se apresenta como a melhor candidata, alcançando 99.67% de acurácia com um tempo de validação médio extremamente baixo (0.000137s). Contudo, é importante notar que o k-Vizinhos mais próximos e a Floresta Aleatória também atingiram 100% de acurácia com *features* de imagem e *features* combinadas em alguns casos, embora com tempos de validação ligeiramente superiores aos da Árvore de Decisão.

5.1.3 Análise dos Testes de Portabilidade no Primeiro Robô Secundário

A Tabela 5 compila os resultados de desempenho dos modelos de aprendizado de máquina quando treinados com dados do robô principal e validados com dados provenientes da

primeira topologia de robô secundário, em ambiente simulado. Esta análise visa avaliar a capacidade de generalização e portabilidade do sensor multimodal e dos modelos para uma configuração robótica distinta daquela utilizada no treinamento. Os dados apresentados são as métricas de acurácia e tempo de validação para cada um dos cinco modelos, considerando os quatro tipos de *datasets*: dados brutos do LiDAR, *features* extraídas das imagens, *features* extraídas do LiDAR e a combinação de *features*. A SqueezeNet, avaliada com imagens brutas, não está incluída nesta tabela resumo. Os valores aqui apresentados são extraídos diretamente das tabelas correspondentes na seção A.2.1 do Apêndice.

Tabela 5 – Acurácia e Tempo de Validação por Modelo e Tipo de Dado (Primeiro Robô Secundário - Multimodal).

Modelo	Tipo de Dado	Acurácia (%)	Tempo de Validação (s)
k-Vizinhos mais próximos	LiDAR bruto	47.83	0.019270
	Features Imagem	70.33	0.012015
	Features LiDAR	79.83	0.012294
	Features Combinadas	70.33	0.011850
Árvore de Decisão	LiDAR bruto	46.00	0.000213
	Features Imagem	69.67	0.000170
	Features LiDAR	79.00	0.000156
	Features Combinadas	69.67	0.000151
Naive Bayes	LiDAR bruto	25.00	0.000745
	Features Imagem	69.83	0.000264
	Features LiDAR	63.00	0.000232
	Features Combinadas	69.83	0.000245
Rede Neural	LiDAR bruto	25.00	0.000745
	Features Imagem	75.00	0.015986
	Features LiDAR	69.17	0.015490
	Features Combinadas	100.00	0.016043
Floresta Aleatória	LiDAR bruto	46.67	0.007682
	Features Imagem	69.67	0.007539
	Features LiDAR	80.33	0.008353
	Features Combinadas	70.33	0.007828

Fonte: Autoria própria (2025), com base nas tabelas do Apêndice A.2.1.

Ao analisar a Tabela 5, que apresenta os resultados dos testes de portabilidade para o primeiro robô secundário, evidencia-se uma queda significativa no desempenho da maioria dos modelos. Em muitos casos, as acurárias médias registradas ficaram abaixo de 80%, atingindo patamares tão baixos quanto 25% para o Naive Bayes e a Rede Neural com dados brutos do LiDAR. Uma acurácia de 25% em um problema de quatro classes sugere que o modelo não conseguiu aprender a diferenciar as classes, indicando que, nessas configurações específicas, os modelos não seriam sequer capazes de realizar as classificações desejadas em uma nova topologia robótica sem retreinamento.

A Árvore de Decisão manteve-se como o modelo com o menor tempo de validação, uma característica consistente com sua simplicidade. No entanto, mesmo em sua melhor configuração para este cenário (utilizando *features* do LiDAR), a acurácia atingiu 79.00%, o que pode não ser considerado um nível de classificação robusto o suficiente para aplicações críticas.

O destaque neste cenário de teste de portabilidade foi a Rede Neural (MLP) quando utilizada com *features* combinadas, alcançando 100% de acurácia. Este resultado é particularmente interessante, pois sugere uma maior capacidade de adaptação e generalização deste modelo para cenários não vistos durante o treinamento, especialmente quando alimentado com um conjunto mais rico e diversificado de *features*. Isso contrasta com seu desempenho inferior utilizando apenas dados brutos do LiDAR nesta mesma topologia.

5.1.4 Análise dos Testes de Portabilidade no Segundo Robô Secundário

A Tabela 6 apresenta os resultados de desempenho dos modelos de aprendizado de máquina quando treinados com dados do robô principal e validados com dados provenientes da segunda topologia de robô secundário, também em ambiente simulado. Similarmente à análise anterior, o objetivo é avaliar a capacidade de generalização e portabilidade do sensor multimodal e dos modelos para esta outra configuração robótica distinta. São apresentadas as métricas de acurácia e tempo de validação para cada um dos cinco modelos clássicos, considerando os quatro tipos de *datasets*: dados brutos do LiDAR, *features* extraídas das imagens, *features* extraídas do LiDAR e a combinação de *features*. A SqueezeNet, avaliada com imagens brutas, não está incluída nesta tabela resumo. Os valores aqui apresentados são extraídos diretamente das tabelas correspondentes na seção A.2.2 do Apêndice.

Tabela 6 – Acurácia e Tempo de Validação por Modelo e Tipo de Dado (Segundo Robô Secundário - Multimodal).

Modelo	Tipo de Dado	Acurácia (%)	Tempo de Validação (s)
k-Vizinhos mais próximos	LiDAR bruto	78.00	0.019251
	Features Imagem	86.67	0.011986
	Features LiDAR	74.33	0.013072
	Features Combinadas	86.67	0.011865
Árvore de Decisão	LiDAR bruto	52.67	0.000230
	Features Imagem	61.17	0.000149
	Features LiDAR	74.50	0.000177
	Features Combinadas	84.67	0.000149
Naive Bayes	LiDAR bruto	36.50	0.000785
	Features Imagem	47.33	0.000305
	Features LiDAR	71.67	0.000255
	Features Combinadas	49.50	0.000358
Rede Neural	LiDAR bruto	76.33	0.015708
	Features Imagem	88.50	0.017994
	Features LiDAR	93.50	0.015820
	Features Combinadas	88.50	0.015628
Floresta Aleatória	LiDAR bruto	77.50	0.007995
	Features Imagem	64.83	0.008024
	Features LiDAR	80.67	0.007872
	Features Combinadas	64.83	0.006904

Fonte: Autoria própria (2025), com base nas Tabelas 56 a 59 do Apêndice A.2.2..

Ao examinar a Tabela 6, que sumariza o desempenho dos modelos na segunda topologia de robô secundário, nota-se que, diferente da primeira topologia secundária, não houve casos em que os modelos falharam completamente em classificar os dados (ou seja, acurácia de 25%). Contudo, a acurácia média geral para a maioria das configurações permaneceu abaixo de 88%, o que ainda indica desafios na portabilidade direta dos modelos treinados com o robô principal.

Uma exceção notável é o modelo Naive Bayes, que, com exceção do uso de *features* do LiDAR (71.67%), apresentou acurárias consistentemente abaixo de 50% para os demais tipos de dados nesta topologia. Este resultado sugere que, para o segundo robô secundário, o Naive Bayes não demonstrou ser uma escolha adequada para uma classificação confiável na maioria dos cenários de dados testados.

A Árvore de Decisão manteve sua característica de apresentar o menor tempo de validação. Seu melhor desempenho em termos de acurácia para esta topologia foi de 84.67% utilizando *features* combinadas. Embora este valor seja um pouco superior ao seu melhor caso na primeira topologia secundária (79.00% com *features* do LiDAR), o ganho não é substancial a ponto de destacá-la como a melhor opção geral para este cenário de portabilidade.

Novamente, a Rede Neural (MLP) demonstrou um desempenho superior, apresentando a melhor combinação de acurácia para esta topologia ao utilizar as *features* extraídas do LiDAR, alcançando 93.50%. Isso reforça a observação anterior de que a Rede Neural, mesmo com uma arquitetura simples, possui uma boa capacidade de generalização quando alimentada com representações de dados mais processadas e potencialmente mais discriminatórias, como as *features* do LiDAR. Curiosamente, o uso de *features* combinadas resultou em uma leve queda na acurácia (88.50%) em comparação com o uso exclusivo das *features* do LiDAR, evidenciando mais uma vez a importância da seleção e combinação criteriosa dos dados de entrada durante a construção e treinamento de redes neurais, pois a simples adição de mais dados nem sempre se traduz em melhor desempenho.

5.2 Análise da Influência do Tipo de Dado

Para aprofundar a análise da influência dos diferentes tipos de dados e da fusão sensorial no desempenho da classificação, a Tabela 7 apresenta um resumo consolidado. Para cada um dos quatro cenários experimentais avaliados neste trabalho — Robô Principal em ambiente Simulado, Robô Principal em ambiente Real, testes de portabilidade com o Primeiro Robô Secundário (simulado) e com o Segundo Robô Secundário (simulado) — a tabela exibe a acurácia média obtida com cada tipo de representação de dados. Foram considerados os seguintes tipos de dados: Imagens Brutas (utilizadas exclusivamente pela rede SqueezeNet), Dados Brutos do LiDAR, *Features* extraídas das Imagens, *Features* extraídas do LiDAR e *Features* Combinadas (fusão das *features* de imagem e LiDAR). Os valores apresentados para os *datasets* que não utilizam a SqueezeNet (ou seja, todos exceto "Imagens Brutas") representam a média de desempenho dos cinco modelos clássicos (k-Vizinhos mais próximos, Árvore de Decisão, Naive

Bayes, Rede Neural e Floresta Aleatória) para aquele tipo de dado específico dentro do cenário correspondente, conforme detalhado no Apêndice A. Para o tipo de dado "Imagens Brutas", os valores referem-se ao desempenho da SqueezeNet.

Tabela 7 – Acurácia Média por Cenário e Tipo de Dado.

Cenário	Tipo de Dado	Acurácia Média (%)
Robô Principal (Simulado)	Imagens Brutas (SqueezeNet)	100.00
	LiDAR bruto (Média Modelos)	97.97
	Features Imagem (Média Modelos)	99.93
	Features LiDAR (Média Modelos)	95.81
	Features Combinadas (Média Modelos)	99.90
Robô Principal (Real)	Imagens Brutas (SqueezeNet)	100.00
	LiDAR bruto (Média Modelos)	92.95
	Features Imagem (Média Modelos)	97.30
	Features LiDAR (Média Modelos)	95.58
	Features Combinadas (Média Modelos)	97.49
Robôs Secundários	Imagens Brutas (SqueezeNet)	97.67
	LiDAR bruto (Média Modelos)	51.15
	Features Imagem (Média Modelos)	70.30
	Features LiDAR (Média Modelos)	76.60
	Features Combinadas (Média Modelos)	75.43

Fonte: Autoria própria (2025), com base nos dados do Apêndice A.

Ao analisar os dados consolidados na Tabela 7, algumas tendências gerais são observadas. A rede SqueezeNet, de fato, apresenta as acuráncias mais elevadas em todos os cenários avaliados, incluindo os testes de portabilidade com as topologias secundárias. Mesmo quando os modelos demonstram uma queda brusca de desempenho neste último cenário, a SqueezeNet mantém uma acurácia robusta (97.67%), reforçando sua capacidade superior de generalização a partir de imagens brutas.

Considerando o robô principal, tanto no ambiente simulado quanto no real, observa-se que a origem dos dados (simulada ou real) e o tipo de dado utilizado para treinamento e validação (LiDAR bruto, features de imagem, features de LiDAR ou features combinadas) não resultaram em impactos dramaticamente negativos nas acuráncias finais dos modelos clássicos, que geralmente se mantiveram altas. Isso indica que, quando os dados de treinamento e validação provêm da mesma topologia robótica (ou de uma representação fiel, como no caso da simulação do robô principal), os modelos conseguem aprender padrões consistentes e realizar a classificação de forma satisfatória. Esses resultados sugerem que ambos os sensores, são capazes de fornecer informações úteis para a classificação dos objetos, tanto individualmente quanto em combinação.

Nos testes de portabilidade para as topologias secundárias (cenário "Robôs Secundários"), a maior queda de desempenho médio dos modelos clássicos ocorreu com os dados brutos do LiDAR, onde a acurácia média (51.15%) foi consideravelmente inferior à observada no cenário do robô principal (97.97% para simulado e 92.95% para real). Essa sensibilidade pode ser atribuída ao fato de que as leituras brutas do sensor LiDAR são altamente dependentes da posição e orientação relativas entre o sensor e o objeto. Pequenas variações na topografia do

robô podem alterar significativamente os ângulos e as distâncias registradas para um mesmo objeto. Utilizando o modelo de Árvore de Decisão como exemplo, que classifica com base em limiares para valores de *features* específicas (neste caso, uma distância em um determinado ângulo), uma mudança na posição do sensor pode fazer com que a *feature* crucial apareça em um ângulo diferente, confundindo o modelo e levando a erros de classificação.

Esperava-se que a extração de *features* dos dados do LiDAR pudesse mitigar esse problema, tornando a representação dos dados mais generalizável. De fato, observa-se um ganho significativo de acurácia ao comparar a média dos modelos clássicos com dados brutos do LiDAR (51.15%) com a média utilizando *features* extraídas do LiDAR (76.60%) no cenário multimodal dos robôs secundários, um aumento de aproximadamente 25%. No cenário do robô principal, a diferença entre usar dados brutos do LiDAR e *features* do LiDAR não foi tão pronunciada (cerca de 2% ou menos de diferença na acurácia média). Isso pode ser explicado pela coleta de dados altamente controlada para o robô principal, onde as variações de ângulo e posicionamento entre o sensor e os objetos foram minimizadas. Tal controle, no entanto, pode não refletir as condições operacionais reais, onde a orientação exata do sensor em relação aos objetos não pode ser garantida. Em suma, a extração de *features* demonstra ser particularmente importante para o sensor LiDAR em contextos de portabilidade, pois tende a normalizar as informações relevantes de forma mais independente da posição espacial exata do sensor e do objeto.

Analizando as *features* extraídas das imagens no cenário das topologias secundárias, nota-se que, em média, seu desempenho (70.30%) foi inferior ao das *features* do LiDAR (76.60%) e das *features* combinadas (75.43%). Isso sugere que as *features* de imagem extraídas neste trabalho, embora computacionalmente eficientes, podem não ter capturado as características dos objetos de forma tão robusta a variações de perspectiva quanto as *features* do LiDAR, o que não era inicialmente esperado, dado o volume de informação potencialmente maior nas imagens de profundidade. Isso indica que as *features* específicas extraídas das imagens podem ser menos generalizáveis para diferentes capturas ou topologias em comparação com as *features* derivadas do LiDAR.

5.3 Análise das Matrizes de Confusão

A análise das matrizes de confusão, apresentadas no Apêndice A, revela padrões específicos de acertos e erros dos modelos em diferentes cenários e com distintos tipos de dados, complementando a visão geral fornecida pelas métricas de acurácia.

No cenário do **robô principal com dados simulados** (Apêndice A.1.1), observa-se, de forma geral, uma baixa taxa de confusão entre as classes. No entanto, alguns erros pontuais ocorreram, predominantemente entre as classes "sinalizador" e "amortecedor", especialmente quando utilizados os dados brutos do LiDAR (conforme pode ser inferido observando as acurácia ligeiramente menores para alguns modelos com este tipo de dado). Embora não fosse um erro inicialmente esperado devido às distintas geometrias desses objetos, uma análise mais

detida dos dados de varredura do LiDAR sugere que, em certas angulações ou distâncias, o perfil inicial de um sinalizador (com diâmetro menor) pode apresentar uma assinatura similar a partes de um amortecedor, levando à confusão por parte dos modelos.

Ao transitar para o cenário do **robô principal com dados reais** (Apêndice A.1.2), utilizando os dados brutos do LiDAR, notou-se uma confusão recorrente entre as classes "isolador" e "amortecedor" em alguns modelos. Este comportamento não era totalmente antecipado, visto que os objetos possuem formatos distintos e posições diferentes no cabo (amortecedor abaixo, isolador acima). Uma possível explicação para este erro reside na proximidade entre os objetos relativa ao cabo (cerca de 10 a 20 centímetros), o que pode dificultar a distinção. Com as *features* extraídas das imagens neste cenário real, a confusão predominante deslocou-se para entre as classes "isolador" e "sinalizador". Isso pode ser explicado pelo fato de que, quando o sensor de profundidade está muito próximo de um isolador, a imagem capturada pode conter muitos elementos do próprio isolador, inflando numericamente as *features* baseadas em contagem de pixels de borda ou área, aproximando-as dos valores típicos de um sinalizador, que é volumetricamente maior. Para as *features* extraídas do LiDAR em ambiente real, as confusões foram similares às observadas no ambiente simulado, indicando uma certa consistência no comportamento do sensor e das *features* derivadas dele. Com o uso de *features* combinadas no cenário real, a taxa de erros tendeu a diminuir para a maioria dos modelos, com exceção da Rede Neural (MLP), que pareceu herdar e até amplificar as confusões presentes nos conjuntos de *features* individuais, resultando em uma classificação geral inferior.

Nos testes de portabilidade com as **topologias secundárias**, utilizando dados do **primeiro robô secundário** (Apêndice A.2.1), a análise das matrizes de confusão revela desafios significativos. Com dados brutos do LiDAR, alguns modelos tenderam a classificar a maioria das instâncias como "sinalizador" ou a agrupar as classes em pares confusos, como "nada" com "amortecedor" e "sinalizador" com "isolador". Este agrupamento faz sentido prático, pois são os pares que compartilham maiores similaridades morfológicas ou de posicionamento relativo. Quando utilizadas *features* extraídas (tanto de imagem quanto de LiDAR), a maior confusão persistiu entre "amortecedor" e "nada", com os modelos frequentemente incapazes de distinguilos, além de manterem confusões entre "sinalizador" e "isolador". Com as *features* combinadas, observou-se uma confusão quase total entre "amortecedor" e "nada" em alguns casos, mas uma leve redução na confusão entre "isolador" e "sinalizador" para certos modelos.

Para a **segunda topologia de robô secundário** (Apêndice A.2.2), com dados brutos do LiDAR, os modelos frequentemente confundiram "isolador" e "sinalizador". O modelo de Árvore de Decisão, por exemplo, também apresentou dificuldade em distinguir "nada" de "amortecedor". O Naive Bayes, neste cenário, tendeu a uma confusão mais generalizada, classificando grande parte dos dados como "sinalizador". Com as *features* extraídas das imagens, persistiu a confusão entre "nada" e "amortecedor", e entre "isolador" e "sinalizador", embora de forma menos acentuada que com os dados brutos do LiDAR. Para as *features* do LiDAR, o Naive Bayes confundiu "nada" e "amortecedor", enquanto outros modelos continuaram a ter dificul-

dade com "isolador" e "sinalizador". Finalmente, com *features* combinadas, o erro mais expressivo concentrou-se novamente entre as classes "amortecedor" e "nada".

6 CONCLUSÕES

REFERÊNCIAS

- BISHOP, C. M.; NASRABADI, N. M. **Pattern recognition and machine learning.** [S.I.]: Springer, 2006. v. 4.
- BRADSKI, G. The OpenCV Library. **Dr. Dobb's Journal of Software Tools**, „, 2000.
- CHENG, S. *et al.* Design and motion stability analysis of a straddle-type live working robot for power distribution lines. **Available at SSRN 5139507**, „, 2025.
- DOMINGUES, A. *et al.* A robotic cable-gripper for reliable inspection of transmission lines. In: MARQUES, L. *et al.* (Ed.). ROBOT 2023: SIXTH IBERIAN ROBOTICS CONFERENCE. 2024, Cham. **Anais [...]** Cham: Springer Nature Switzerland, 2024. p. 519–530. ISBN 978-3-031-58676-7.
- FAN, F. *et al.* Multi-robot cyber physical system for sensing environmental variables of transmission line. **Sensors**, MDPI v. 18, n. 9, p. 3146, 2018.
- FONSECA, A.; ABDO, R.; ALBERTO, J. Robot for inspection of transmission lines. In: IEEE. 2012 2ND INTERNATIONAL CONFERENCE ON APPLIED ROBOTICS FOR THE POWER INDUSTRY (CARPI). 2012. **Anais [...] [S.I.]**, 2012. p. 83–87.
- HARRIS, C. R. *et al.* Array programming with NumPy. **Nature**, v. 585,, p. 357–362, 2020.
- IANDOLA, F. N. *et al.* SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. **arXiv:1602.07360**, „, 2016.
- Intel Corporation. **realsense-ros – ROS Wrapper for Intel® RealSense™ Devices.** [S.I.]: , 2025. <https://github.com/IntelRealSense/realsense-ros/tree/ros1-legacy>. Acesso em: 20 maio 2025.
- JIANG, W. *et al.* Research on dual-arm coordination motion control strategy for power cable mobile robot. **Transactions of the Institute of Measurement and Control**, SAGE Publications Sage UK: London, England v. 41, n. 11, p. 3235–3247, 2019.
- LIMA, E. J.; BOMFIM, M. H. S.; MOURÃO, M. A. d. M. Polibot–power lines inspection robot. **Industrial Robot: An International Journal**, Emerald Publishing Limited v. 45, n. 1, p. 98–109, 2018.
- MCKINNEY, W. *et al.* Data structures for statistical computing in python. In: AUSTIN, TX. PROCEEDINGS OF THE 9TH PYTHON IN SCIENCE CONFERENCE. 445., 2010. **Anais [...] [S.I.]**, 2010. p. 51–56.
- MENDEZ-FLORES, E.; POURSHAHIDI, A.; EGERSTEDT, M. Raccoonbot: An autonomous wire-traversing solar-tracking robot for persistent environmental monitoring. **arXiv preprint arXiv:2501.14151**, „, 2025.
- MERKEL, D. Docker: lightweight linux containers for consistent development and deployment. **Linux journal**, v. 2014, n. 239, p. 2, 2014.
- Open Source Robotics Foundation. **rosserial – ROS protocol for embedded devices.** [S.I.]: , 2025. <https://wiki.ros.org/rosserial>. Acesso em: 20 maio 2025.
- PASZKE, A. *et al.* Pytorch: An imperative style, high-performance deep learning library. In: **Advances in Neural Information Processing Systems 32**. Curran Associates, Inc. 2019. p. 8024–8035. Disponível em: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.

- PEDREGOSA, F. *et al.* Scikit-learn: Machine learning in python. **Journal of machine learning research**, v. 12, n. Oct, p. 2825–2830, 2011.
- PETERS, J. F.; AHN, T.-C.; BORKOWSKI, M. Obstacle classification by a line-crawling robot: A rough neurocomputing approach. *In:* SPRINGER. INTERNATIONAL CONFERENCE ON ROUGH SETS AND CURRENT TRENDS IN COMPUTING. 2002. **Anais [...] [S.I.]**, 2002. p. 594–601.
- POULIOT, N.; RICHARD, P.-L.; MONTAMBAULT, S. Linescout power line robot: Characterization of a utm-30lx lidar system for obstacle detection. *In:* IEEE. 2012 IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS. 2012. **Anais [...] [S.I.]**, 2012. p. 4327–4334.
- Python Software Foundation. **Python: A programming language for general-purpose programming**. [S.I.]: , 2025. <https://www.python.org>. Versão 3.x, acesso em: 20 maio 2025.
- QIN, X. *et al.* Detecting inspection objects of power line from cable inspection robot lidar data. **Sensors**, MDPI v. 18, n. 4, p. 1284, 2018.
- QIN, X. *et al.* A novel method to reconstruct overhead high-voltage power lines using cable inspection robot lidar data. **Remote sensing**, MDPI v. 9, n. 7, p. 753, 2017.
- QING, Z. *et al.* Mechanical design and research of a novel power lines inspection robot. *In:* IEEE. 2016 INTERNATIONAL CONFERENCE ON INTEGRATED CIRCUITS AND MICROSYSTEMS (ICICM). 2016. **Anais [...] [S.I.]**, 2016. p. 363–366.
- QUIGLEY, M. *et al.* Ros: an open-source robot operating system. *In:* KOBE, JAPAN. ICRA WORKSHOP ON OPEN SOURCE SOFTWARE. 3 n. 3.2., 2009. **Anais [...] [S.I.]**, 2009. p. 5.
- RICHARD, P.-L.; POULIOT, N.; MONTAMBAULT, S. Introduction of a lidar-based obstacle detection system on the linescout power line robot. *In:* IEEE. 2014 IEEE/ASME INTERNATIONAL CONFERENCE ON ADVANCED INTELLIGENT MECHATRONICS. 2014. **Anais [...] [S.I.]**, 2014. p. 1734–1740.
- ROHMER, E.; SINGH, S. P.; FREESE, M. Vrep: A versatile and scalable robot simulation framework. *In:* IEEE. 2013 IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS. 2013. **Anais [...] [S.I.]**, 2013. p. 1321–1326.
- Slamtec. **rplidar_ros – ROS Driver for RPLIDAR Laser Scanners**. [S.I.]: , 2025. https://github.com/Slamtec/rplidar_ros. Acesso em: 20 maio 2025.
- SONG, Y. *et al.* A vision-based method for the broken spacer detection. *In:* IEEE. 2015 IEEE INTERNATIONAL CONFERENCE ON CYBER TECHNOLOGY IN AUTOMATION, CONTROL, AND INTELLIGENT SYSTEMS (CYBER). 2015. **Anais [...] [S.I.]**, 2015. p. 715–719.
- WANG, Y. *et al.* Systematic comparison of power line classification methods from als and mls point cloud data. **Remote Sensing**, MDPI v. 10, n. 8, p. 1222, 2018.
- XIAO, Q. *et al.* Campus security inspection robot based on yolo algorithm. *In:* PROCEEDINGS OF THE 2024 2ND INTERNATIONAL CONFERENCE ON FRONTIERS OF INTELLIGENT MANUFACTURING AND AUTOMATION. 2024. **Anais [...] [S.I.: s.n.]**, 2024. p. 800–806.
- YUE, X.; WANG, H.; JIANG, Y. A novel 110 kv power line inspection robot and its climbing ability analysis. **International Journal of Advanced Robotic Systems**, SAGE Publications Sage UK: London, England v. 14, n. 3, p. 1729881417710461, 2017.
- ZHU, M. *et al.* Target recognition of multi source machine vision pan tilt integrated inspection robot for power inspection. **IEEE Access**, IEEE,, 2024.

ZHU, Y.; WANG, X.; XU, B. Design of vision-based obstacle crossing of high-voltage line inspection robot. In: IEEE. 2016 IEEE INTERNATIONAL CONFERENCE ON CYBER TECHNOLOGY IN AUTOMATION, CONTROL, AND INTELLIGENT SYSTEMS (CYBER). 2016. **Anais** [...] [S.I.], 2016. p. 506–511.

APÊNDICE A – Resultados

Este apêndice consolida todos os resultados detalhados levantados durante a execução deste trabalho. Ele serve como um repositório completo dos dados obtidos a partir dos experimentos realizados com os modelos de aprendizado de máquina, os quais foram treinados sobre os diferentes conjuntos de dados gerados e descritos ao longo do texto principal.

Relembrando, os testes foram conduzidos com o objetivo de avaliar o desempenho da abordagem proposta em dois cenários distintos: (i) validação cruzada com os dados do robô principal, englobando tanto simulações quanto coletas reais, e (ii) testes de portabilidade em duas outras topologias robóticas simuladas, utilizados para verificar a robustez dos modelos e do sensor multimodal diante de variações físicas e estruturais.

A organização dos resultados apresentada a seguir mantém a estrutura originalmente planejada: os dados estão agrupados de acordo com a origem (topologia principal [simulado ou real] ou topologias secundárias). Dentro de cada grupo, são apresentados separadamente para cada tipo de *dataset* processado: imagens brutas da câmera de profundidade, dados brutos do *LiDAR*, *features* extraídas das imagens, *features* extraídas do *LiDAR*, e a combinação dessas *features*. Para cada um destes cenários, são exibidas tabelas com as métricas quantitativas de desempenho — acurácia média, perda final, tempo de validação e época final (quando aplicável) — para os cinco modelos de aprendizado de máquina avaliados. Matrizes de confusão também são apresentadas nos casos em que auxiliam na compreensão dos resultados, sobretudo quando o desempenho foi inferior ao esperado ou quando ocorreram confusões significativas entre as classes de interesse.

A.1 Resultados com o robô principal

Nesta seção são apresentados os resultados obtidos com os dados provenientes do robô principal, utilizado tanto em ambiente simulado quanto em testes reais em laboratório. Para ambas as origens, foi aplicada a técnica de validação cruzada, conforme descrito anteriormente. Os resultados estão organizados em subseções separadas, de modo a permitir uma análise clara do desempenho dos modelos em cada tipo de *dataset*. Cada conjunto de dados — imagens brutas, leituras do *LiDAR*, *features* extraídas e combinação de *features* — é avaliado individualmente, permitindo comparar o impacto de diferentes formas de representação dos dados na tarefa de classificação. As métricas consideradas incluem acurácia média, tempo de validação, perda final e número de épocas, além da matriz de confusão quando relevante. A distinção entre os resultados simulados e reais permite também avaliar a consistência dos modelos frente à variação entre os ambientes.

A.1.1 Dados simulados

Nesta subseção são apresentados os resultados obtidos a partir dos dados simulados gerados pelo robô principal durante as execuções no ambiente virtual.

A.1.1.1 Imagens brutas

Para avaliar as imagens foi utilizado a *SqueezeNet*. O *dataset* utilizado nesta etapa contém imagens de profundidade capturadas por simulação no robô principal, organizadas em quatro classes: amortecedor, isolador, sinalizador e nada. O treinamento foi conduzido utilizando validação cruzada com 5 dobras. A Tabela 8 apresenta os resultados obtidos, no final da tabela são mostradas as médias dos resultados.

Tabela 8 – Desempenho da SqueezeNet com imagens brutas (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	7	0.000308	100.0	0.6784
2	3	0.000024	100.0	0.6945
3	5	0.000005	100.0	0.6862
4	10	0.000450	100.0	0.6993
5	6	0.000076	100.0	0.6767
Média	6.2	0.000173	100.0	0.6870

Fonte: Autoria própria (2025).

A.1.1.2 Dados brutos do LiDAR

Esta subseção apresenta os resultados obtidos com os dados brutos de distância coletados pelo sensor *LiDAR*, simulados no robô principal.

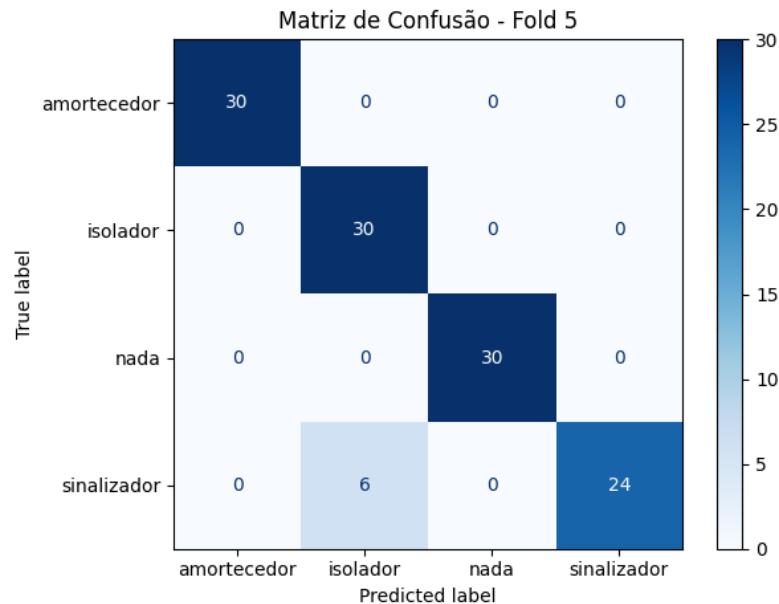
A.1.1.2.1 *k*-Vizinhos mais próximos

Tabela 9 – Desempenho do k-Vizinhos mais próximos com dados brutos do LiDAR (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	98.33	0.0034
2	-	-	96.67	0.0034
3	-	-	98.33	0.0033
4	-	-	98.33	0.0033
5	-	-	95.00	0.0034
Média	-	-	97.33	0.00336

Fonte: Autoria própria (2025).

Figura 16 – Matriz de confusão do k-Vizinhos mais próximos na dobra 5 com dados brutos do LiDAR (simulação).



Fonte: Autoria própria (2025).

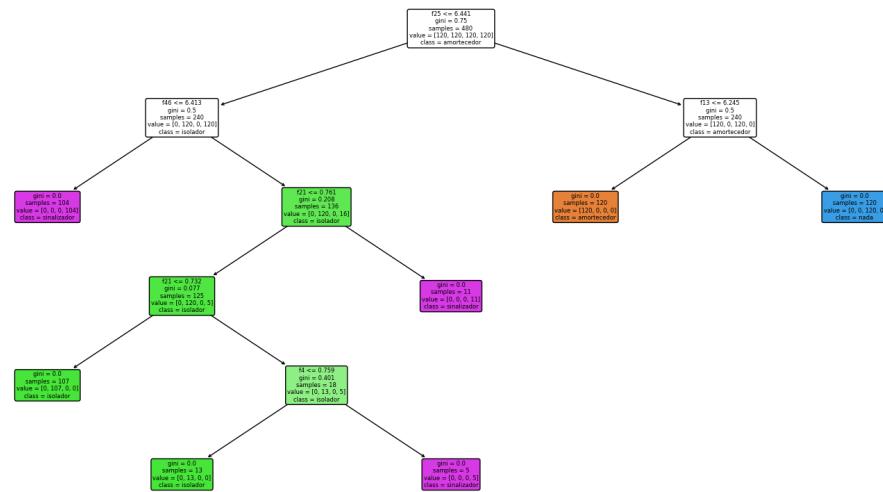
A.1.1.2.2 Árvore de Decisão

Tabela 10 – Desempenho da Árvore de Decisão com dados brutos do LiDAR (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100.00	0.0001
2	-	-	99.17	0.0001
3	-	-	99.17	0.0002
4	-	-	100.00	0.0002
5	-	-	99.17	0.0001
Média	-	-	99.50	0.0001

Fonte: Autoria própria (2025).

Figura 17 – Árvore de decisão gerada para a dobra 1 utilizando dados brutos do LiDAR (simulação).



Fonte: Autoria própria (2025).

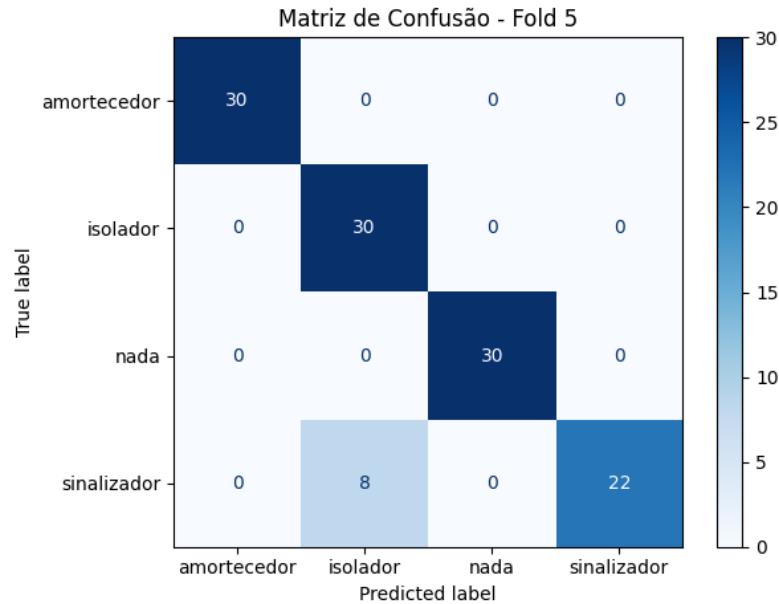
A.1.1.2.3 Naive Bayes

Tabela 11 – Desempenho do Naive Bayes com dados brutos do LiDAR (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	99.17	0.0003
2	-	-	97.50	0.0005
3	-	-	97.50	0.0003
4	-	-	97.50	0.0003
5	-	-	93.33	0.0003
Média	-	-	96.60	0.0003

Fonte: Autoria própria (2025).

Figura 18 – Matriz de confusão do Naive Bayes na dobra 5 com dados brutos do LiDAR (simulação).



Fonte: Autoria própria (2025).

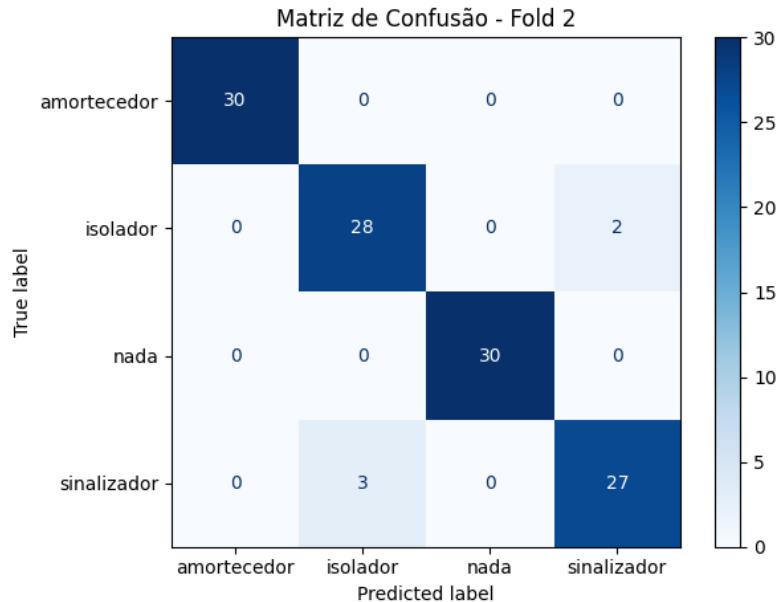
A.1.1.2.4 Rede Neural

Tabela 12 – Desempenho da Rede Neural com dados brutos do LiDAR (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	200	0.033461	96.67	0.0094
2	200	0.025889	95.83	0.0021
3	200	0.029690	98.33	0.0019
4	200	0.029371	97.50	0.0019
5	200	0.031941	93.33	0.0022
Média	200.0	0.030070	96.73	0.0035

Fonte: Autoria própria (2025).

Figura 19 – Matriz de confusão da Rede Neural na dobra 2 com dados brutos do LiDAR (simulação).



Fonte: Autoria própria (2025).

A.1.1.2.5 Floresta Aleatória

Tabela 13 – Desempenho da Floresta Aleatória com dados brutos do LiDAR (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100.0	0.0061
2	-	-	99.17	0.0061
3	-	-	99.17	0.0059
4	-	-	100.0	0.0059
5	-	-	100.0	0.0062
Média	-	-	99.7	0.0060

Fonte: Autoria própria (2025).

A.1.1.3 Features extraídas das imagens

Esta subseção apresenta os resultados obtidos com o uso de *features* extraídas das imagens de profundidade simuladas no robô principal. As imagens foram previamente processadas para extração de características numéricas representativas.

A.1.1.3.1 *k-Vizinhos mais próximos*

Tabela 14 – Desempenho do k-Vizinhos mais próximos com features extraídas das imagens (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100.0	0.0029
2	-	-	100.0	0.0054
3	-	-	100.0	0.0027
4	-	-	100.0	0.0028
5	-	-	100.0	0.0028
Média	-	-	100.0	0.0033

Fonte: Autoria própria (2025).

A.1.1.3.2 Árvore de Decisão

Tabela 15 – Desempenho da Árvore de Decisão com features extraídas das imagens (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100.0	0.000138
2	-	-	100.0	0.000123
3	-	-	100.0	0.000124
4	-	-	100.0	0.000124
5	-	-	100.0	0.000124
Média	-	-	100.0	0.000127

Fonte: Autoria própria (2025).

A.1.1.3.3 Naive Bayes

Tabela 16 – Desempenho do Naive Bayes com features extraídas das imagens (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100.0	0.000209
2	-	-	99.17	0.000212
3	-	-	100.0	0.000194
4	-	-	100.0	0.000211
5	-	-	100.0	0.000214
Média	-	-	99.83	0.000208

Fonte: Autoria própria (2025).

A.1.1.3.4 Rede Neural

Tabela 17 – Desempenho da Rede Neural com features extraídas das imagens (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	79	0.000395	100.0	0.0117
2	34	0.000362	100.0	0.0021
3	63	0.000620	100.0	0.0024
4	74	0.000415	99.17	0.0021
5	49	0.000421	100.0	0.0025
Média	59.8	0.000443	99.83	0.0042

Fonte: Autoria própria (2025).

A.1.1.3.5 Floresta Aleatória

Tabela 18 – Desempenho da Floresta Aleatória com features extraídas das imagens (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100.0	0.0055
2	-	-	100.0	0.0055
3	-	-	100.0	0.0053
4	-	-	100.0	0.0052
5	-	-	100.0	0.0052
Média	-	-	100.0	0.0054

Fonte: Autoria própria (2025).

A.1.1.4 Features extraídas do LiDAR

Esta subseção apresenta os resultados obtidos a partir de atributos derivados dos dados brutos de distância do sensor *LiDAR*, coletados em simulações com o robô principal. As leituras foram processadas para extrair informações relevantes (*features*) que pudessem melhorar a capacidade dos modelos de aprendizado de máquina.

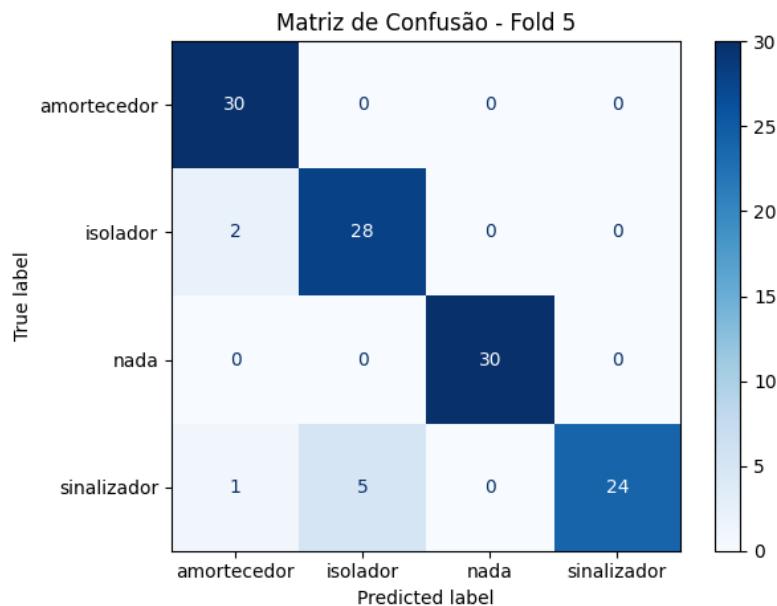
A.1.1.4.1 *k*-Vizinhos mais próximos

Tabela 19 – Desempenho do k-Vizinhos mais próximos com features extraídas do LiDAR (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	97.5	0.0027
2	-	-	95.0	0.0026
3	-	-	96.67	0.0030
4	-	-	96.67	0.0029
5	-	-	93.33	0.0028
Média	-	-	95.83	0.0028

Fonte: Autoria própria (2025).

Figura 20 – Matriz de confusão do k-Vizinhos mais próximos na dobra 5 com features extraídas do LiDAR (simulação).



Fonte: Autoria própria (2025).

A.1.1.4.2 Árvore de Decisão

Tabela 20 – Desempenho da Árvore de Decisão com features extraídas do LiDAR (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	98.33	0.0001
2	-	-	98.33	0.0002
3	-	-	99.17	0.0001
4	-	-	98.33	0.0001
5	-	-	96.67	0.0002
Média	-	-	98.17	0.0001

Fonte: Autoria própria (2025).

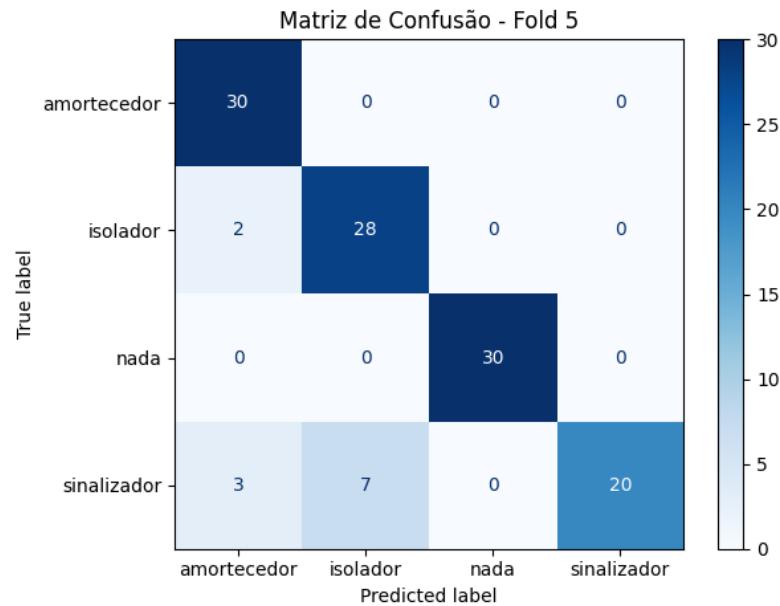
A.1.1.4.3 Naive Bayes

Tabela 21 – Desempenho do Naive Bayes com features extraídas do LiDAR (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	94.17	0.00022
2	-	-	95.00	0.00021
3	-	-	94.17	0.00034
4	-	-	95.00	0.00033
5	-	-	90.00	0.00020
Média	-	-	93.67	0.00026

Fonte: Autoria própria (2025).

Figura 21 – Matriz de confusão do Naive Bayes na dobra 5 com features extraídas do LiDAR (simulação).



Fonte: Autoria própria (2025).

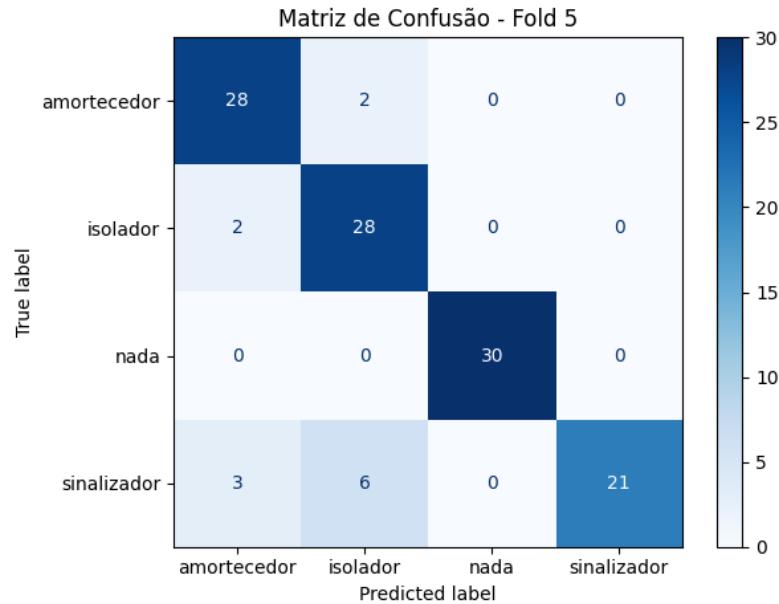
A.1.1.4.4 Rede Neural

Tabela 22 – Desempenho da Rede Neural com features extraídas do LiDAR (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	200	0.214619	95.00	0.00914
2	200	0.203636	94.17	0.00183
3	200	0.202239	93.33	0.00210
4	200	0.223546	95.83	0.00187
5	200	0.213401	89.17	0.00193
Média	200	0.211088	93.90	0.00377

Fonte: Autoria própria (2025).

Figura 22 – Matriz de confusão da Rede Neural na dobra 5 com features extraídas do LiDAR (simulação).



Fonte: Autoria própria (2025).

A.1.1.4.5 Floresta Aleatória

Tabela 23 – Desempenho da Floresta Aleatória com features extraídas do LiDAR (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	98.33	0.00586
2	-	-	95.00	0.00551
3	-	-	98.33	0.00582
4	-	-	98.33	0.00550
5	-	-	97.50	0.00572
Média	-	-	97.50	0.00566

Fonte: Autoria própria (2025).

A.1.1.5 Features combinadas

Esta subseção apresenta os resultados obtidos a partir da combinação das *features* extraídas das imagens e dos dados do sensor *LiDAR*, coletados em simulações com o robô principal.

A.1.1.5.1 *k-Vizinhos mais próximos*

Tabela 24 – Desempenho do k-Vizinhos mais próximos com features combinadas (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100.0	0.0027
2	-	-	100.0	0.0028
3	-	-	100.0	0.0029
4	-	-	100.0	0.0027
5	-	-	100.0	0.0030
Média	1	-	100.0	0.00282

Fonte: Autoria própria (2025).

A.1.1.5.2 Árvore de Decisão

Tabela 25 – Desempenho da Árvore de Decisão com features combinadas (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100.0	0.000137
2	-	-	100.0	0.000128
3	-	-	100.0	0.000121
4	-	-	100.0	0.000174
5	-	-	100.0	0.000122
Média	1	-	100.0	0.000136

Fonte: Autoria própria (2025).

A.1.1.5.3 Naive Bayes

Tabela 26 – Desempenho do Naive Bayes com features combinadas (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100.0	0.000205
2	-	-	99.17	0.000200
3	-	-	100.0	0.000215
4	-	-	100.0	0.000197
5	-	-	100.0	0.000232
Média	1	-	99.83	0.000210

Fonte: Autoria própria (2025).

A.1.1.5.4 *Rede Neural*

Tabela 27 – Desempenho da Rede Neural com features combinadas (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	49	0.000374	100.0	0.0096
2	66	0.000453	99.17	0.0022
3	98	0.000035	100.0	0.0023
4	75	0.000934	100.0	0.0019
5	38	0.000445	100.0	0.0020
Média	65.2	0.000428	99.67	0.0036

Fonte: Autoria própria (2025).

A.1.1.5.5 *Floresta Aleatória*

Tabela 28 – Desempenho da Floresta Aleatória com features combinadas (simulação)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100.0	0.0057
2	-	-	100.0	0.0053
3	-	-	100.0	0.0057
4	-	-	100.0	0.0058
5	-	-	100.0	0.0054
Média	-	-	100.0	0.0056

Fonte: Autoria própria (2025).

A.1.2 Dados reais

Nesta subseção são apresentados os resultados obtidos a partir dos dados reais gerados pelo robô principal durante as execuções no ambiente do laboratório.

A.1.2.1 Imagens brutas

Esta subseção apresenta os resultados obtidos com as imagens de profundidade brutas capturadas pelo robô principal em ambiente real no laboratório.

Tabela 29 – Desempenho da SqueezeNet com imagens brutas (real)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	10	0.000032	100.0	0.7084
2	11	0.000280	100.0	0.6005
3	9	0.000295	100.0	0.6091
4	2	0.000004	100.0	0.6110
5	4	0.000735	100.0	0.6307
Média	7.2	0.000269	100.0	0.6319

Fonte: Autoria própria (2025).

A.1.2.2 Dados brutos do LiDAR

Esta subseção apresenta os resultados obtidos diretamente dos dados brutos de distância fornecidos pelo sensor *LiDAR*, coletados em ambientes reais com o robô principal.

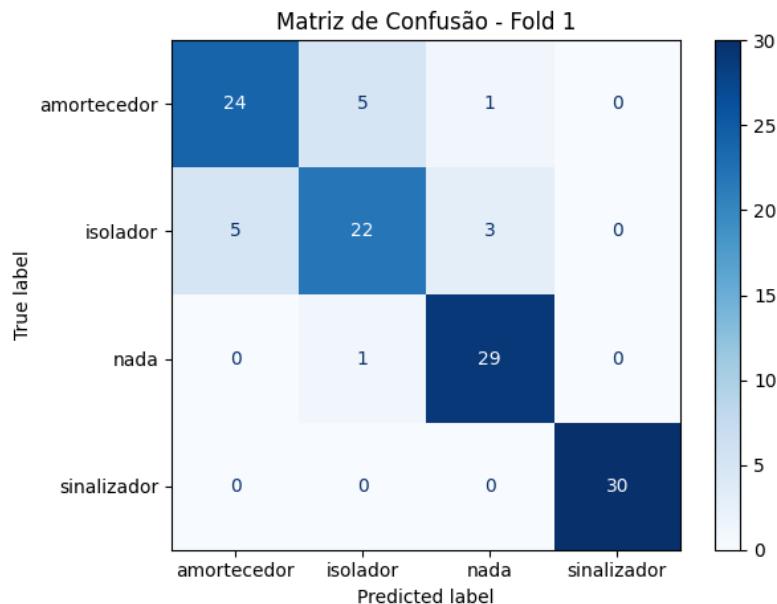
A.1.2.2.1 *k*-Vizinhos mais próximos

Tabela 30 – Desempenho do k-Vizinhos mais próximos com dados brutos do LiDAR (real)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	87.50	0.0039
2	-	-	93.33	0.0037
3	-	-	93.33	0.0031
4	-	-	91.67	0.0032
5	-	-	86.67	0.0034
Média	-	-	90.90	0.00346

Fonte: Autoria própria (2025).

Figura 23 – Matriz de confusão do k-Vizinhos mais próximos na dobraria 1 com dados brutos do LiDAR (real).



Fonte: Autoria própria (2025).

A.1.2.2.2 Árvore de Decisão

Tabela 31 – Desempenho do Árvore de Decisão com dados brutos do LiDAR (real)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100.00	0.000144
2	-	-	99.17	0.000139
3	-	-	100.00	0.000136
4	-	-	99.17	0.000215
5	-	-	97.50	0.000159
Média	-	-	99.37	0.000159

Fonte: Autoria própria (2025).

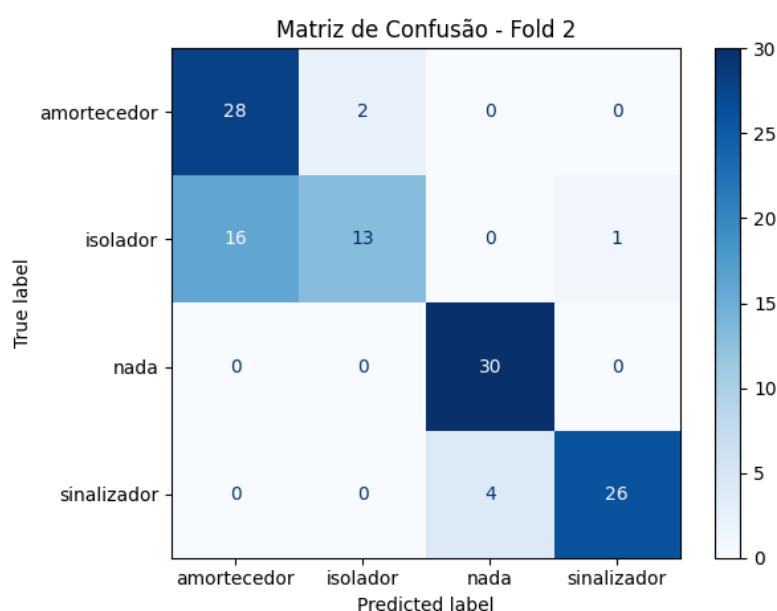
A.1.2.2.3 Naive Bayes

Tabela 32 – Desempenho do Naive Bayes com dados brutos do LiDAR (real)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	86.67	0.000273
2	-	-	80.83	0.000262
3	-	-	81.67	0.000290
4	-	-	80.00	0.000286
5	-	-	77.50	0.000270
Média	-	-	81.33	0.000276

Fonte: Autoria própria (2025).

Figura 24 – Matriz de confusão do Naive Bayes na dobra 2 com dados brutos do LiDAR (real).



Fonte: Autoria própria (2025).

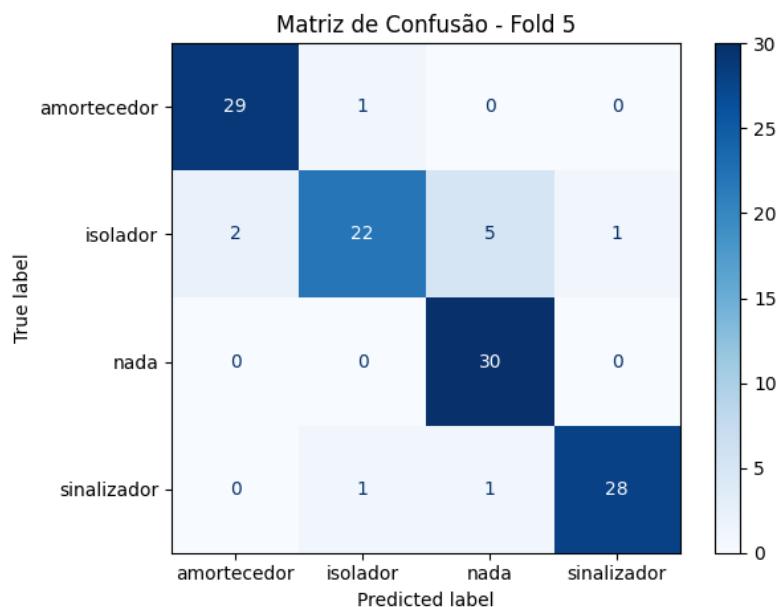
A.1.2.2.4 Rede Neural

Tabela 33 – Desempenho da Rede Neural com dados brutos do LiDAR (real)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	200	0.003282	93.33	0.010296
2	200	0.004890	94.17	0.003278
3	200	0.004671	93.33	0.001895
4	160	0.000999	94.17	0.001972
5	200	0.004385	90.83	0.001836
Média	192	0.003645	93.17	0.003855

Fonte: Autoria própria (2025).

Figura 25 – Matriz de confusão da Rede Neural na dobra 5 com dados brutos do LiDAR (real).



Fonte: Autoria própria (2025).

A.1.2.2.5 Floresta Aleatória

Tabela 34 – Desempenho da Floresta Aleatória com dados brutos do LiDAR (real)

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100.0	0.005681
2	-	-	100.0	0.005755
3	-	-	100.0	0.005916
4	-	-	100.0	0.006184
5	-	-	100.0	0.006251
Média	-	-	100.0	0.005957

Fonte: Autoria própria (2025).

A.1.2.3 Features extraídas das imagens

Esta subseção apresenta os resultados obtidos a partir de atributos derivados das imagens reais coletadas pelo robô principal. As imagens foram processadas para extrair informações relevantes (*features*) que pudessem melhorar a capacidade dos modelos de aprendizado de máquina na identificação das classes de objetos.

A.1.2.3.1 *k*-Vizinhos mais próximos

Tabela 35 – Desempenho do k-Vizinhos mais próximos com features extraídas das imagens (real).

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100,0	0,0030
2	-	-	100,0	0,0030
3	-	-	100,0	0,0029
4	-	-	100,0	0,0030
5	-	-	100,0	0,0031
Média	-	-	100,0	0,0030

Fonte: Autoria própria (2025).

A.1.2.3.2 Árvore de Decisão

Tabela 36 – Desempenho da Árvore de Decisão com features extraídas das imagens (real).

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100,0	0,000154
2	-	-	100,0	0,000129
3	-	-	100,0	0,000138
4	-	-	100,0	0,000135
5	-	-	99,17	0,000128
Média	-	-	99,67	0,000137

Fonte: Autoria própria (2025).

A.1.2.3.3 Naive Bayes

Tabela 37 – Desempenho do Naive Bayes com features extraídas das imagens (real).

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	99,17	0,000196
2	-	-	100,0	0,000192
3	-	-	99,17	0,000350
4	-	-	100,0	0,000243
5	-	-	98,33	0,000200
Média	-	-	99,33	0,000236

Fonte: Autoria própria (2025).

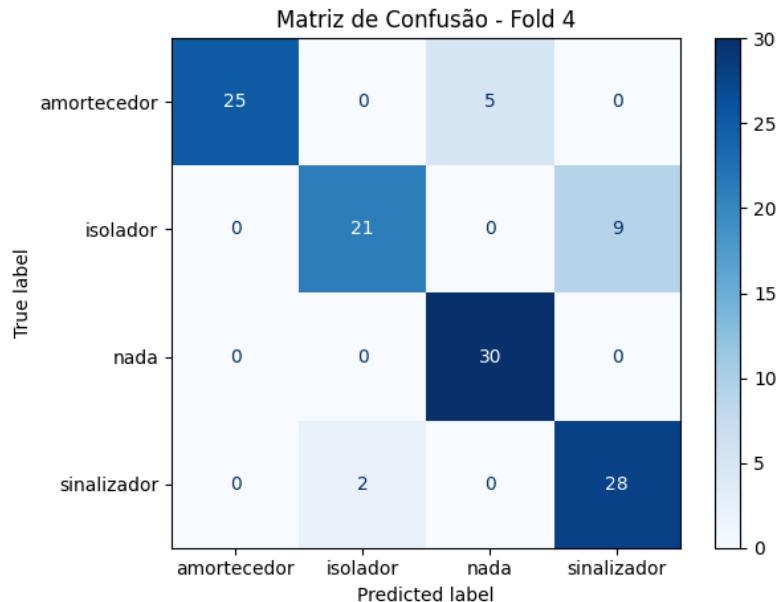
A.1.2.3.4 Rede Neural

Tabela 38 – Desempenho da Rede Neural com features extraídas das imagens (real).

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	200	7,458555	89,17	0,009501
2	200	1,502896	87,5	0,001873
3	200	3,814955	89,17	0,001903
4	200	10,270095	86,67	0,001998
5	200	4,025579	86,67	0,001852
Média	200	5,014204	87,84	0,003025

Fonte: Autoria própria (2025).

Figura 26 – Matriz de confusão da Rede Neural na dobraria 4 com features extraídas das imagens (real).



Fonte: Autoria própria (2025).

A.1.2.3.5 Floresta Aleatória

Tabela 39 – Desempenho da Floresta Aleatória com features extraídas das imagens (real).

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	99,17	0,005706
2	-	-	100,0	0,005852
3	-	-	100,0	0,005603
4	-	-	100,0	0,005454
5	-	-	100,0	0,005498
Média	-	-	99,67	0,005623

Fonte: Autoria própria (2025).

A.1.2.4 Features extraídas do LiDAR

Esta subseção apresenta os resultados obtidos a partir de atributos derivados dos dados brutos reais de distância do sensor *LiDAR*, coletados com o robô principal. As leituras foram processadas para extrair informações relevantes (*features*) que pudessem melhorar a capacidade dos modelos de aprendizado de máquina.

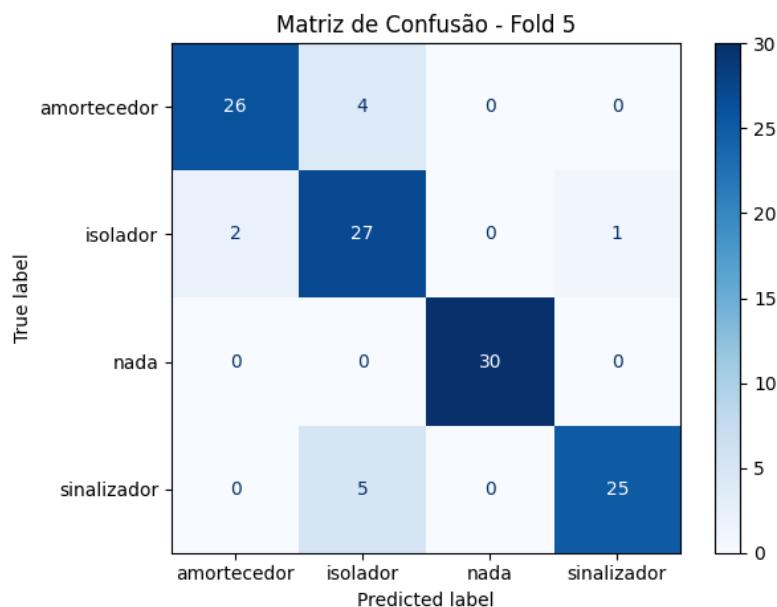
A.1.2.4.1 *k*-Vizinhos mais próximos

Tabela 40 – Desempenho do k-Vizinhos mais próximos com features extraídas do LiDAR (real).

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	91,67	0,0030
2	-	-	94,17	0,0027
3	-	-	94,17	0,0028
4	-	-	95,83	0,0028
5	-	-	90,00	0,0027
Média	-	-	93,17	0,0028

Fonte: Autoria própria (2025).

Figura 27 – Matriz de confusão do k-Vizinhos mais próximos na dobraria 5 com features extraídas do LiDAR (real).



Fonte: Autoria própria (2025).

A.1.2.4.2 Árvore de Decisão

Tabela 41 – Desempenho da Árvore de Decisão com features extraídas do LiDAR (real).

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	97,50	0,000133
2	-	-	99,17	0,000135
3	-	-	97,50	0,000124
4	-	-	98,33	0,000137
5	-	-	95,00	0,000138
Média	-	-	97,50	0,000134

Fonte: Autoria própria (2025).

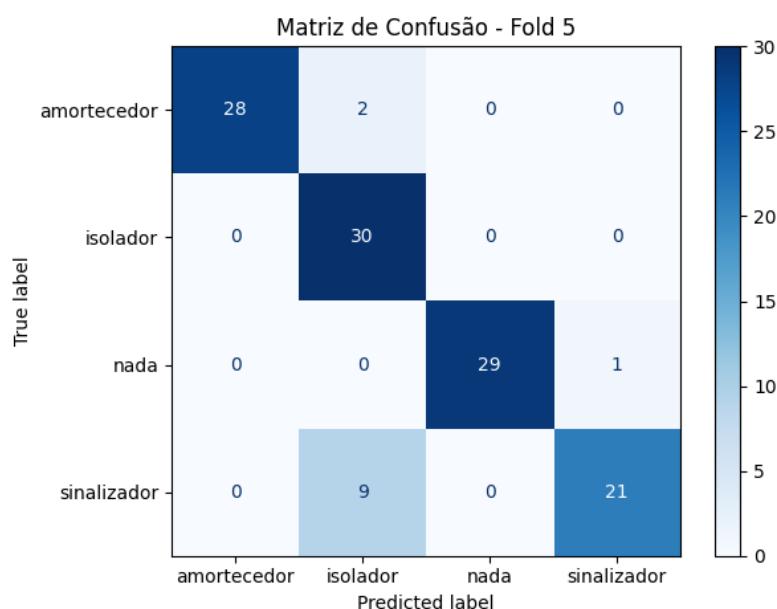
A.1.2.4.3 Naive Bayes

Tabela 42 – Desempenho do Naive Bayes com features extraídas do LiDAR (real).

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	96,67	0,00026
2	-	-	94,17	0,000252
3	-	-	93,33	0,000207
4	-	-	93,33	0,000205
5	-	-	90,00	0,000295
Média	-	-	93,90	0,000244

Fonte: Autoria própria (2025).

Figura 28 – Matriz de confusão do Naive Bayes na dobra 5 com features extraídas do LiDAR (real).



Fonte: Autoria própria (2025).

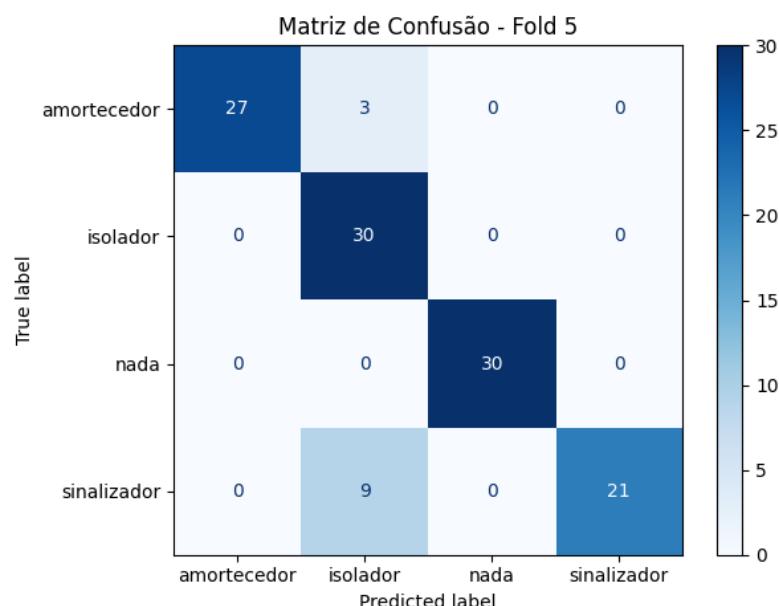
A.1.2.4.4 Rede Neural

Tabela 43 – Desempenho da Rede Neural com features extraídas do LiDAR (real).

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	200	0.135437	96.67	0.010137
2	200	0.121473	95.83	0.002069
3	200	0.143646	95.83	0.002144
4	200	0.109989	96.67	0.002189
5	200	0.168874	90.00	0.001935
Média			95.00	0.003295

Fonte: Autoria própria (2025).

Figura 29 – Matriz de confusão da Rede Neural na dobra 5 com features extraídas do LiDAR (real).



Fonte: Autoria própria (2025).

A.1.2.4.5 Floresta Aleatória

Tabela 44 – Desempenho da Floresta Aleatória com features extraídas do LiDAR (real).

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100.0	0.005577
2	-	-	98.33	0.005834
3	-	-	99.17	0.006009
4	-	-	96.67	0.005491
5	-	-	97.5	0.005741
Média			98.33	0.005730

Fonte: Autoria própria (2025).

A.1.2.5 Features combinadas

Esta subseção apresenta os resultados obtidos a partir da combinação das *features* extraídas das imagens e dos dados do sensor *LiDAR* reais, coletados com o robô principal.

A.1.2.5.1 *k*-Vizinhos mais próximos

Tabela 45 – Desempenho do k-Vizinhos mais próximos com features combinadas (real).

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100.0	0.0028
2	-	-	100.0	0.0028
3	-	-	100.0	0.0029
4	-	-	100.0	0.0028
5	-	-	100.0	0.0031
Média			100.0	0.0029

Fonte: Autoria própria (2025).

A.1.2.5.2 Árvore de Decisão

Tabela 46 – Desempenho do Naive Bayes com features combinadas (real).

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100.0	0.000131
2	-	-	100.0	0.000139
3	-	-	100.0	0.000138
4	-	-	98.33	0.000134
5	-	-	99.17	0.000134
Média			99.5	0.000134

Fonte: Autoria própria (2025).

A.1.2.5.3 Naive Bayes

Tabela 47 – Desempenho do Naive Bayes com features combinadas (real).

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	100.0	0.000224
2	-	-	100.0	0.000198
3	-	-	100.0	0.000210
4	-	-	100.0	0.000235
5	-	-	98.33	0.000205
Média			99.67	0.000214

Fonte: Autoria própria (2025).

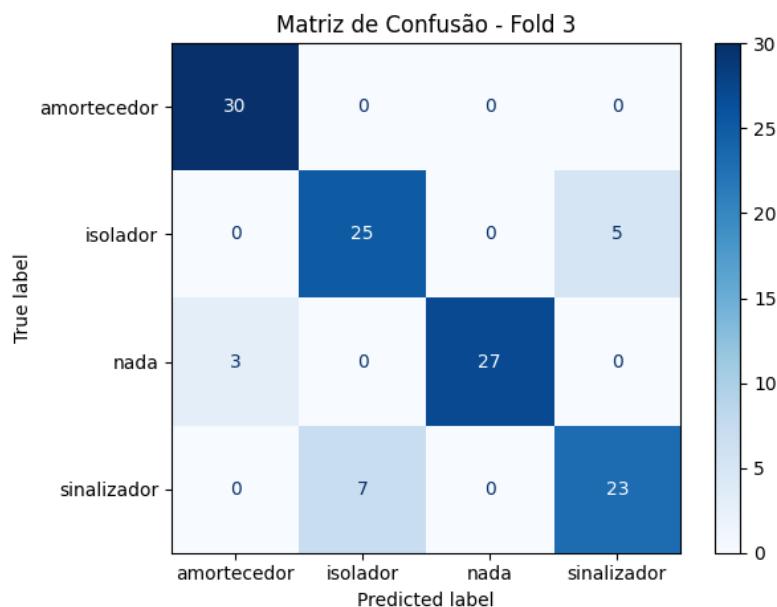
A.1.2.5.4 Rede Neural

Tabela 48 – Desempenho da Rede Neural com features combinadas (real).

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	200	1.0448	94.17	0.009344
2	200	1.9525	87.50	0.001929
3	200	3.6604	87.50	0.001974
4	200	3.4093	95.00	0.002141
5	200	6.8385	90.83	0.001893
Média		90.60	0.003456	

Fonte: Autoria própria (2025).

Figura 30 – Matriz de confusão da dobra 3 para a Rede Neural com features combinadas (real).



Fonte: Autoria própria (2025).

A.1.2.5.5 Floresta Aleatória

Tabela 49 – Desempenho da Floresta Aleatória com features combinadas (real).

Dobra	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
1	-	-	99.17	0.006221
2	-	-	100.0	0.005471
3	-	-	100.0	0.005714
4	-	-	100.0	0.005598
5	-	-	100.0	0.005813
Média		99.67	0.005763	

Fonte: Autoria própria (2025).

A.2 Resultados com os robôs secundários

Nesta seção são apresentados os resultados obtidos com os dados provenientes do robôs secundários, utilizados em ambiente simulado. Para ambas topologias analisadas, foi aplicada a validação usando os dados da topologia principal para treinamento e os dados das topologias secundárias para validação, conforme descrito anteriormente. Os resultados estão organizados em subseções separadas, de modo a permitir uma análise clara do desempenho dos modelos em cada tipo de *dataset*. Cada conjunto de dados — imagens brutas, leituras do *LiDAR*, *features* extraídas e combinação de *features* — é avaliado individualmente. As métricas consideradas incluem acurácia média, tempo de validação, perda final e número de épocas, além da matriz de confusão quando relevante. A distinção entre as topologias permite também avaliar a consistência do sensor multimodal frente à variação entre os ambientes.

A.2.1 Dados do primeiro robô

Nesta subseção são apresentados os resultados obtidos a partir dos dados simulados gerados pela primeira topologia secundária em conjunto com os dados do robô principal usados para treinamento.

A.2.1.1 Imagens brutas

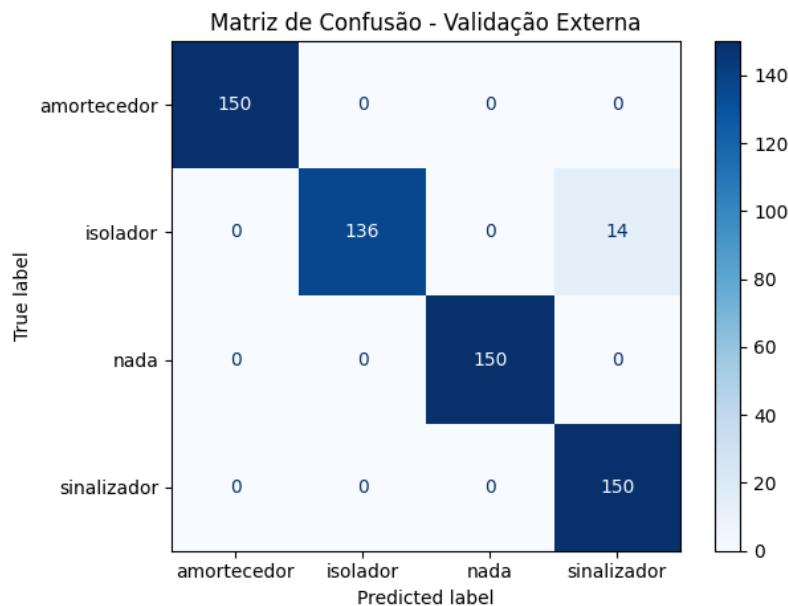
Esta subseção apresenta os resultados obtidos com as imagens de profundidade brutas capturadas pelo primeiro robô das topologias secundárias em ambiente simulado com o modelo treinado pelos dados do robô principal.

Tabela 50 – Desempenho da SqueezeNet com imagens brutas (robô 1 das topologias secundárias).

Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
3	0.00025	97.67	3.371939

Fonte: Autoria própria (2025).

Figura 31 – Matriz de confusão do modelo SqueezeNet utilizando imagens brutas (primeiro robô das topologias secundárias).



Fonte: Autoria própria (2025).

A.2.1.2 Dados brutos do LiDAR

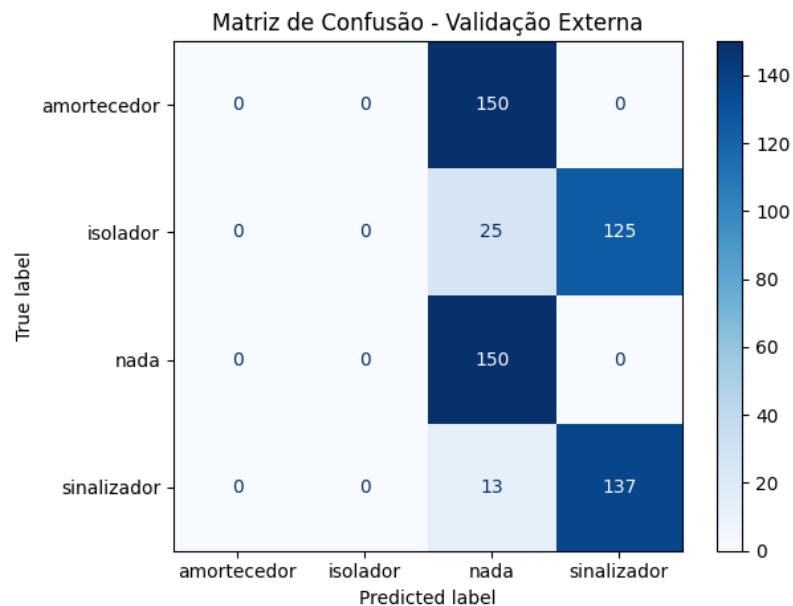
Esta subseção apresenta os resultados obtidos com os dados brutos do LiDAR capturadas pelo primeiro robô das topologias secundárias em ambiente simulado com o modelo treinado pelos dados do robô principal.

Tabela 51 – Comparativo de desempenho entre diferentes modelos.

Modelo	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
kNN	-	-	47.83	0.01927
Árvore	-	-	46.00	0.000213
Naive	-	-	25.00	0.000745
Rede	-	-	25.00	0.000745
Floresta	-	-	46.67	0.007682
Média			38.10	0.056262

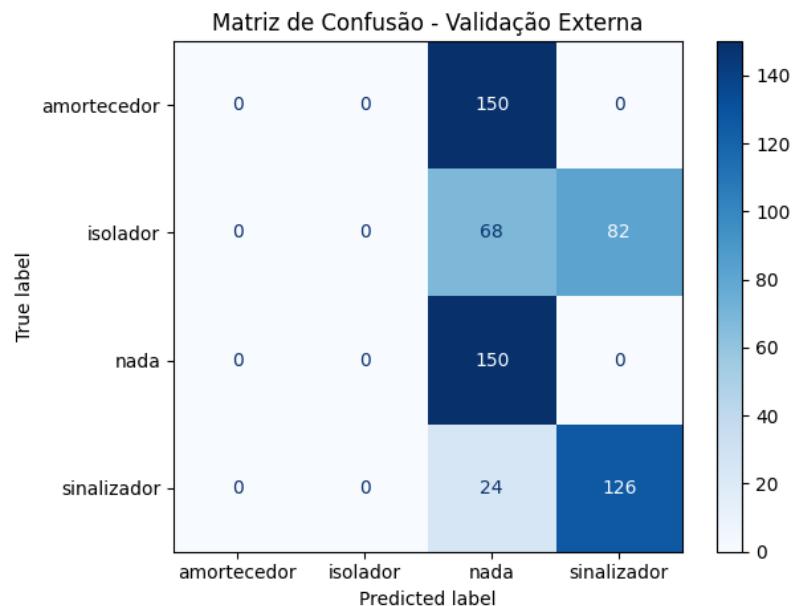
Fonte: Autoria própria (2025).

Figura 32 – Matriz de confusão para o k-Vizinhos mais próximos com dados brutos do LiDAR (primeiro robô das topologias secundárias).



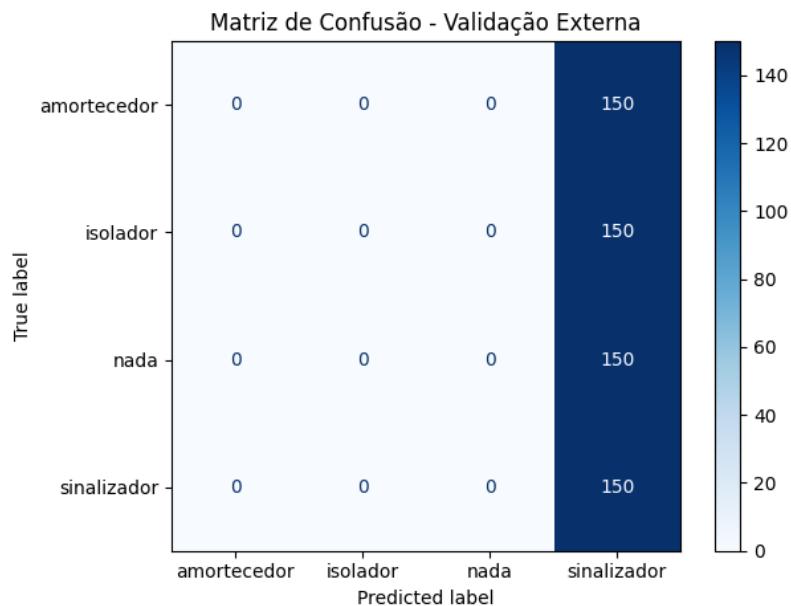
Fonte: Autoria própria (2025).

Figura 33 – Matriz de confusão para a Árvore de Decisão com dados brutos do LiDAR (primeiro robô das topologias secundárias).



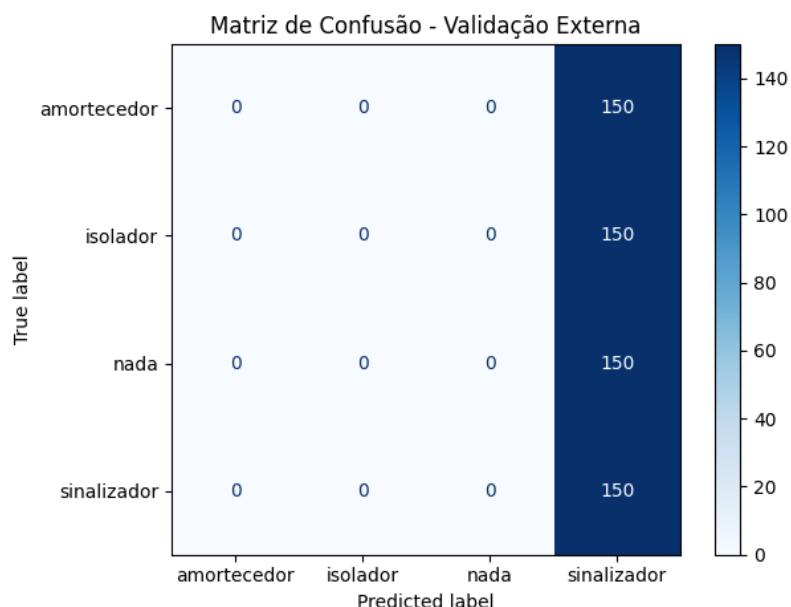
Fonte: Autoria própria (2025).

Figura 34 – Matriz de confusão para o Naive Bayes com dados brutos do LiDAR (primeiro robô das topologias secundárias).



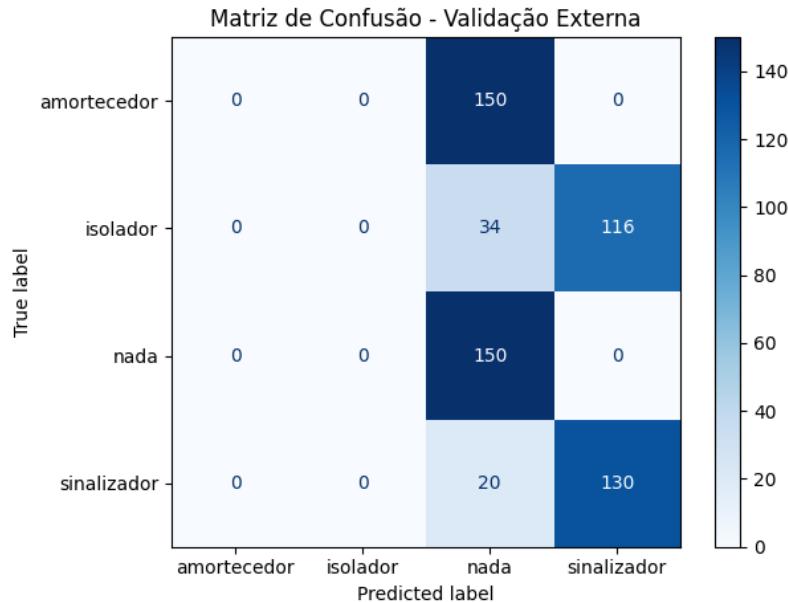
Fonte: Autoria própria (2025).

Figura 35 – Matriz de confusão para a Rede Neural com dados brutos do LiDAR (primeiro robô das topologias secundárias).



Fonte: Autoria própria (2025).

Figura 36 – Matriz de confusão para a Floresta Aleatória com dados brutos do LiDAR (primeiro robô das topologias secundárias).



Fonte: Autoria própria (2025).

A.2.1.3 Features extraídas das imagens

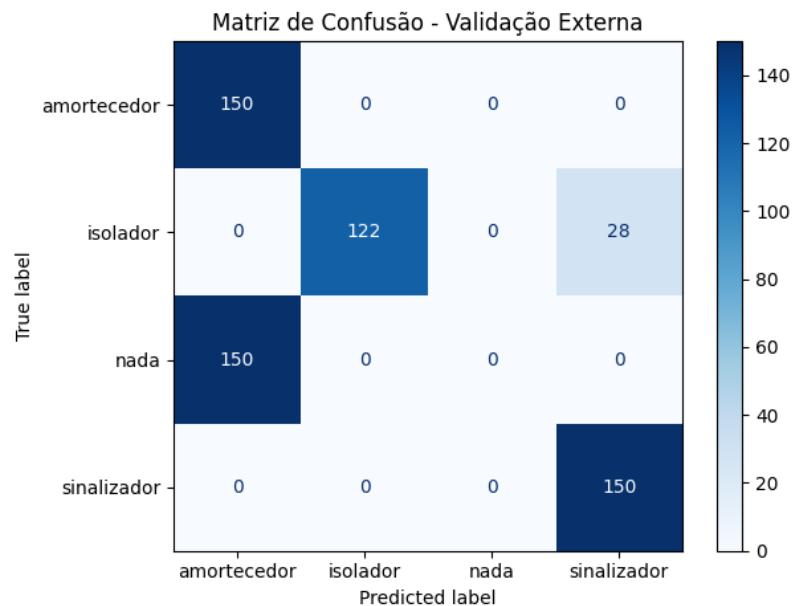
Esta subseção apresenta os resultados obtidos a partir de atributos derivados das imagens capturadas pelo primeiro robô das topologias secundárias em ambiente simulado com o modelo treinado pelos dados do robô principal. As imagens foram processadas para extrair informações relevantes (*features*) que pudessem melhorar a capacidade dos modelos de aprendizado de máquina na identificação das classes de objetos.

Tabela 52 – Desempenho dos modelos com features extraídas das imagens (primeiro robô das topologias secundárias).

Modelo	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
kNN	-	-	70.33	0.012015
Árvore	-	-	69.67	0.000170
Naive	-	-	69.83	0.000264
Rede	43	0.000551	75.00	0.015986
Floresta	-	-	69.67	0.007539
Média			70.90	0.007195

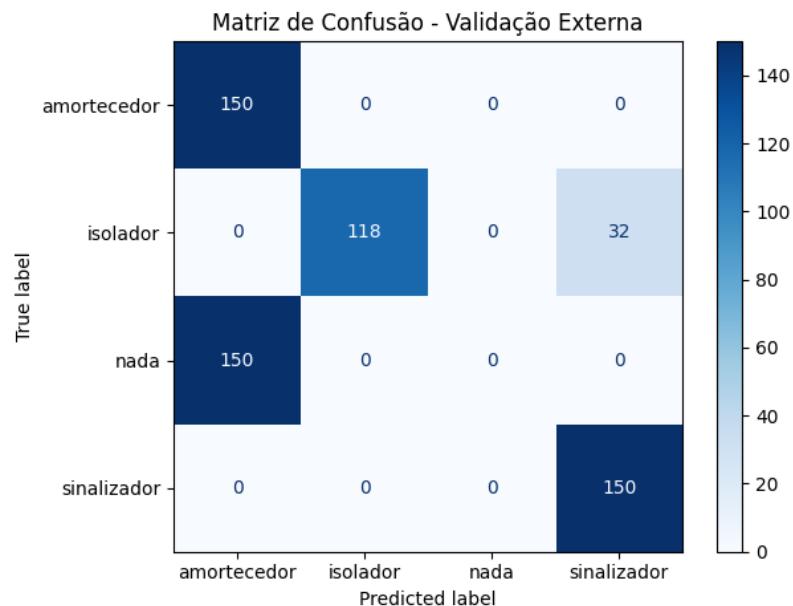
Fonte: Autoria própria (2025).

Figura 37 – Matriz de confusão para o k-Vizinhos mais próximos com features extraídas das imagens (primeiro robô das topologias secundárias).



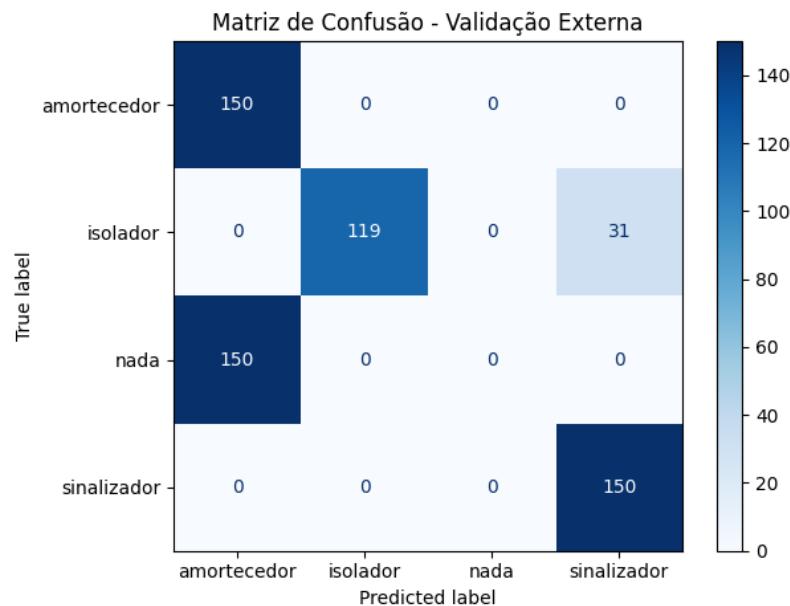
Fonte: Autoria própria (2025).

Figura 38 – Matriz de confusão para a Árvore de Decisão com features extraídas das imagens (primeiro robô das topologias secundárias).



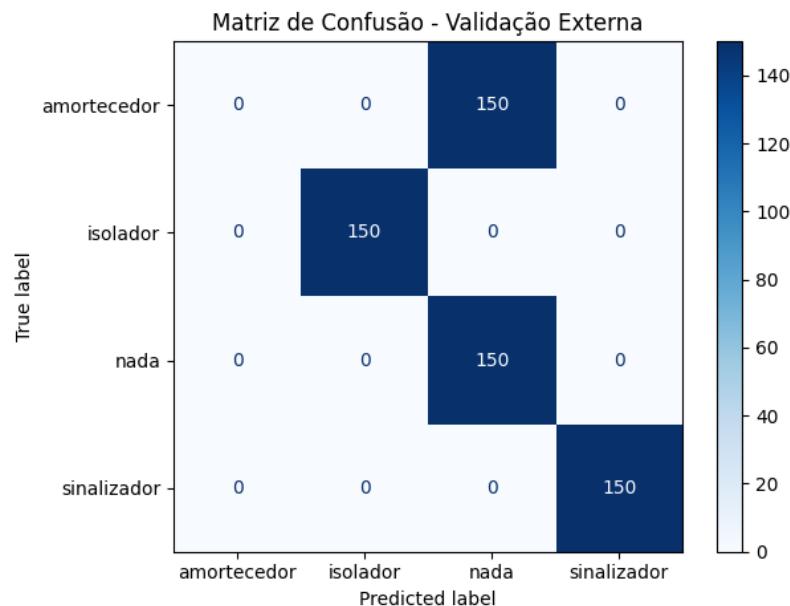
Fonte: Autoria própria (2025).

Figura 39 – Matriz de confusão para o Naive Bayes com features extraídas das imagens (primeiro robô das topologias secundárias).



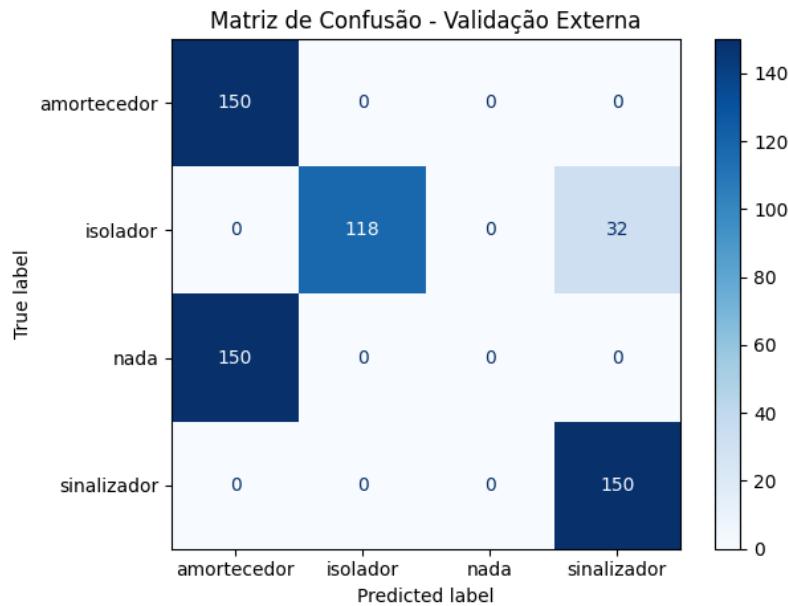
Fonte: Autoria própria (2025).

Figura 40 – Matriz de confusão para a Rede Neural com features extraídas das imagens (primeiro robô das topologias secundárias).



Fonte: Autoria própria (2025).

Figura 41 – Matriz de confusão para a Floresta Aleatória com features extraídas das imagens (primeiro robô das topologias secundárias).



Fonte: Autoria própria (2025).

A.2.1.4 Features extraídas do LiDAR

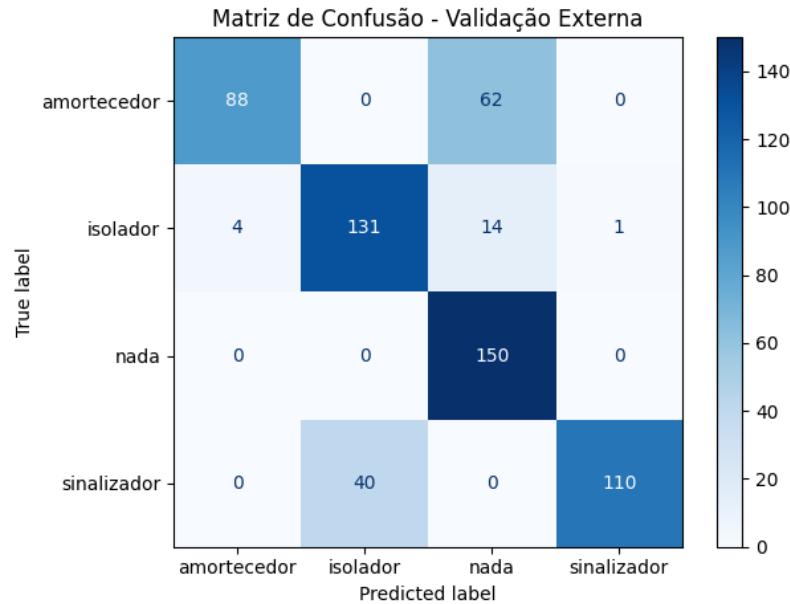
Esta subseção apresenta os resultados obtidos a partir de atributos derivados dos dados de distância do sensor *LiDAR*, capturadas pelo primeiro robô das topologias secundárias em ambiente simulado com o modelo treinado pelos dados do robô principal. As imagens foram processadas para extrair informações relevantes (*features*) que pudesse melhorar a capacidade dos modelos de aprendizado de máquina na identificação das classes de objetos.

Tabela 53 – Desempenho dos modelos com features extraídas do LiDAR (primeiro robô das topologias secundárias).

Modelo	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
kNN	-	-	79.83	0.012294
Árvore	-	-	79.00	0.000156
Naive	-	-	63.00	0.000232
Rede	200	0.205509	69.17	0.015490
Floresta	-	-	80.33	0.008353
Média			74.27	0.007305

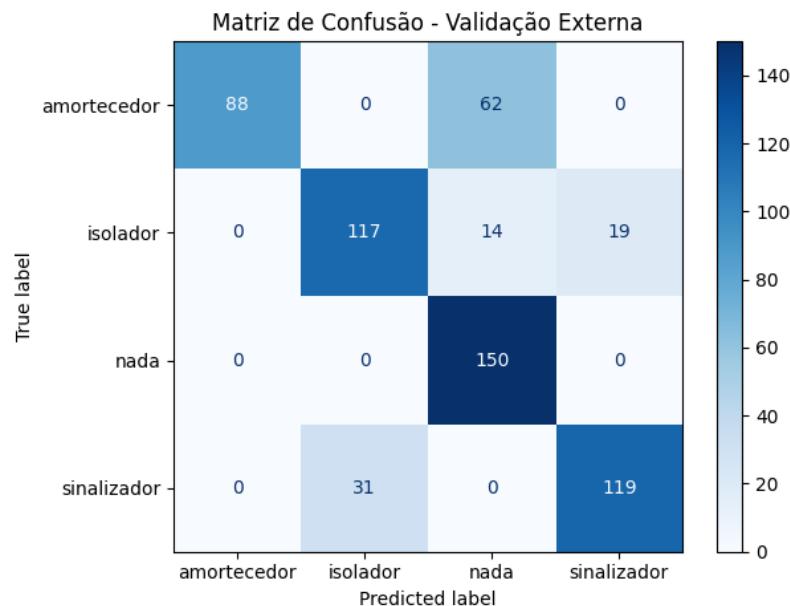
Fonte: Autoria própria (2025).

Figura 42 – Matriz de confusão para o k-Vizinhos mais próximos com features extraídas do LiDAR (primeiro robô das topologias secundárias).



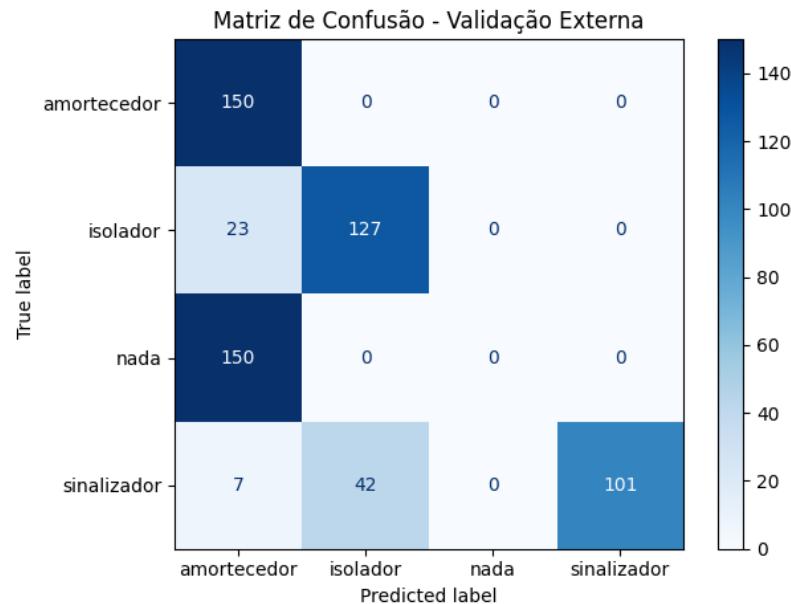
Fonte: Autoria própria (2025).

Figura 43 – Matriz de confusão para a Árvore de Decisão com features extraídas do LiDAR (primeiro robô das topologias secundárias).



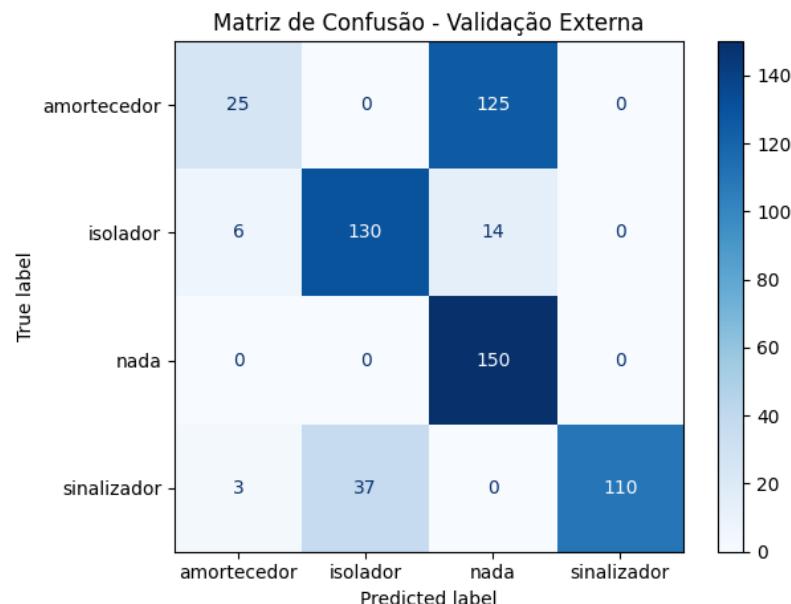
Fonte: Autoria própria (2025).

Figura 44 – Matriz de confusão para o Naive Bayes com features extraídas do LiDAR (primeiro robô das topologias secundárias).



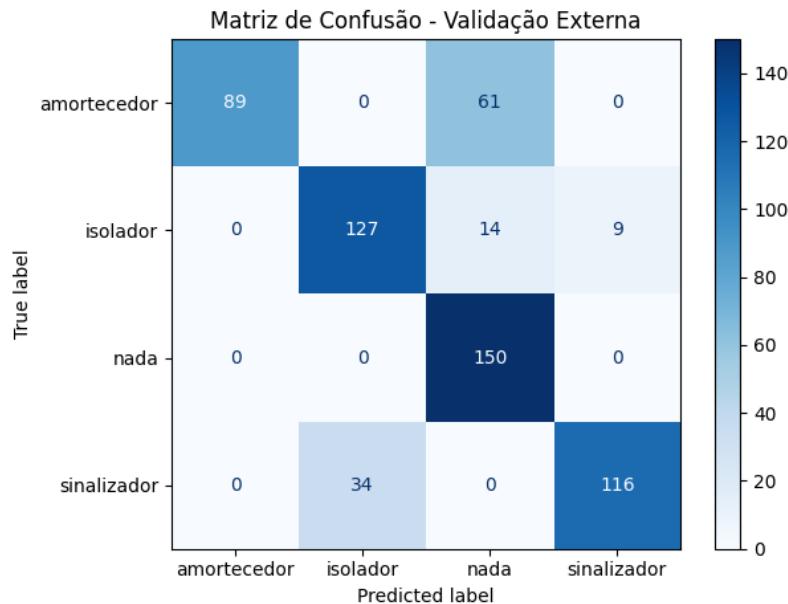
Fonte: Autoria própria (2025).

Figura 45 – Matriz de confusão para a Rede Neural com features extraídas do LiDAR (primeiro robô das topologias secundárias).



Fonte: Autoria própria (2025).

Figura 46 – Matriz de confusão para a Floresta Aleatória com features extraídas do LiDAR (primeiro robô das topologias secundárias).



Fonte: Autoria própria (2025).

A.2.1.5 Features combinadas

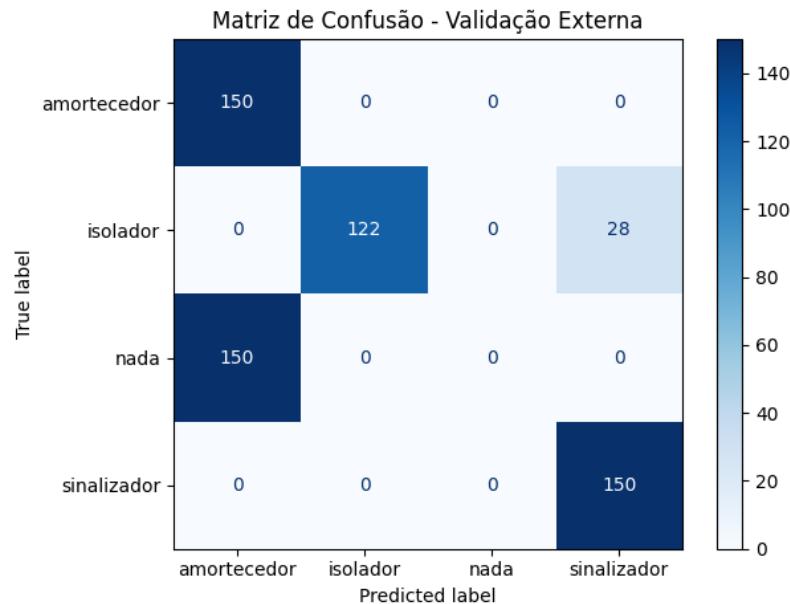
Esta subseção apresenta os resultados obtidos a partir da combinação das *features* extraídas das imagens e dos dados do sensor *LiDAR*, capturadas pelo primeiro robô das topologias secundárias em ambiente simulado com o modelo treinado pelos dados do robô principal.

Tabela 54 – Desempenho dos modelos com features combinadas (primeiro robô das topologias secundárias).

Modelo	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
kNN	-	-	70.33	0.011850
Árvore	-	-	69.67	0.000151
Naive	-	-	69.83	0.000245
Rede	61	0.000034	100.00	0.016043
Floresta	-	-	70.33	0.007828
Média			76.03	0.007223

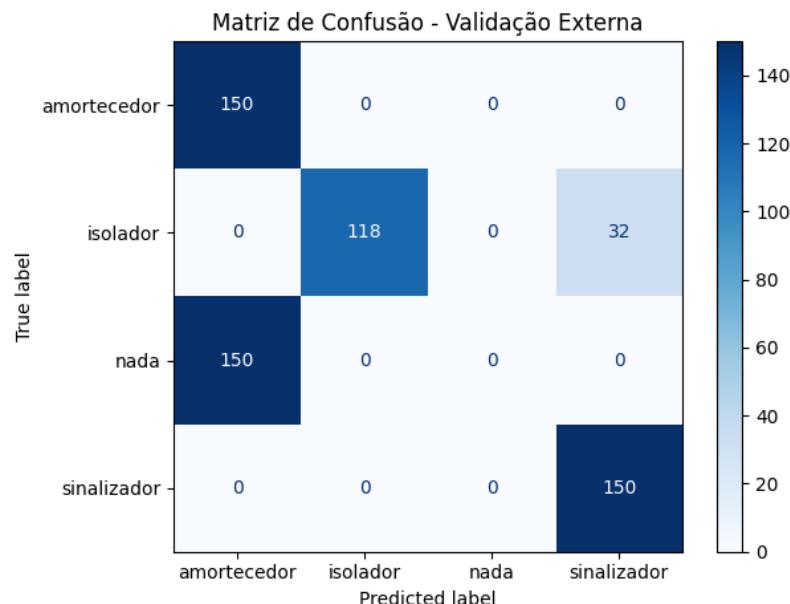
Fonte: Autoria própria (2025).

Figura 47 – Matriz de confusão para o k-Vizinhos mais próximos com features combinadas (primeiro robô das topologias secundárias).



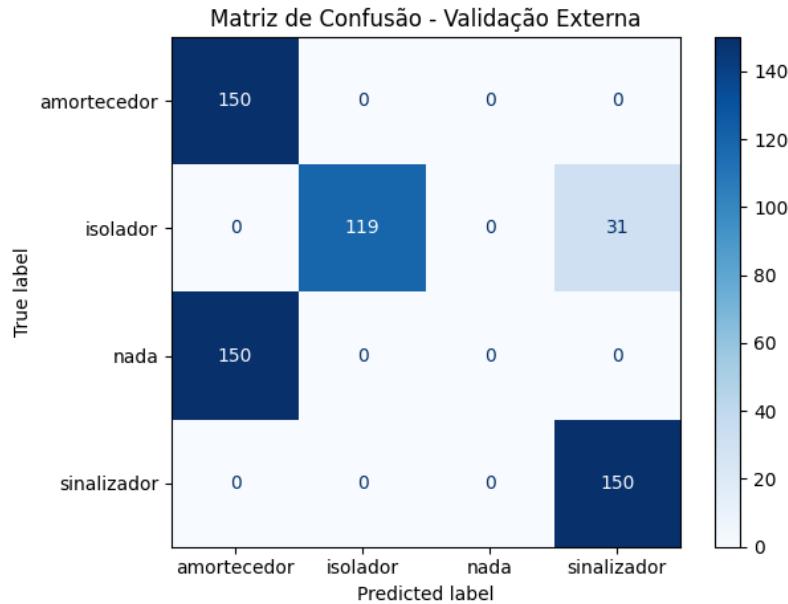
Fonte: Autoria própria (2025).

Figura 48 – Matriz de confusão para a Árvore de Decisão com features combinadas (primeiro robô das topologias secundárias).



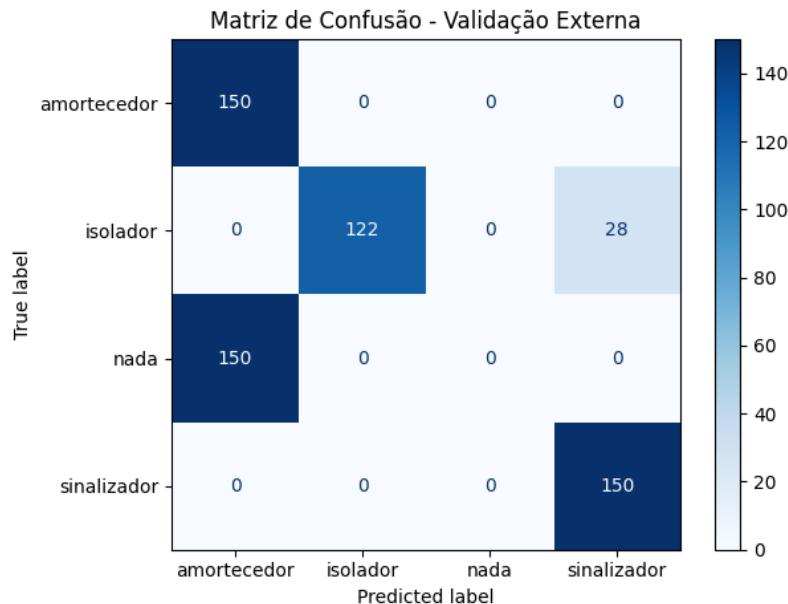
Fonte: Autoria própria (2025).

Figura 49 – Matriz de confusão para o Naive Bayes com features combinadas (primeiro robô das topologias secundárias).



Fonte: Autoria própria (2025).

Figura 50 – Matriz de confusão para a Floresta Aleatória com features combinadas (primeiro robô das topologias secundárias).



Fonte: Autoria própria (2025).

A.2.2 Dados do segundo robô

Nesta subseção são apresentados os resultados obtidos a partir dos dados simulados gerados pela segunda topologia secundária em conjunto com os dados do robô principal usados para treinamento. Os resultados serão resumidos nas médias obtidas.

A.2.2.1 Imagens brutas

Esta subseção apresenta os resultados obtidos com as imagens de profundidade brutas capturadas pelo segundo robô das topologias secundárias em ambiente simulado com o modelo treinado pelos dados do robô principal.

Tabela 55 – Desempenho da SqueezeNet com imagens brutas (robô 1 das topologias secundárias).

Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
5	0.000236	97.67	3.340613

Fonte: Autoria própria (2025).

A.2.2.2 Dados brutos do LiDAR

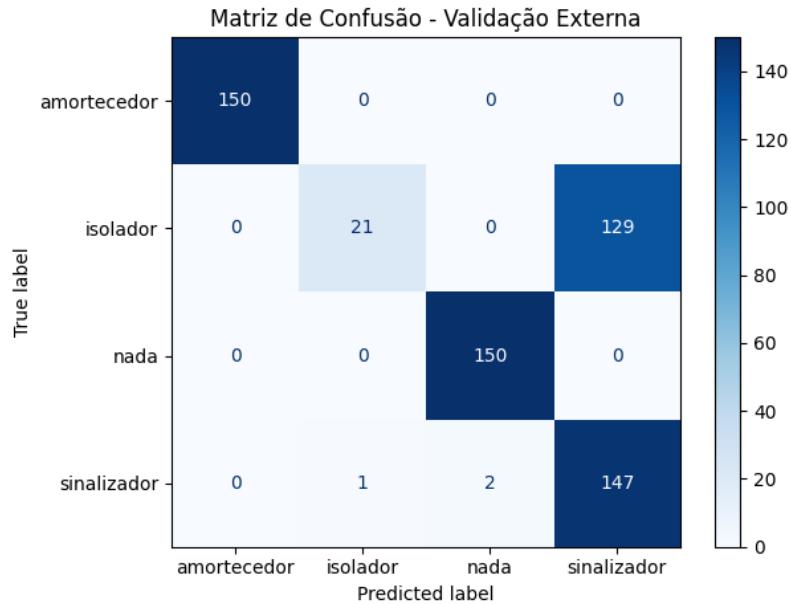
Esta subseção apresenta os resultados obtidos com os dados brutos do LiDAR capturadas pelo segundo robô das topologias secundárias em ambiente simulado com o modelo treinado pelos dados do robô principal.

Tabela 56 – Desempenho dos modelos com dados brutos do LiDAR (segundo robô das topologias secundárias).

Modelo	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
kNN	-	-	78.00	0.019251
Árvore	-	-	52.67	0.000230
Naive	-	-	36.50	0.000785
Rede	200	0.032724	76.33	0.015708
Floresta	-	-	77.50	0.007995
Média			64.20	0.043447

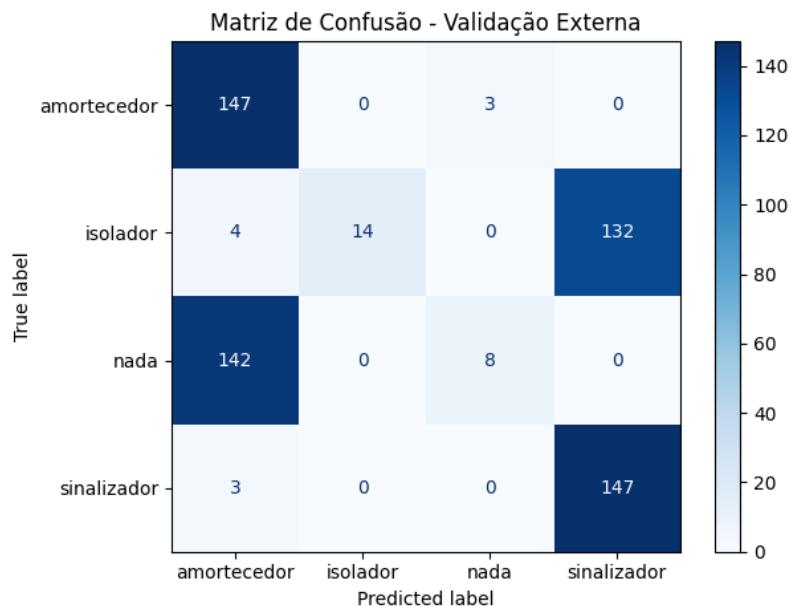
Fonte: Autoria própria (2025).

Figura 51 – Matriz de confusão para o k-Vizinhos mais próximos com dados brutos do LiDAR (segundo robô das topologias secundárias).



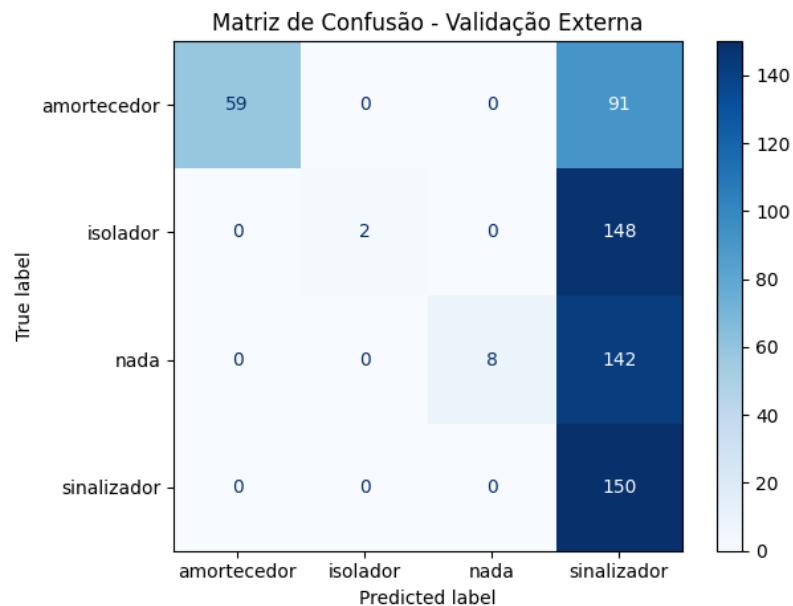
Fonte: Autoria própria (2025).

Figura 52 – Matriz de confusão para a Árvore de Decisão com dados brutos do LiDAR (segundo robô das topologias secundárias).



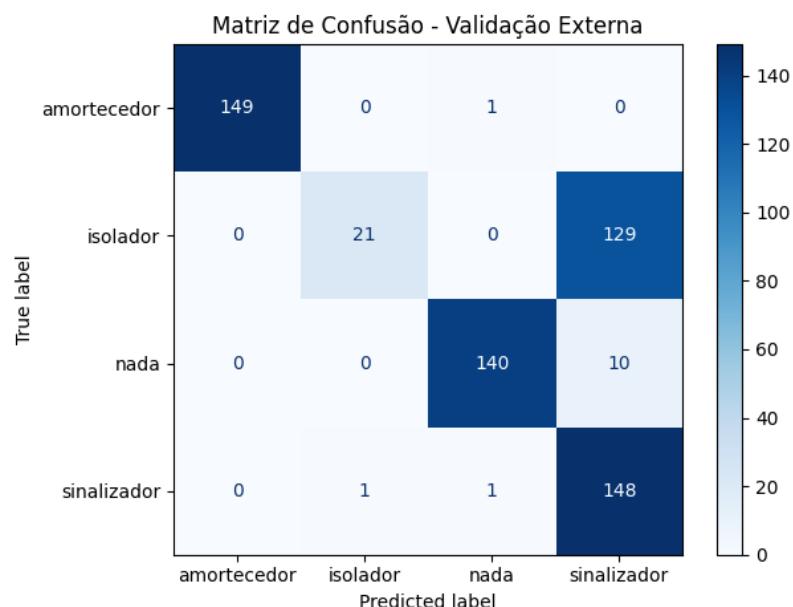
Fonte: Autoria própria (2025).

Figura 53 – Matriz de confusão para o Naive Bayes com dados brutos do LiDAR (segundo robô das topologias secundárias).



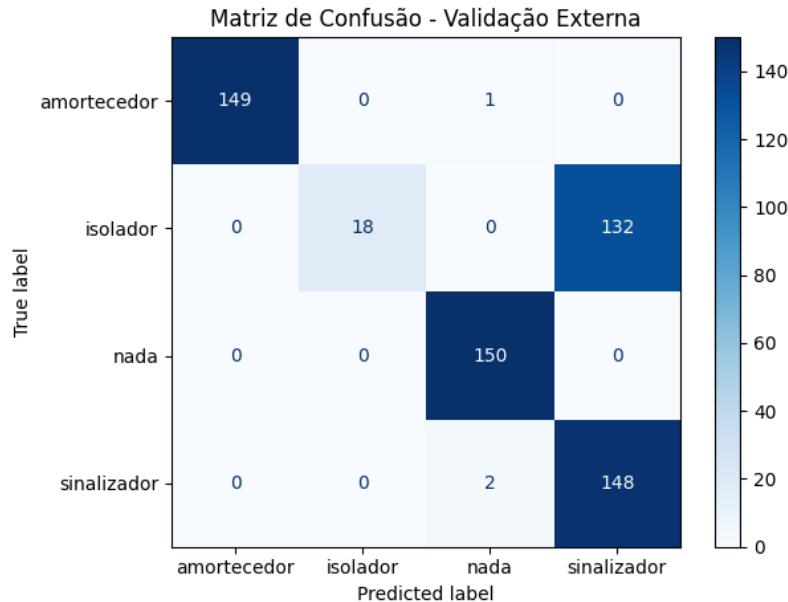
Fonte: Autoria própria (2025).

Figura 54 – Matriz de confusão para a Rede Neural com dados brutos do LiDAR (segundo robô das topologias secundárias).



Fonte: Autoria própria (2025).

Figura 55 – Matriz de confusão para a Floresta Aleatória com dados brutos do LiDAR (segundo robô das topologias secundárias).



Fonte: Autoria própria (2025).

A.2.2.3 Features extraídas das imagens

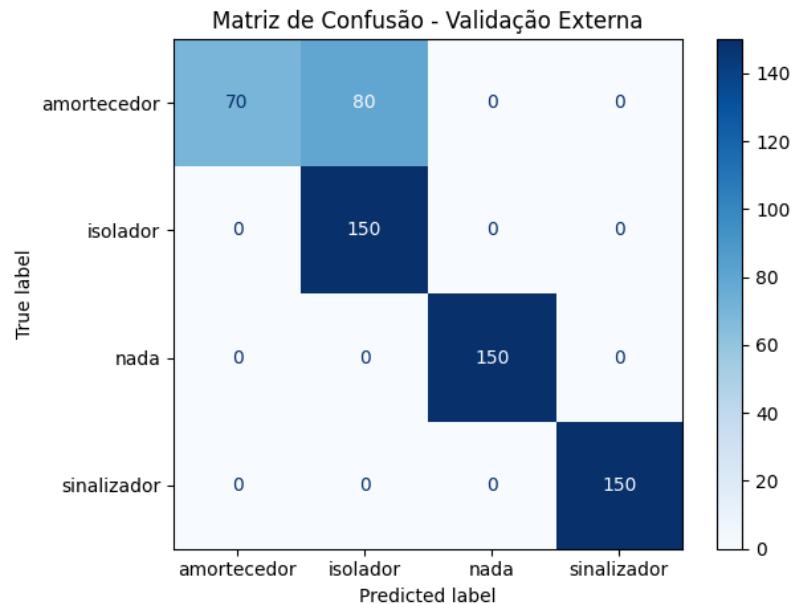
Esta subseção apresenta os resultados obtidos a partir de atributos derivados das imagens capturadas pelo segundo robô das topologias secundárias em ambiente simulado com o modelo treinado pelos dados do robô principal. As imagens foram processadas para extrair informações relevantes (*features*) que pudessem melhorar a capacidade dos modelos de aprendizado de máquina na identificação das classes de objetos.

Tabela 57 – Desempenho dos modelos com features extraídas das imagens (segundo robô das topologias secundárias).

Modelo	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
kNN	-	-	86.67	0.011986
Árvore	-	-	61.17	0.000149
Naive	-	-	47.33	0.000305
Rede	60	0.000764	88.50	0.017994
Floresta	-	-	64.83	0.008024
Média			69.70	0.007692

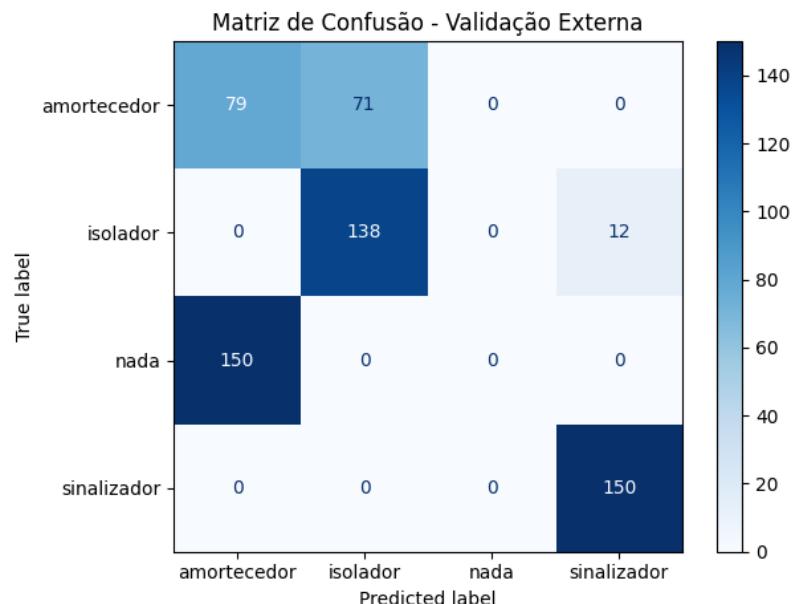
Fonte: Autoria própria (2025).

Figura 56 – Matriz de confusão para o k-Vizinhos mais próximos com features extraídas das imagens (segundo robô das topologias secundárias).



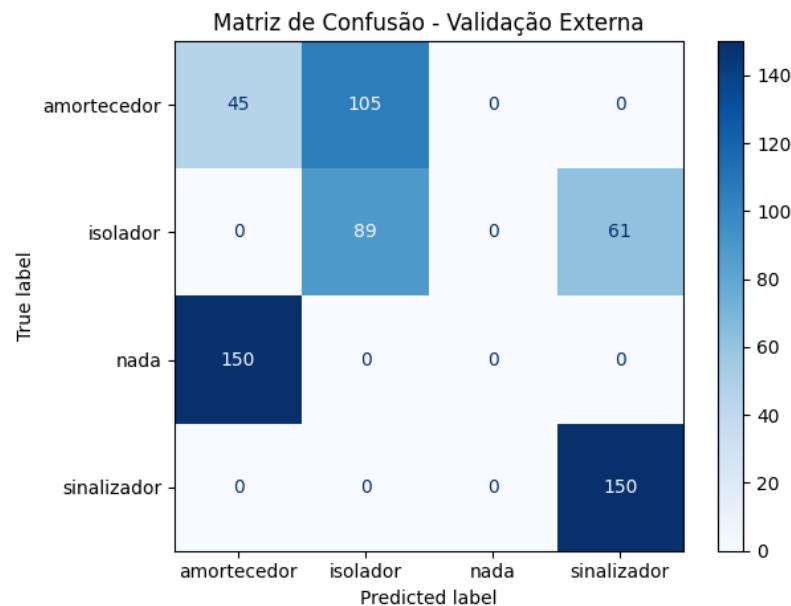
Fonte: Autoria própria (2025).

Figura 57 – Matriz de confusão para a Árvore de Decisão com features extraídas das imagens (segundo robô das topologias secundárias).



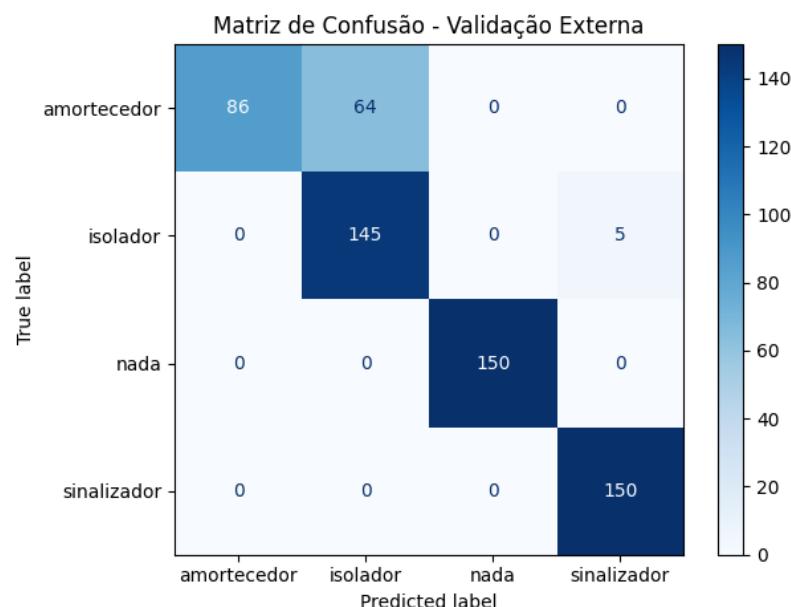
Fonte: Autoria própria (2025).

Figura 58 – Matriz de confusão para o Naive Bayes com features extraídas das imagens (segundo robô das topologias secundárias).



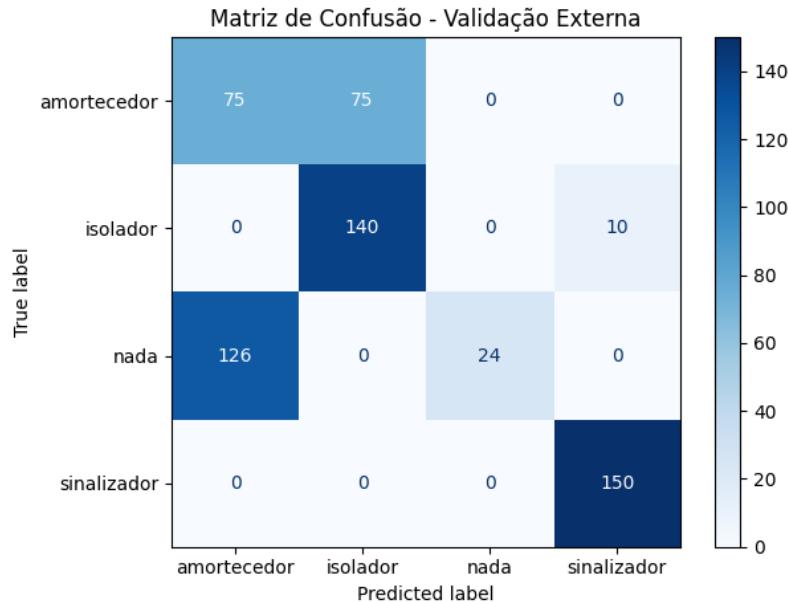
Fonte: Autoria própria (2025).

Figura 59 – Matriz de confusão para a Rede Neural com features extraídas das imagens (segundo robô das topologias secundárias).



Fonte: Autoria própria (2025).

Figura 60 – Matriz de confusão para a Floresta Aleatória com features extraídas das imagens (segundo robô das topologias secundárias).



Fonte: Autoria própria (2025).

A.2.2.4 Features extraídas do LiDAR

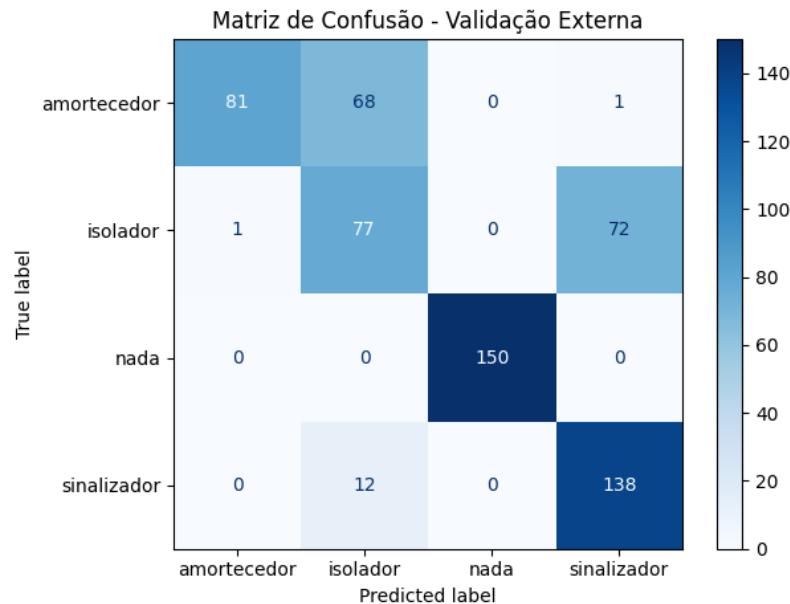
Esta subseção apresenta os resultados obtidos a partir de atributos derivados dos dados de distância do sensor *LiDAR*, capturadas pelo segundo robô das topologias secundárias em ambiente simulado com o modelo treinado pelos dados do robô principal. As imagens foram processadas para extrair informações relevantes (*features*) que pudessem melhorar a capacidade dos modelos de aprendizado de máquina na identificação das classes de objetos.

Tabela 58 – Desempenho dos modelos com features extraídas do LiDAR (segundo robô das topologias secundárias).

Modelo	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
kNN	-	-	74.33	0.013072
Árvore	-	-	74.50	0.000177
Naive	-	-	71.67	0.000255
Rede	200	0.283318	93.50	0.015820
Floresta	-	-	80.67	0.007872
Média			78.93	0.007439

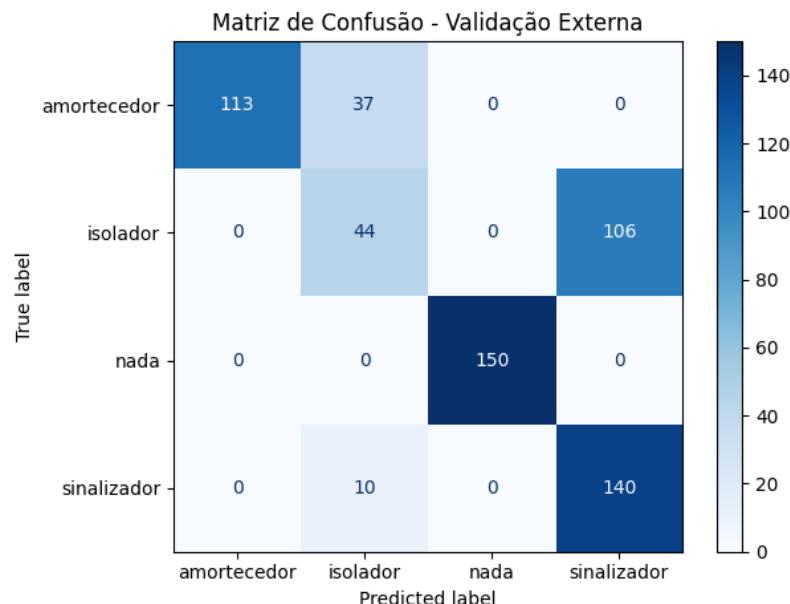
Fonte: Autoria própria (2025).

Figura 61 – Matriz de confusão para o k-Vizinhos mais próximos com features extraídas do LiDAR (segundo robô das topologias secundárias).



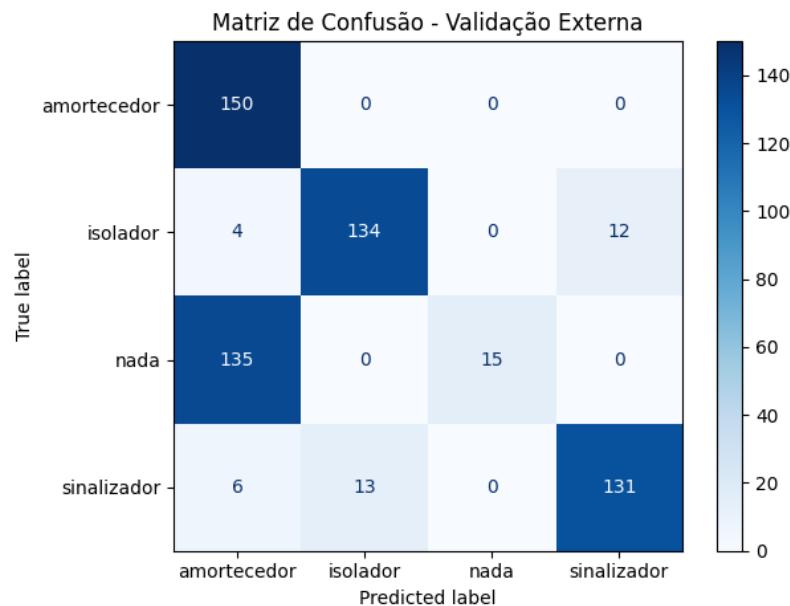
Fonte: Autoria própria (2025).

Figura 62 – Matriz de confusão para a Árvore de Decisão com features extraídas do LiDAR (segundo robô das topologias secundárias).



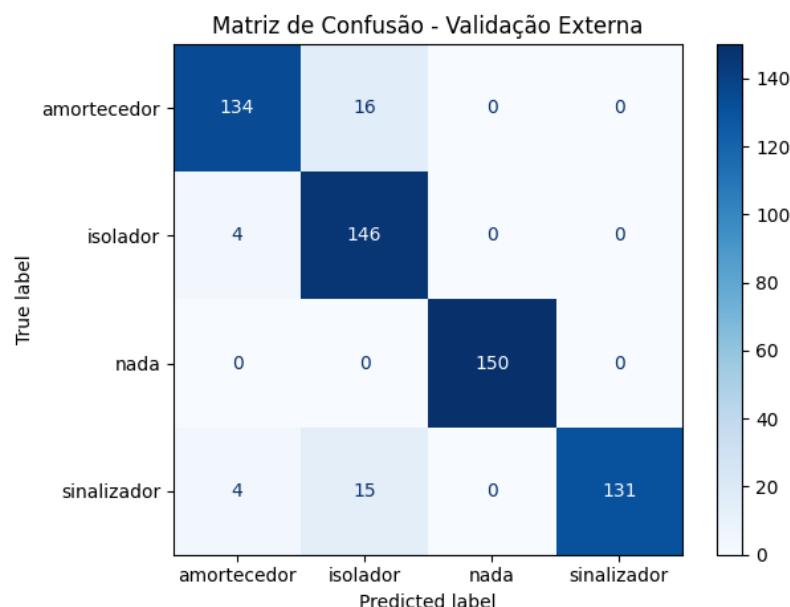
Fonte: Autoria própria (2025).

Figura 63 – Matriz de confusão para o Naive Bayes com features extraídas do LiDAR (segundo robô das topologias secundárias).



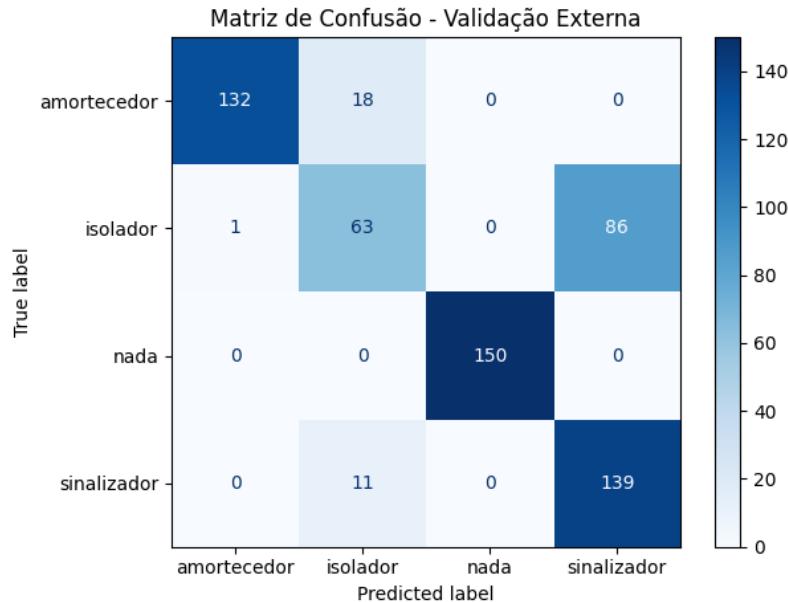
Fonte: Autoria própria (2025).

Figura 64 – Matriz de confusão para a Rede Neural com features extraídas do LiDAR (segundo robô das topologias secundárias).



Fonte: Autoria própria (2025).

Figura 65 – Matriz de confusão para a Floresta Aleatória com features extraídas do LiDAR (segundo robô das topologias secundárias).



Fonte: Autoria própria (2025).

A.2.2.5 Features combinadas

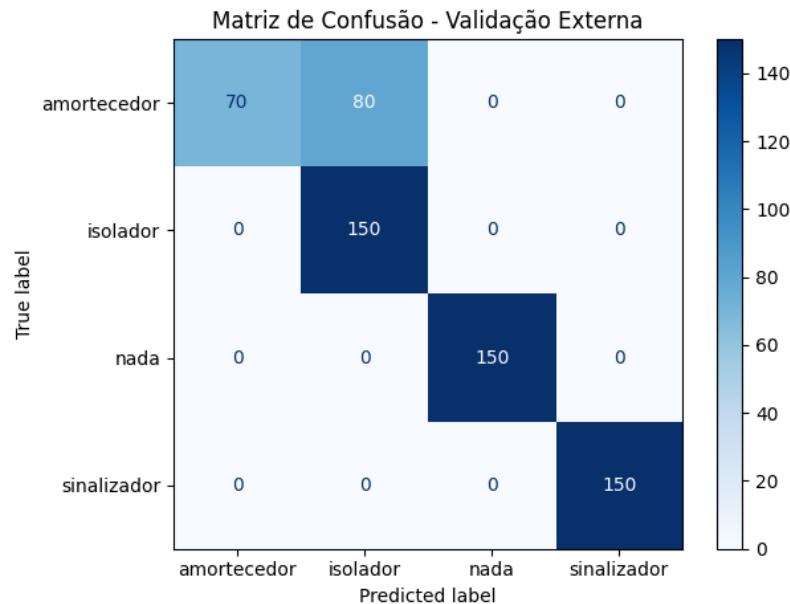
Esta subseção apresenta os resultados obtidos a partir da combinação das *features* extraídas das imagens e dos dados do sensor *LiDAR*, capturadas pelo segundo robô das topologias secundárias em ambiente simulado com o modelo treinado pelos dados do robô principal.

Tabela 59 – Desempenho dos modelos com features combinadas (segundo robô das topologias secundárias).

Modelo	Época Final	Perda Final	Acurácia (%)	Tempo de Validação (s)
kNN	-	-	86.67	0.011865
Árvore	-	-	84.67	0.000149
Naive	-	-	49.50	0.000358
Rede	71	0.000686	88.50	0.015628
Floresta	-	-	64.83	0.006904
Média			74.83	0.006981

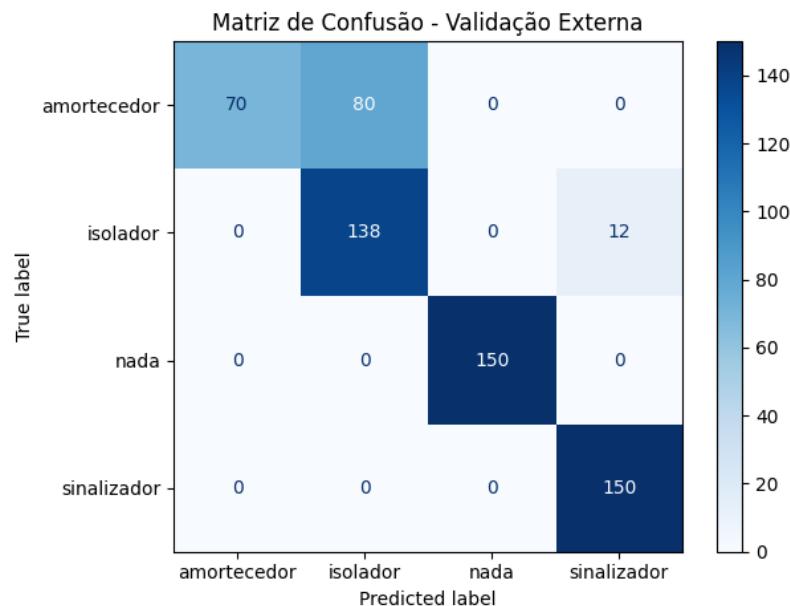
Fonte: Autoria própria (2025).

Figura 66 – Matriz de confusão para o k-Vizinhos mais próximos com features combinadas (segundo robô das topologias secundárias).



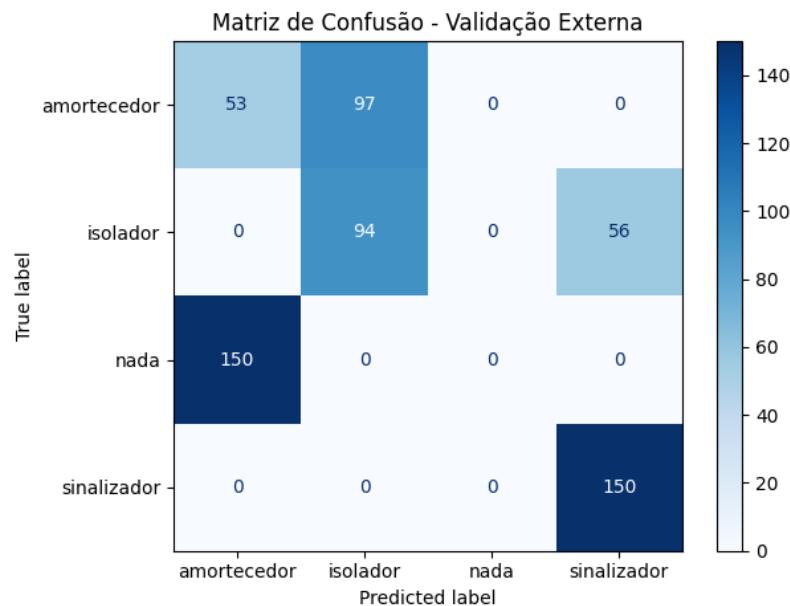
Fonte: Autoria própria (2025).

Figura 67 – Matriz de confusão para a Árvore de Decisão com features combinadas (segundo robô das topologias secundárias).



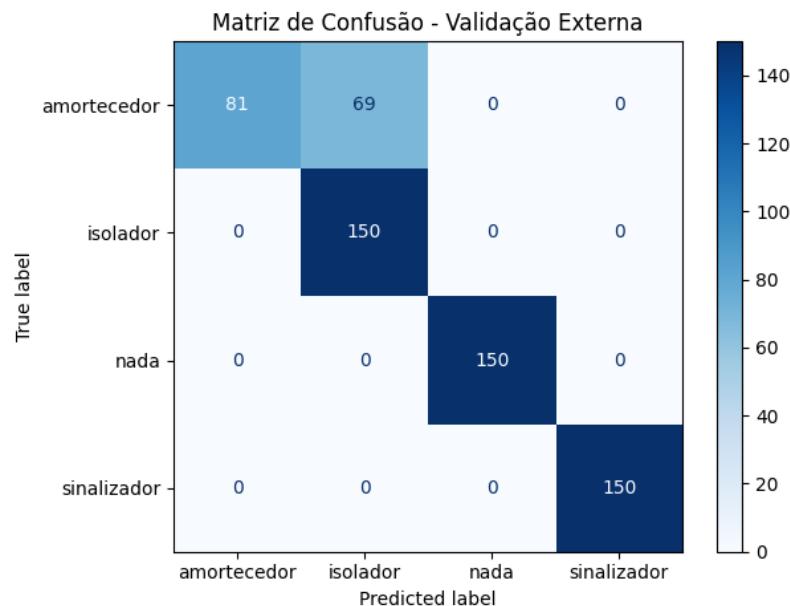
Fonte: Autoria própria (2025).

Figura 68 – Matriz de confusão para o Naive Bayes com features combinadas (segundo robô das topologias secundárias).



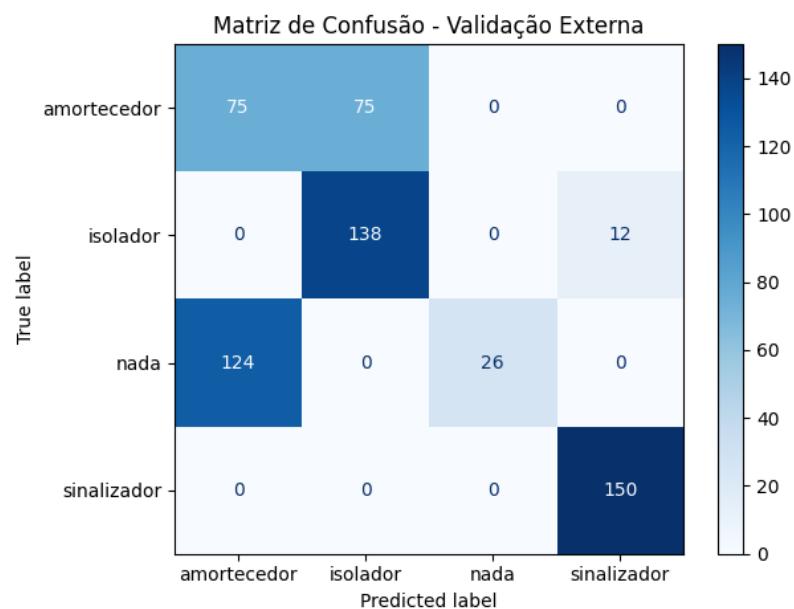
Fonte: Autoria própria (2025).

Figura 69 – Matriz de confusão para a Rede Neural com features combinadas (segundo robô das topologias secundárias).



Fonte: Autoria própria (2025).

Figura 70 – Matriz de confusão para a Floresta Aleatória com features combinadas (segundo robô das topologias secundárias).



Fonte: Autoria própria (2025).