

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM AMBIENTE CONSTRUÍDO

Flávio Francisco dos Reis

**Visão computacional embarcada para acessibilidade: Otimização da rede
YOLO em *Raspberry Pi* para detecção e classificação tipológica de cadeiras de
rodas**

Flávio Francisco dos Reis

**Visão computacional embarcada para acessibilidade: Otimização da rede
YOLO em *Raspberry Pi* para detecção e classificação tipológica de cadeiras de
rodas**

Dissertação apresentada ao Programa de
Pós-graduação em Ambiente Construído
da Universidade Federal de Juiz de Fora
como requisito parcial à obtenção do título
de Mestre em Ambiente Construído. Área
de concentração: Ambiente Construído.

Orientadora: Prof. Dra. Ercília de Stefano

Coorientadora: Prof. Dra. Lilian Genaro Motti

Juiz de Fora

2026

Ficha catalográfica elaborada através do programa de geração automática da Biblioteca Universitária da UFJF, com os dados fornecidos pelo(a) autor(a)

Reis, Flávio.

Visão Computacional Embarcada para Acessibilidade :
Otimização da Rede YOLO em Raspberry Pi para Detecção e
Classificação Tipológica de Cadeiras de Rodas / Flávio Reis. -- 2026.
163 p. : il.

Orientadora: Ercilia de Stefano

Coorientadora: Lilian Genaro Motti

Dissertação (mestrado acadêmico) - Universidade Federal de Juiz de Fora, Faculdade de Engenharia. Programa de Pós-Graduação em Ambiente Construído, 2026.

1. Acessibilidade. 2. Visão computacional. 3. Cadeira de rodas. 4. YOLO. 5. Raspberry Pi. I. de Stefano, Ercilia, orient. II. Genaro Motti, Lilian, coorient. III. Título.

Flávio Francisco dos Reis

**Visão Computacional Embarcada para Acessibilidade: Otimização da Rede
YOLO em Raspberry Pi para Detecção e Classificação Tipológica de Cadeiras
de Rodas**

Dissertação apresentada ao Programa de Pós-graduação em Ambiente Construído da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Ambiente Construído. Área de concentração: Ambiente Construído.

Aprovada em 29 de Maio de 2026

BANCA EXAMINADORA



Documento assinado digitalmente

ERCILIA DE STEFANO

Data: 09/07/2026 13:06:04-0300

Verifique em <https://validar.iti.gov.br>

Dra. Ercilia de Stefano - Orientadora
Universidade Federal de Juiz de Fora



Documento assinado digitalmente

LILIAN GENARO MOTTI ADER

Data: 09/07/2026 12:51:07-0300

Verifique em <https://validar.iti.gov.br>

Prof. Dra. Lilian Genaro Motti - Coorientadora
Universidade de Limerick



Documento assinado digitalmente

JOSE CRISTIANO PEREIRA

Data: 09/07/2026 08:12:36-0300

Verifique em <https://validar.iti.gov.br>

Dr. José Cristiano Pereira
Universidade Federal de Juiz de Fora



Documento assinado digitalmente

PATRICK BARBOSA MURATORI

Data: 06/07/2026 16:18:32-0300

Verifique em <https://validar.iti.gov.br>

Prof. Dr. Patrick Barbosa Moratori
Universidade Federal Fluminense

Dedico este trabalho primeiramente a Deus, aos meus pais que se esforçaram tanto para que eu conseguisse alcançar meus objetivos. Aos meus amigos que sempre são um ponto de apoio e aos meus alunos que também me ensinam todos os dias.

AGRADECIMENTOS

A Deus, agradeço primeiramente, por me ter dado a sabedoria para persistir sempre.

Aos meus pais, Jurandina e Francisco, agradeço com profunda emoção. Nunca mediram esforços para que eu pudesse alcançar meus objetivos, e o que sou hoje devo, antes de tudo, a vocês.

À minha irmã, Sirlei, agradeço de modo muito particular. Foi a sua experiência cotidiana que deu origem a esta pesquisa e que sustentou cada decisão técnica aqui registrada. Esta dissertação só existe porque você existe.

Aos meus amigos, agradeço por estarem sempre presentes, por alegrarem meus dias e por tornarem mais leve o que, sozinho, eu sentiria pesado.

Aos meus alunos, agradeço por me mostrarem, dia após dia, que a vida tem muito mais camadas do que parece e que, às vezes, é necessário mudar o ponto de vista. Alguns deles tornaram-se amigos, desses que ficam para a vida toda.

À minha orientadora, Profa. Dra. Ercilia de Stefano, e à minha coorientadora, Profa. Dra. Lilian Genaro Motti, agradeço por todo o apoio e conhecimento transmitidos ao longo desta caminhada. Sem o cuidado e o rigor de vocês, eu não teria alcançado os resultados que aqui apresento.

À Barbara, amiga que a vida me trouxe, agradeço a disposição constante em me ajudar e em revisar meu texto sempre que precisei. Estendo esse agradecimento aos demais amigos próximos que, ao longo desses dois anos, ofereceram leitura atenta, palavra certa ou simples escuta. Cada um sabe a parte que teve nesse percurso.

Aos meus amigos de mestrado, Arthur e Natália, e a tantos outros que dividiram comigo essa travessia, agradeço por terem feito dela uma experiência ímpar.

À Universidade Federal de Juiz de Fora, em particular ao Programa de Pós-graduação em Ambiente Construído e a seu corpo docente, agradeço pela formação rigorosa e pela infraestrutura institucional que tornaram esta pesquisa possível. Aos membros da banca examinadora, agradeço a disponibilidade e as contribuições que aperfeiçoam este trabalho.

Por fim, agradeço aos dias difíceis. Foram eles que me mostraram que, quando estamos prestes a desistir, é sinal de que estamos perto de conseguir.

"Neste mundo aterrorizante, tudo o que temos são as conexões que fazemos."
(Bob-Waksberg, 2014).

RESUMO

A acessibilidade no ambiente construído depende da capacidade do espaço de reconhecer demandas específicas das pessoas que o utilizam. Os sistemas atuais de automação predial, contudo, operam de forma indiferenciada diante das pessoas que se deslocam em cadeira de rodas e tratam equipamentos morfologicamente distintos como uma classe única, o que limita a precisão das adaptações ambientais e dispersa parte do potencial das tecnologias assistivas inteligentes. Esta dissertação investiga em que medida a combinação entre uma classificação tipológica consolidada e a otimização de modelos de detecção em *hardware* embarcado pode constituir uma plataforma de baixo custo capaz de identificar tipologias específicas de cadeiras de rodas em tempo real, fornecendo subsídios técnicos para a futura integração com sistemas de automação como portas, elevadores e sinalizações inteligentes. O estudo, de abordagem aplicada e exploratória, fundamenta-se no método hipotético-dedutivo conforme as diretrizes de Marconi e Lakatos (2017) e adota como hipótese operacional o atingimento de precisão média $mAP@0.5$ igual ou superior a 0,85 e taxa de inferência igual ou superior a 15 quadros por segundo no *Raspberry Pi*, sob a premissa transversal de baixo custo e preservação da privacidade do usuário. A pesquisa consolida, a partir das normas ISO 9999:2022 e ABNT NBR 9050:2020 e dos catálogos de quatro fabricantes nacionais e internacionais, uma classificação de onze tipologias para fins de detecção computacional. O *corpus* reúne 19.452 imagens brutas oriundas de quatro repositórios públicos e duas rodadas de busca direcionada, reduzidas, após filtragens sucessivas por resolução, deduplicação e triagem visual, a 2.000 imagens anotadas no *software LabelImg* e particionadas estratificadamente em 1.394 imagens de treinamento, 394 de validação e 212 de teste, distribuídas de forma desbalanceada entre as onze classes. A pesquisa treina em CPU, ao longo de aproximadamente 41 horas distribuídas em cinco dias contínuos, os modelos *YOLOv5n*, *YOLOv5s*, *YOLOv8n* e *YOLOv8s*, dos quais o *YOLOv8n* é selecionado para implantação por apresentar o melhor $mAP@0.5$ agregado, de 0,711; embora esse valor não atinja o limiar formal de 0,85, quatro tipologias superaram individualmente esse limiar e uma quinta permaneceu em situação fronteiriça. O modelo é convertido para o formato *NCNN* em meia precisão com queda de apenas 0,003 ponto absoluto na precisão ($mAP@0.5$ de 0,708) e opera no *Raspberry Pi 5* a 13,3 quadros por segundo em cenário real e 15,1 quadros por segundo em *benchmark*

sintético (de modo que o limiar de 15 quadros por segundo foi alcançado apenas em condição sintética), valor amplamente compatível com a velocidade de deslocamento de usuários de cadeira de rodas reportada por Pinnock Branford *et al.* (2025) em 0,43 metros por segundo. O sistema integra três aplicativos articulados como triangulação metodológica, opera com autonomia entre quatro e cinco horas em modo bateria por meio do módulo *UPS Geekworm X1202* e mantém estabilidade térmica em regime sustentado. Os achados evidenciam um paradoxo arquitetural pouco discutido na literatura (em corpus desbalanceado de escala moderada, o aumento de parâmetros pode prejudicar o desempenho) e demonstram a viabilidade técnica e econômica da detecção tipológica embarcada como camada de percepção para uma nova geração de adaptações sensíveis à diversidade dos equipamentos de mobilidade no ambiente construído.

Palavras-chave: acessibilidade; visão computacional; cadeira de rodas; *YOLO*; *Raspberry Pi*; tecnologia assistiva.

ABSTRACT

Accessibility in the built environment depends on a space's ability to recognize the specific needs of the people who use it. Current building automation systems, however, treat people who use *wheelchairs* in the same way as everyone else and treat morphologically distinct devices as a single category, which limits the precision of environmental adaptations and undermines the potential of smart assistive technologies. This dissertation investigates to what extent the combination of a consolidated typological classification and the optimization of detection models in embedded *hardware* can constitute a low-cost platform capable of identifying specific types of *wheelchairs* in real time, providing technical support for future integration with automation systems such as doors, elevators, and smart signage. The study, which takes an applied and exploratory approach, is based on the hypothetical-deductive method in accordance with the guidelines of Marconi and Lakatos (2017) and adopts as its operational hypothesis the achievement of a mean accuracy $mAP@0.5$ of 0.85 or higher and an inference rate of 15 frames per second or higher on the *Raspberry Pi*, under the overarching premise of low cost and preservation of user privacy. Based on the ISO 9999:2022 and ABNT NBR 9050:2020 standards, as well as catalogs from four domestic and international manufacturers, this study establishes a classification of eleven types for the purposes of computer-aided detection. The *corpus* comprises 19,452 raw images sourced from four public repositories and two rounds of targeted searches, reduced, after successive filtering by resolution, deduplication, and visual screening, to 2,000 images annotated in *Labellmg software* and stratified into 1,394 training images, 394 validation images, and 212 test images, unevenly distributed across the eleven classes. The study trains the *YOLOv5n*, *YOLOv5s*, *YOLOv8n*, and *YOLOv8s*, of which *YOLOv8n* is selected for deployment due to its aggregate $mAP@0.5$ of 0.711; although this value does not reach the formal threshold of 0.85, four classes individually exceeded it and a fifth remained in a borderline situation. The model is converted to the *NCNN* format in half-precision with a drop of only 0.003 absolute points in accuracy ($mAP@0.5$ of 0.708) and operates on the *Raspberry Pi 5* at 13.3 frames per second in a real-world scenario and 15.1 frames per second in a synthetic benchmark (so that the 15 FPS threshold was reached only under synthetic conditions), a value largely consistent with the travel speed of *wheelchair* users reported by Pinnock Branford *et al.* (2025) at 0.43 meters per second. The system

integrates three applications linked through methodological triangulation, operates autonomously for four to five hours on battery power via the Geekworm X1202 UPS module, and maintains thermal stability under sustained operation. The findings highlight an architectural paradox rarely discussed in the literature, according to which, in a moderately unbalanced *corpus*, increasing the number of parameters can impair performance, and demonstrate the technical and economic feasibility of embedded typological detection as a perception layer for a new generation of adaptations sensitive to the diversity of mobility equipment in the built environment.

Keywords: accessibility; computer vision; *wheelchair*; YOLO; *Raspberry Pi*; assistive technology.

LISTA DE ILUSTRAÇÕES

Figura 1 - A: Cadeiras de rodas manual; B: Cadeira de rodas motorizada e C: Cadeira de rodas esportiva (Basquete)	29
Figura 2 - Módulo de Referência	31
Figura 3 - Comparação entre extração de atributos clássica e aprendizado profundo	34
Figura 4 - Estrutura básica de uma Rede Neural Convolucional (CNN).....	35
Figura 5 - Representação visual da métrica IoU em detecção de objetos	37
Figura 6 - O Sistema de Detecção YOLO	39
Figura 7 - Distribuição da base de dados e exemplo de matriz de confusão para validação da detecção de cadeiras de rodas	43
Figura 8 - Fluxograma das etapas metodológicas.....	53
Figura 9 - Raspberry Pi 5	82
Figura 10 - Webcam Logitech 720p	83
Figura 11 - Tela IPS touchscreen de 7 Polegadas 1024x600 pixels	83
Figura 12 - Aplicativo Especialista em modo “Aguardando”.....	97
Figura 13 - Aplicativo Especialista em modo de Detecção.....	98
Figura 14 - Aplicativo Detector Genérico.....	100
Figura 15 - Aplicativo Coletor	102
Figura 16 - Módulo UPS Geekworm X1202	106
Figura 17 - Distribuição das instâncias por tipologia no conjunto de treinamento...	115
Figura 18 - Evolução das funções de perda no conjunto de treinamento.....	119
Figura 19 - Curvas de perda de treinamento e validação dos modelos YOLOv8n e YOLOv8s.....	120
Figura 20 - Matriz de confusão normalizada do modelo YOLOv8n no conjunto de validação.	125
Figura 21 - Curva de Precisão × Revocação por classe para o modelo YOLOv8n.	126
Figura 22 - Sistema assistivo em operação de campo.....	133

LISTA DE ILUSTRAÇÕES ANEXO

Figura A 1 - Manual Dobrável.....	155
Figura A 2 - Manual Ativa (Monobloco/Rígida).....	155
Figura A 3 - Motorizada (Elétrica).....	156
Figura A 4 - Postural (Tilt-in-Space/Reclinável).....	156
Figura A 5 - Pediátrica/Infantil	157
Figura A 6 - Bariátrica (Heavy Duty).....	157
Figura A 7 - Esportiva.....	158
Figura A 8 - Banho/Higiênica.....	158
Figura A 9 - Hospitalar/Transporte	159
Figura A 10 - Standing (Verticalização).....	159
Figura A 11 - Anfíbia/Praia	160

LISTA DE QUADROS

Quadro 1 - Relação entre o modelo biopsicossocial e o paradigma tecnológico da TA	27
Quadro 2 - Quadro comparativo entre as normas ISO 9999:2022 e a ABNT NBR 9050:2020	32
Quadro 3 - Evolução e características das versões YOLO	40
Quadro 4 - Fontes de imagens, volumes e status de inclusão no corpus	57
Quadro 5 - Classificação Tipológica Consolidada de Cadeiras de Rodas	59
Quadro 6 - Configuração de hardware da estação de trabalho utilizada para treinamento	71
Quadro 7 - Hiperparâmetros de treinamento dos modelos YOLOv5 e YOLOv8	73
Quadro 8 - Síntese das métricas de avaliação	79
Quadro 9 - Configuração do Raspberry Pi 5 e do ambiente de software	82
Quadro 10 - Modelos exportados para a arquitetura ARM	86
Quadro 11 - Benchmark de inferência TFLite no Raspberry Pi 5 (CPU, XNNPACK, 2,4 GHz)	90
Quadro 12 - Evolução do desempenho do YOLOv8n com otimizações incrementais (benchmark sintético, 640 × 640)	91
Quadro 13 - Desempenho do YOLOv8n NCNN FP16 com câmera real e display (640×640, 2,8 GHz)	92
Quadro 14 - Resultados do teste de estabilidade térmica do sistema embarcado (60 minutos de inferência contínua)	93
Quadro 15 - Síntese dos três aplicativos do sistema operacional embarcado	103
Quadro 16 - Comparação experimental entre as configurações de alimentação avaliadas para o Raspberry Pi 5	107
Quadro 17 - Síntese comparativa dos resultados de treinamento dos quatro modelos avaliados	118
Quadro 18 - Desempenho mAP@0.5 por tipologia nos três modelos com dados completos	123

LISTA DE ABREVIATURAS E SIGLAS

ABNT Associação Brasileira de Normas Técnicas

ADAS *Advanced Driver Assistance System (Sistema Avançado de Assistência ao Condutor)*

AMD *Advanced Micro Devices*

API *Application Programming Interface (Interface de Programação de Aplicações)*

ARM *Advanced RISC Machines (arquitetura de processadores)*

ARP *Address Resolution Protocol*

CAT Comitê de Ajudas Técnicas

CIF Classificação Internacional de Funcionalidade, Incapacidade e Saúde

CloU *Complete Intersection over Union*

CNN *Convolutional Neural Network (Rede Neural Convolucional)*

COCO *Common Objects in Context*

CPU *Central Processing Unit (Unidade Central de Processamento)*

CSPNet *Cross Stage Partial Network*

CSV *Comma-Separated Values (Valores Separados por Vírgula)*

CUDA *Compute Unified Device Architecture*

DDR5 *Double Data Rate 5*

DICOM *Digital Imaging and Communications in Medicine*

F1 F1-score (média harmônica entre precisão e revocação)

FN *False Negative (Falso Negativo)*

FP *False Positive (Falso Positivo)*

FP16 *Floating Point 16-bit (Ponto Flutuante de 16 bits)*

FP32 *Floating Point 32-bit (Ponto Flutuante de 32 bits)*

FPS *Frames Per Second (Quadros por Segundo)*

GB Gigabyte

GDDR6 *Graphics Double Data Rate 6*

GPIO *General Purpose Input/Output (Entrada/Saída de Uso Geral)*

GPU *Graphics Processing Unit (Unidade de Processamento Gráfico)*

HAT *Hardware Attached on Top*

HSV *Hue, Saturation, Value (Matiz, Saturação, Valor)*

HUD *Head-Up Display*

IA Inteligência Artificial

IBGE Instituto Brasileiro de Geografia e Estatística
INT8 Integer 8-bit (Inteiro de 8 bits)
IoU Intersection over Union
IP Internet Protocol
IPS In-Plane Switching
ISO International Organization for Standardization
JPEG Joint Photographic Experts Group
LLC Limited Liability Company
mAP mean Average Precision (Precisão Média)
MB Megabyte
MIT Massachusetts Institute of Technology
MKL Math Kernel Library
MPS Metal Performance Shaders
MR Módulo de Referência
NBR Norma Brasileira
NCNN Neural Network framework otimizado para dispositivos móveis (Tencent)
NEON Advanced SIMD extension (extensão de instruções vetoriais ARM)
NVIDIA NVIDIA Corporation
NVMe Non-Volatile Memory Express
ODS Objetivos de Desenvolvimento Sustentável
ONNX Open Neural Network Exchange
PCIe Peripheral Component Interconnect Express
PD Power Delivery
PNS Pesquisa Nacional de Saúde
PVC Polyvinyl Chloride (Policloreto de Vinila)
RAM Random Access Memory (Memória de Acesso Aleatório)
R-CNN Region-based Convolutional Neural Network (Rede Neural Convolucional baseada em Regiões)
RDNA Radeon DNA (arquitetura de GPU AMD)
RGB Red, Green, Blue (Vermelho, Verde, Azul)
RGB-D Red, Green, Blue, Depth (Vermelho, Verde, Azul, Profundidade)
RISC Reduced Instruction Set Computer
ROCm Radeon Open Compute
ROS Robot Operating System

SCP Secure Copy Protocol

SGD Stochastic Gradient Descent (Gradiente Descendente Estocástico)

SIG Sistema de Informação Geográfica

SORT Simple Online and Realtime Tracking

SPPF Spatial Pyramid Pooling – Fast

SSD Solid State Drive

SSH Secure Shell

TA Tecnologia Assistiva

TB Terabyte

TFLite TensorFlow Lite (framework de inferência para dispositivos embarcados)

TN True Negative (Verdadeiro Negativo)

TP True Positive (Verdadeiro Positivo)

TXT Text file (extensão de arquivo de texto simples)

UPS Uninterruptible Power Supply (Fonte de Alimentação Ininterrupta)

USB Universal Serial Bus

USB-C Universal Serial Bus Type-C

XNNPACK biblioteca de operadores otimizados para redes neurais (Google)

YAML YAML Ain't Markup Language

YOLO You Only Look Once

YOLOv1 You Only Look Once version 1

YOLOv3 You Only Look Once version 3

YOLOv4 You Only Look Once version 4

YOLOv5 You Only Look Once version 5

YOLOv7 You Only Look Once version 7

YOLOv8 You Only Look Once version 8

SUMÁRIO

1 INTRODUÇÃO	18
1.1 CONTEXTUALIZAÇÃO	18
1.2 PROBLEMA DE PESQUISA.....	21
1.3 ESTRUTURA DO TRABALHO	23
2 FUNDAMENTAÇÃO TEÓRICA	26
2.1 TECNOLOGIAS ASSISTIVAS E MOBILIDADE ACESSÍVEL	26
2.1.1 Conceitos e classificações de tecnologia assistiva.....	26
2.1.2 Demandas de mobilidade e identificação de cadeirantes	28
2.1.3 Tipologia e Classificações de Cadeiras de Rodas	29
2.2 VISÃO COMPUTACIONAL PARA DETECÇÃO DE OBJETOS	32
2.2.1 Evolução da detecção: de métodos clássicos a <i>deep learning</i>	33
2.2.2 Arquiteturas baseadas em redes convolucionais	34
2.2.3 Métricas de Desempenho em Detecção de Objetos.....	36
2.3 A FAMÍLIA YOLO E SUAS PRINCIPAIS APLICAÇÕES EM AMBIENTES REAIS	38
2.3.1 Princípios de funcionamento do YOLO.....	38
2.3.2 Comparação entre versões (YOLOv5, YOLOv7 etc.).....	39
2.3.3 Uso do YOLO em dispositivos de baixa potência	41
2.4 PROCESSOS DE TREINAMENTO DE MODELOS DE DETECÇÃO.....	41
2.4.1 Preparação e anotação de bases de Imagens	42
2.4.2 Estratégias de divisão de dados (<i>train/val/test</i>)	43
2.4.3 Parâmetros de treinamento e critérios de convergência.....	44
2.5 IMPLEMENTAÇÃO EM <i>HARDWARE</i> EMBARCADO.....	45
2.5.1 Características e limitações do <i>Raspberry Pi</i>	45
2.5.2 Conversão e otimização de modelos (<i>TensorFlow Lite</i> , quantização)	46
2.5.3 Desempenho Esperado em <i>Edge Computing</i> (FPS, latência, consumo)	46
2.6 TRABALHOS RELACIONADOS.....	47
2.6.1 Detecção de pessoas e cadeirantes com <i>deep learning</i>	47
2.6.2 Sistemas assistivos com reconhecimento visual	48
2.6.3 Aplicações em ambientes públicos e acessibilidade	50
2.7 SÍNTESE E CONSIDERAÇÕES PARCIAIS	51
3 METODOLOGIA	52
3.1 TIPO E ENFOQUE DA PESQUISA	53
3.2 OBJETO DE ESTUDO E CONTEXTO DE APLICAÇÃO	54

3.3 CONSTITUIÇÃO DO <i>CORPUS</i> DE IMAGENS.....	55
3.4 ELABORAÇÃO DO QUADRO DE CLASSIFICAÇÃO TIPOLOGICA.....	58
3.5 ANOTAÇÃO E DIVISÃO DOS DADOS	62
3.5.1 Protocolo de Anotação	63
3.5.2 Divisão dos Dados (Treinamento/Validação/Teste)	67
3.6 PROCESSOS DE TREINAMENTO DOS MODELOS.....	70
3.6.1 Configurações dos Modelos e Hiperparâmetros	71
3.6.2 CrITÉrios de Avaliação e Métricas	76
3.7 IMPLEMENTAÇÃO NO DISPOSITIVO EMBARCADO.....	80
3.7.1 Plataforma Embarcada e Configuração do Ambiente	81
3.7.2 Conversão dos Modelos para Arquitetura <i>ARM</i>	84
3.7.3 Protocolos de Teste no <i>Raspberry Pi</i>	87
3.8 SISTEMA OPERACIONAL EMBARCADO	94
3.8.1 Arquitetura modular e o módulo comum	95
3.8.2 Aplicativo 1: Detector Especialista com Registro de Eventos	96
3.8.3 Aplicativo 2: Detector Genérico para Validação Comparativa	99
3.8.4 Aplicativo 3: Coletor para Retreinamento Iterativo	100
3.9 VIABILIDADE OPERACIONAL EM CAMPO	103
3.10 ANÁLISE DOS RESULTADOS.....	108
3.11 LIMITAÇÕES E PREMISSAS	110
3.12 SÍNTESE E CONSIDERAÇÕES PARCIAIS	113
4 RESULTADOS	114
4.1 CARACTERIZAÇÃO DO <i>CORPUS</i> DE TREINAMENTO	114
4.2 RESULTADOS DO TREINAMENTO COMPARATIVO DOS QUATRO MODELOS.....	117
4.3 ANÁLISE DO DESEMPENHO POR CLASSE	121
4.4 DESEMPENHO DO MODELO NO DISPOSITIVO EMBARCADO	128
4.5 OPERAÇÃO SUSTENTADA EM CAMPO	131
4.6 DISCUSSÃO CRÍTICA: ATINGIMENTO PARCIAL DOS CRITÉRIOS DA HIPÓTESE OPERACIONAL.....	134
4.7 SÍNTESE E CONSIDERAÇÕES PARCIAIS	136
5 CONCLUSÃO	138
5.1 CONTRIBUIÇÕES CIENTÍFICAS E TÉCNICAS	139
5.2 LIMITAÇÕES	139
5.3 TRABALHOS FUTUROS.....	140
REFERÊNCIAS	142
APÊNDICE A - DETALHAMENTO TÉCNICO E REPRODUTIBILIDADE	148
A.1 SCRIPTS DE CONSTITUIÇÃO E ORGANIZAÇÃO DO <i>CORPUS</i>	148

A.2 ESTRUTURA DE DIRETÓRIOS DO CONJUNTO DE DADOS	148
A.3 CORREÇÃO DO LABELIMG PARA COMPATIBILIDADE COM O PYTHON.....	149
A.4 COMANDOS DE EXPORTAÇÃO DOS MODELOS PARA A ARQUITETURA ARM.....	150
A.5 CONFIGURAÇÃO DO AMBIENTE EMBARCADO.....	150
A.6 REGISTRO DE PERCALÇOS TÉCNICOS	151
ANEXO I - Exemplos Visuais das Tipologias de Cadeiras de Rodas.....	155

1 INTRODUÇÃO

Esta dissertação apresenta a otimização e implementação de um sistema assistivo de baixo custo voltado à identificação de cadeiras de rodas, fundamentado no emprego da arquitetura de aprendizado profundo (do termo em inglês *Deep Learning*) YOLO (do termo em inglês *You Only Look Once*) e na plataforma de *hardware Raspberry Pi*. A pesquisa busca integrar os avanços da visão computacional às demandas de acessibilidade no ambiente construído, visando propor uma solução tecnológica acessível que contribua para a autonomia e para a inclusão social de pessoas com deficiência física por meio da detecção automatizada de seus dispositivos de mobilidade.

1.1 CONTEXTUALIZAÇÃO

A promoção da acessibilidade consolidou-se como um desafio global e multidimensional, impulsionado pela persistência de desigualdades estruturais no acesso a espaços urbanos, pela insuficiência de políticas públicas efetivas e pela lenta incorporação de inovações tecnológicas aos sistemas de mobilidade e inclusão. Tal desafio ultrapassa a esfera técnica ou infraestrutural, constituindo-se como um imperativo ético, jurídico e social intrinsecamente relacionado ao exercício pleno da cidadania e à consolidação de uma sociedade inclusiva (Poojary, 2025). No âmbito das pessoas com deficiência, em especial aquelas que apresentam mobilidade reduzida ou fazem uso de cadeira de rodas, ainda se observam barreiras de diferentes naturezas: físicas, informacionais e atitudinais, que restringem de forma expressiva a autonomia e o exercício pleno da cidadania. As barreiras físicas referem-se a obstáculos concretos do ambiente, como ausência de rampas, calçadas irregulares, portas estreitas e sanitários inadequados. As barreiras informacionais relacionam-se à falta de comunicação acessível e sinalização apropriada. As barreiras atitudinais manifestam-se em preconceitos e estigmas sociais que reforçam a exclusão. Em conjunto, tais limitações perpetuam um modelo urbano excludente, que negligencia a diversidade humana e compromete a participação equitativa nos espaços de convivência, educação e trabalho (Cruz *et al.*, 2023).

Sob a perspectiva social, a escassez de infraestrutura acessível e a reduzida disseminação de tecnologias assistivas em larga escala revelam um quadro estrutural

de exclusão que atinge grande parcela da população com deficiência, especialmente em países em desenvolvimento como o Brasil (Sakellariou; Rotarou, 2017). A falta de soluções tecnológicas adaptadas à realidade econômica da população de baixa renda amplia as desigualdades e perpetua a dependência de familiares ou terceiros para a realização de atividades cotidianas. Essa condição opera em dupla dimensão: limita a autonomia funcional e gera repercussões psicológicas profundas, expressas em sentimento de impotência, frustração e diminuição da autoestima (Cruz *et al.*, 2023; Poojary, 2025).

O modo como a sociedade compreende a deficiência influenciou, ao longo da história, as formas de enfrentamento dessas barreiras. O modelo médico, predominante até a metade do século XX, entendia a deficiência como uma anomalia individual a ser tratada ou corrigida (Zaks, 2024), até que o surgimento do modelo social reformulou esse entendimento, deslocando o foco do indivíduo para as barreiras criadas pelo ambiente. Mais recentemente, a *Classificação Internacional de Funcionalidade, Incapacidade e Saúde* (CIF), proposta pela Organização Mundial da Saúde, consolidou o modelo biopsicossocial, que considera a deficiência como resultado da interação entre as limitações pessoais e os fatores contextuais (Lutte *et al.*, 2024). Essa visão ampliada fundamenta as políticas públicas de inclusão e justifica a necessidade de abordagens tecnológicas voltadas à eliminação de barreiras e à promoção da autonomia.

A inclusão das pessoas com deficiência está amparada por instrumentos normativos nacionais e internacionais, como a Convenção sobre os Direitos das Pessoas com Deficiência, adotada pela Organização das Nações Unidas em 2006 e incorporada ao ordenamento jurídico brasileiro com status de emenda constitucional. Nesse contexto, as tecnologias assistivas emergem como um campo estratégico, capaz de integrar o desenvolvimento científico e tecnológico a objetivos sociais, promovendo a equidade e a cidadania. Conforme evidenciam Navas-Bonilla *et al.* (2025), dispositivos móveis, ferramentas digitais e sistemas inteligentes podem ampliar a participação social e reduzir as desigualdades, desde que concebidos com enfoque inclusivo e centrado nas necessidades do usuário.

Esses instrumentos normativos encontram concretização técnica nas normas brasileiras que padronizam especificamente as tipologias e requisitos para cadeiras de rodas, fornecendo base técnica para soluções assistivas inteligentes. A ABNT NBR ISO 9999:2022 (Tecnologia assistiva: Classificação e terminologia) classifica

tecnologias assistivas, incluindo cadeiras manuais (código 12 22) e motorizadas (12 23), enquanto a ABNT NBR ISO 7176-15:2009 (Cadeiras de rodas, Parte 15: Requisitos de divulgação de informação, documentação e identificação) estabelece requisitos para documentação, identificação e divulgação de dimensões e massa máxima, e a ABNT NBR ISO 7176-1:2018 (Cadeiras de rodas, Parte 1: Determinação da estabilidade estática) define testes de estabilidade estática para diferentes configurações. Complementarmente, a ABNT NBR 9050:2020 (Acessibilidade a edificações, mobiliário, espaços e equipamentos urbanos) orienta dimensões para pessoas em cadeiras de rodas em rotas acessíveis. Esses padrões subsidiam a construção de bancos de imagens diversificados, essenciais para o treinamento de modelos de detecção que distinguem variações tipológicas e promovem adaptações ambientais precisas.

Nesse cenário normativo-tecnológico, o avanço da inteligência artificial (IA) e da visão computacional surge como ferramenta estratégica para operacionalizar essas especificações técnicas. A possibilidade de interpretar imagens e reconhecer padrões em tempo real por meio de redes neurais convolucionais (do termo em inglês *Convolutional Neural Networks, CNNs*) abriu novas perspectivas para o desenvolvimento de sistemas assistivos inteligentes. Entre os algoritmos de destaque, a rede YOLO tem se consolidado como uma das arquiteturas mais eficientes para detecção de objetos, graças à sua capacidade de processar imagens em tempo real e à sua adaptabilidade a diferentes tipos de *hardware*. Estudos recentes demonstram a robustez das versões *Scaled-YOLOv4*, *YOLOv5* e *YOLOv7* mesmo quando aplicadas em bases de dados reduzidas e em dispositivos com limitações de processamento (Ferrante *et al.* 2024). Essa característica torna viável sua implementação em plataformas de baixo custo, como o *Raspberry Pi*, microcomputador criado pela *Raspberry Pi Foundation* que, com placas historicamente comercializadas a partir de cerca de 35 dólares em seus modelos de entrada, arquitetura compacta baseada em processadores ARM (do termo em inglês *Advanced RISC Machines*) e compatibilidade com sistemas operacionais de código aberto, permite a execução de algoritmos complexos de aprendizado de máquina (Paiva; Moreira, 2014). Tal convergência entre algoritmos eficientes e *hardware* acessível viabiliza o desenvolvimento de soluções assistivas portáteis, replicáveis e economicamente viáveis.

No cenário brasileiro, a aplicação de tecnologias assistivas continua sendo um campo em consolidação, limitado pela escassez de investimentos e pela carência de soluções adaptadas à realidade local. Segundo a Pesquisa Nacional de Saúde (PNS), realizada pelo Instituto Brasileiro de Geografia e Estatística (IBGE, 2024) cerca de 7,8 milhões de pessoas apresentam deficiência física nos membros inferiores, o que impacta diretamente sua autonomia e integração social. Apesar dessa magnitude, observa-se uma lacuna significativa na produção científica voltada à aplicação prática de sistemas inteligentes de detecção e automação para a acessibilidade.

A hipótese norteadora desta pesquisa é que é possível desenvolver um sistema assistivo de baixo custo, baseado na rede *YOLO* e implementado em um *Raspberry Pi*, capaz de detectar e classificar tipologicamente cadeiras de rodas em tempo real, fornecendo subsídios técnicos para a futura integração a sistemas de automação no ambiente construído, como portas automáticas, elevadores e dispositivos domésticos inteligentes. Pressupõe-se, ainda, que o reconhecimento da tipologia específica do equipamento de mobilidade pode, em implementações futuras, orientar adaptações mais precisas do ambiente às necessidades de cada usuário, promovendo uma experiência de inclusão mais efetiva e personalizada. Diferente de sistemas que monitoram indivíduos, esta proposta foca exclusivamente na detecção do equipamento assistivo, garantindo a privacidade dos usuários através de um processamento de dados volátil, no qual as imagens capturadas são descartadas imediatamente após a análise, sem armazenamento ou identificação pessoal.

Ao propor uma ferramenta acessível e tecnicamente eficiente, esta pesquisa alinha-se aos Objetivos de Desenvolvimento Sustentável (ODS) da Agenda 2030, especialmente ao ODS 11, Cidades e Comunidades Sustentáveis. A implementação de sistemas inteligentes de baixo custo representa um compromisso ético com a equidade, a autonomia e a dignidade humana, pilares fundamentais de uma sociedade verdadeiramente acessível.

1.2 PROBLEMA DE PESQUISA

Promover a acessibilidade no ambiente construído exige mais do que formulações teóricas; é preciso que as soluções tecnológicas se mostrem viáveis em plataformas acessíveis, de baixo custo e passíveis de replicação. Nesse contexto, o problema central desta pesquisa nasce da dificuldade em transpor algoritmos de visão

computacional de alta performance, em especial a arquitetura de rede neural *YOLO*, para um *hardware* tão compacto e limitado quanto o *Raspberry Pi*, assegurando que a detecção e a classificação tipológica de cadeiras de rodas aconteçam de forma eficiente e em tempo real, dentro das restrições de processamento próprias de dispositivos embarcados.

Soma-se a esse desafio a necessidade de estruturar o treinamento do modelo a partir de uma base de imagens que reflita a diversidade tipológica real das cadeiras de rodas. Estudos recentes que empregam arquiteturas *YOLO* para detecção de pessoas com mobilidade reduzida, como o trabalho de Adorno *et al.* (2023), que utiliza o *dataset MobilityAids* com *YOLOv5* alcançando acurácia de 99%, e a pesquisa de Alruwaili *et al.* (2024), que compara *YOLOv5*, *YOLOv7* e *YOLOv8* obtendo precisão de 90,7%, tratam a cadeira de rodas predominantemente como uma classe única. Embora Adorno *et al.* (2023) contemplem cinco classes distintas, cadeira de rodas; cadeira auxiliada; andador; pessoa e muletas, e Alruwaili *et al.* (2024) diferenciem entre “*wheelchair*” (cadeira de rodas) e “*push_wheelchair*” (cadeira de rodas “empurrada”), ambos os trabalhos voltam-se à identificação do cadeirante e à situação de uso do equipamento, sem distinguir as características tipológicas intrínsecas da cadeira de rodas em si, tais como cadeiras manuais, motorizadas, esportivas ou monoblocos. Essa lacuna metodológica pode comprometer tanto a acurácia da detecção quanto a capacidade do sistema em reconhecer diferentes categorias de equipamento de mobilidade, especialmente em contextos nos quais a tipologia específica da cadeira influencia as adaptações necessárias no ambiente. A originalidade desta pesquisa reside justamente em superar essa lacuna. Enquanto os trabalhos correlatos tratam a cadeira de rodas como objeto único ou, no máximo, distinguem situações de uso, como o cadeirante autônomo em contraste com a cadeira empurrada por terceiro, esta dissertação propõe e operacionaliza uma classificação tipológica em onze categorias funcionais, ancorada nas normas ISO 9999:2022 e ABNT NBR 9050:2020 e em catálogos de quatro fabricantes nacionais e internacionais, e estrutura a detecção computacional em torno de tipologias morfologicamente distintas, condição necessária para que adaptações ambientais futuras possam responder com precisão à diversidade real dos equipamentos de mobilidade.

Diante desse cenário, o estudo busca responder de que maneira a combinação entre uma base de imagens orientada pela tipologia de cadeiras de rodas e a

otimização de modelos *YOLO* para arquitetura *ARM* pode viabilizar um sistema assistivo de baixo custo capaz de fornecer subsídios técnicos para futuras aplicações de automação no ambiente construído. A hipótese operacional sustenta que o treinamento fundamentado na diversidade tipológica, aliado às técnicas de conversão e quantização para dispositivos embarcados, permite alcançar um equilíbrio satisfatório entre precisão de detecção e velocidade de inferência, representando uma alternativa acessível e escalável para o *design* universal.

O objetivo geral desta dissertação consiste em desenvolver e avaliar um sistema de visão computacional de baixo custo, baseado na rede neural *YOLO* e implementado em *Raspberry Pi*, voltado à detecção e à classificação tipológica de cadeiras de rodas em tempo real. Os objetivos específicos compreendem: (1) investigar e sistematizar as classificações nacionais e internacionais de cadeiras de rodas, de modo a fundamentar a construção de um banco de imagens representativo da diversidade tipológica existente; (2) treinar e validar os modelos *YOLOv5* e *YOLOv8* com um banco de dados diversificado que contemple diferentes tipologias, marcas e condições de oclusão; (3) realizar a análise comparativa entre as versões, avaliando o equilíbrio entre precisão métrica e velocidade de processamento em *hardware* com recursos limitados; e (4) implementar e otimizar o modelo no dispositivo embarcado *Raspberry Pi*, avaliando seu desempenho por meio de testes controlados de inferência, e apontando a validação em cenários reais de uso e a integração como gatilho de automação em elementos do ambiente construído como direcionamentos para trabalhos futuros.

1.3 ESTRUTURA DO TRABALHO

Esta dissertação está organizada em cinco capítulos, descritos a seguir.

O Capítulo 1, Introdução, apresenta o objeto de estudo, o problema de pesquisa e a hipótese operacional. Define-se aqui o objetivo geral e os objetivos específicos, estabelecendo o escopo da investigação voltada para a detecção automatizada e classificação tipológica de cadeiras de rodas como subsídio técnico para acessibilidade no ambiente construído.

O Capítulo 2, Fundamentação Teórica, organiza-se em seis eixos. Inicialmente, abordam-se as Tecnologias Assistivas e a mobilidade acessível, incluindo conceitos, classificações e a tipologia de cadeiras de rodas. Em seguida, apresenta-se a

evolução da visão computacional para detecção de objetos, dos métodos clássicos às redes convolucionais, e as métricas de desempenho associadas. Discute-se então a família *YOLO* e suas aplicações em ambientes reais, com atenção ao uso em dispositivos de baixa potência. Os eixos seguintes tratam dos processos de treinamento de modelos de detecção, da implementação em *hardware* embarcado, com ênfase no *Raspberry Pi*, em estratégias de conversão e quantização e no desempenho esperado em *edge computing*, e por fim, dos trabalhos relacionados em detecção de pessoas e cadeirantes, sistemas assistivos com reconhecimento visual e aplicações em ambientes públicos.

O Capítulo 3, Metodologia, detalha o delineamento da pesquisa, o objeto de estudo e o contexto de aplicação, descrevendo a constituição do *corpus* de imagens e a elaboração do quadro de classificação tipológica de cadeiras de rodas. Apresentam-se o protocolo de anotação, a divisão dos dados em treinamento, validação e teste, as configurações dos modelos, hiperparâmetros e critérios de avaliação. São descritas a implementação no dispositivo embarcado, incluindo a plataforma, a configuração do ambiente, a conversão dos modelos para arquitetura *ARM* e os protocolos de teste no *Raspberry Pi* e o sistema operacional embarcado, organizado em arquitetura modular composta por um módulo comum e três aplicativos: um detector especialista com registro de eventos, um detector genérico para validação comparativa e um coletor para retreinamento iterativo. O capítulo encerra-se com a viabilidade operacional em campo, a análise dos resultados e a explicitação de limitações e premissas.

O Capítulo 4, Resultados, apresenta inicialmente a caracterização do *corpus* de treinamento e, em seguida, os resultados do treinamento comparativo dos quatro modelos avaliados, seguidos da análise do desempenho por classe tipológica. Reporta-se o desempenho do modelo no dispositivo embarcado e a operação sustentada em campo, encerrando-se com uma discussão crítica acerca do atingimento parcial dos critérios estabelecidos pela hipótese operacional e com a síntese dos resultados que prepara a transição para as considerações finais.

O Capítulo 5, Conclusão, sintetiza as contribuições científicas e técnicas da pesquisa, retomando o atingimento da hipótese operacional e a viabilidade da detecção tipológica como subsídio para automação ambiental. Discutem-se as implicações para o Ambiente Construído, as limitações metodológicas, as

recomendações para validação em cenários reais e as perspectivas de escalonamento da solução sob a ótica da segurança de dados e do *design* universal.

Os quatro objetivos específicos formulados na Seção 1.2 são tratados ao longo do trabalho da seguinte forma. O objetivo específico (1), referente à sistematização das classificações nacionais e internacionais de cadeiras de rodas e à fundamentação do banco de imagens, é desenvolvido na Seção 2.1.3 e operacionalizado na Seção 3.4, que consolida o Quadro 5. O objetivo específico (2), de treinar e validar os modelos *YOLOv5* e *YOLOv8* com banco diversificado em tipologias, marcas e condições de oclusão, é abordado nas Seções 3.5 e 3.6 quanto ao delineamento experimental e na Seção 4.2 quanto aos resultados quantitativos. O objetivo específico (3), referente à análise comparativa entre as versões avaliadas e ao equilíbrio entre precisão e velocidade de processamento, é desenvolvido nas Seções 4.2 e 4.3, com aprofundamento na Seção 4.4 quanto ao desempenho no dispositivo embarcado. O objetivo específico (4), de implementar e otimizar o modelo no *Raspberry Pi*, é detalhado nas Seções 3.7, 3.8 e 3.9 e validado experimentalmente nas Seções 4.4 e 4.5; os direcionamentos para trabalhos futuros nele indicados, relativos à validação em cenários reais e à integração como gatilho de automação no ambiente construído, são retomados no Capítulo 5.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo estabelece a fundamentação teórica necessária para a compreensão da convergência entre Tecnologias Assistivas e Visão Computacional. São discutidos os marcos teóricos da mobilidade acessível, a evolução das redes neurais convolucionais com ênfase na arquitetura YOLO e as estratégias de implementação em *hardware* embarcado, provendo o embasamento teórico para o desenvolvimento do sistema assistivo proposto.

2.1 TECNOLOGIAS ASSISTIVAS E MOBILIDADE ACESSÍVEL

A Tecnologia Assistiva (TA) é consolidada no Brasil como uma área do conhecimento de caráter interdisciplinar, compreendendo recursos, estratégias e serviços que visam promover a funcionalidade e a participação de pessoas com deficiência ou mobilidade reduzida. Sob a ótica do Ambiente Construído, a TA deixa de ser um dispositivo isolado para se tornar um elemento de mediação entre o indivíduo e o espaço, garantindo autonomia e independência por meio da mitigação de barreiras físicas e informacionais (Brasil, 2009).

2.1.1 Conceitos e classificações de tecnologia assistiva

O conceito de TA consolidou-se internacionalmente como qualquer item, equipamento ou sistema de produtos, seja ele adquirido comercialmente, modificado ou personalizado, utilizado para aumentar, manter ou melhorar as capacidades funcionais de indivíduos com deficiência (Alper; Raharinirina, 2006). Esta definição é reforçada pela *Classificação Internacional de Funcionalidade, Incapacidade e Saúde* (CIF), que desloca o entendimento da deficiência do campo puramente biológico para o modelo biopsicossocial, no qual a funcionalidade é o resultado da interação entre as condições de saúde e os fatores contextuais e ambientais (Lutte *et al.*, 2024).

No Brasil, o Comitê de Ajudas Técnicas (CAT) institucionalizou essa visão ao propor a transição do “Paradigma do Déficit Individual” para o “Paradigma Tecnológico”, em que a tecnologia atua na adaptação do meio para o sujeito (Brasil, 2009). Dentro da taxonomia proposta pelo CAT, este trabalho insere-se na intersecção entre as categorias de “Auxílios de Mobilidade” e “Sistemas de Controle de Ambiente”.

Enquanto a primeira abrange o equipamento que viabiliza o deslocamento, como a cadeira de rodas, a segunda utiliza a visão computacional para automatizar a identificação de demandas de acesso e comandar a infraestrutura ao redor do usuário. O Quadro 1 mostra a relação entre o modelo biopsicossocial e o paradigma tecnológico.

Quadro 1 - Relação entre o modelo biopsicossocial e o paradigma tecnológico da TA

Critério de Comparação	Modelo Biopsicossocial (CIF/ Lutte <i>et al.</i>, 2024)	Paradigma Tecnológico (CAT/ Brasil, 2009)
Foco Central	Estruturar a compreensão da funcionalidade humana como interação entre funções/estruturas do corpo, atividades e participação, sempre em relação a fatores ambientais e pessoais.	Estruturar a Tecnologia Assistiva como área do conhecimento (produtos, recursos, metodologias, estratégias, práticas e serviços) destinada a promover funcionalidade, atividade e participação.
Visão da Deficiência	Deficiência entendida como resultado da interação entre impairments, limitações de atividade e restrições de participação em contextos específicos; a deficiência é inerentemente relacional e contextual, não apenas lesão.	Deficiência vista principalmente como condição que pode gerar incapacidade e desvantagem, a ser prevenida, compensada, aliviada ou neutralizada por produtos e serviços de TA, aproximando-se de um modelo social-tecnológico e ecológico.
Objetivo Primordial	Oferecer uma linguagem comum e um modelo científico para descrever saúde, funcionalidade e deficiência, subsidiando avaliação, planejamento de intervenções, políticas e pesquisa em perspectiva biopsicossocial.	Promover autonomia, independência, qualidade de vida e inclusão social por meio da ampliação do desempenho de atividades e da participação de pessoas com deficiência, incapacidades ou mobilidade reduzida.
Papel da Tecnologia	Tecnologia aparece como um entre vários fatores ambientais, podendo ser barreira ou facilitador; o foco é descrever seu impacto na performance e na participação, não a classificar em si.	Tecnologia é o núcleo do paradigma: TA é definida, classificada (ISO 9999, <i>HEART</i> , classificações nacionais) e organizada em sistemas de prestação de serviços,

		financiamento, avaliação e desenvolvimento, sendo vista como meio central para remover barreiras sistêmicas e ampliar oportunidades.
Modelo de intervenção subjacente	Incentiva intervenções que combinem cuidados médicos, reabilitação e modificações ambientais, incluindo adaptações arquitetônicas, apoio social e uso de TA, para reduzir restrições de participação.	Propõe transição de um “paradigma do déficit individual” (centrado no paciente e na reabilitação médica) para um “paradigma tecnológico-ecológico” (centrado em sistemas de recursos de TA, acesso a ferramentas, informação e treinamento, guiado pelas necessidades do usuário/consumidor).

Fonte: Adaptado de Brasil (2009) e Lutte *et al.* (2024).

O Quadro 1 sintetiza a convergência entre essas perspectivas, demonstrando como o entendimento da deficiência como um fenômeno relacional, resultante da interação entre condições do corpo, atividades, participação e contexto, fundamenta a intervenção ativa no espaço. Enquanto o modelo biopsicossocial oferece a base teórica para identificar as barreiras e facilitadores na relação entre sujeito e ambiente, o paradigma tecnológico organiza a estrutura para que a Tecnologia Assistiva atue de forma ecológica, transformando o ambiente construído em um sistema responsivo às necessidades de funcionalidade e participação das pessoas com deficiência.

2.1.2 Demandas de mobilidade e identificação de cadeirantes

A mobilidade acessível depende da capacidade do ambiente construído em reconhecer demandas específicas de forma dinâmica. Conforme preconizado pelo CAT (Brasil, 2009), a eficácia de uma TA está ligada à promoção da funcionalidade no espaço. Nesse sentido, a detecção automatizada de cadeiras de rodas configura-se como um gatilho tecnológico que antecipa a necessidade de adaptação de sistemas físicos, permitindo que portas, elevadores e sinalizações inteligentes respondam de forma autônoma à presença do equipamento.

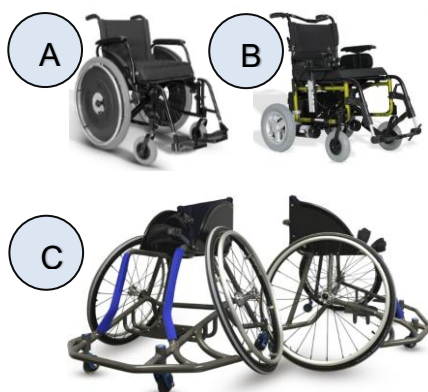
A escolha técnica pelo reconhecimento exclusivo da cadeira de rodas, em detrimento da imagem do usuário, fundamenta-se na premissa ética de que a

tecnologia deve servir à inclusão sem comprometer a privacidade individual. Ao focar nas características estruturais e morfológicas do dispositivo de mobilidade, o sistema cumpre sua função assistiva respeitando os preceitos de segurança de dados. Complementarmente, a implementação baseia-se no princípio da volatilidade de dados: o processamento da imagem ocorre localmente e em tempo real, sendo a informação descartada imediatamente após a detecção do objeto. Tal abordagem assegura que nenhum registro pessoal ou biométrico seja armazenado ou transmitido, alinhando-se às normas contemporâneas de proteção à privacidade e dignidade da pessoa com deficiência.

2.1.3 Tipologia e Classificações de Cadeiras de Rodas

A compreensão da tecnologia assistiva, incluindo as cadeiras de rodas, vai além de simples recurso funcional, atuando como mediação indispensável para atividades formais e informais de cidadania, participação comunitária e justiça social (Austin *et al.*, 2021). Sua classificação tipológica integra tecnologia assistiva, normatização técnica e direitos, traduzindo necessidades clínicas em parâmetros para projeto e regulação. A diversidade morfológica, de modelos manuais e motorizados a esportivos, como no basquete em cadeira de rodas (Figura 1), exige que profissionais atendam essas nuances, promovendo autonomia em vez de barreiras, com impacto direto na mobilidade, conforto, segurança e inclusão digna.

Figura 1 - A: Cadeiras de rodas manual; B: Cadeira de rodas motorizada e C: Cadeira de rodas esportiva (Basquete)



Fonte: Adaptado de Loja Ortobras e Jumper Equipamentos (2026).

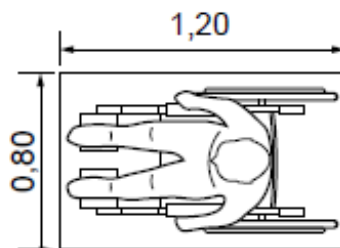
No cenário internacional, a ISO 9999 (2022) estabelece uma taxonomia geral de produtos de tecnologia assistiva que organiza as cadeiras de rodas em grupos e

subgrupos de acordo com a função principal e o modo de propulsão. Nesse enquadramento, cadeiras manuais e motorizadas aparecem como categorias distintas, o que orienta tanto a prescrição profissional quanto o desenvolvimento industrial de linhas de produto com requisitos técnicos diferenciados.

O conjunto de normas da série ISO 7176, adotado no Brasil pela ABNT como NBR ISO 7176 (2009), aprofunda essa distinção ao tratar cadeiras manuais e motorizadas como famílias com requisitos específicos de desempenho, segurança e documentação. Embora a ênfase da série recaia sobre ensaios (estabilidade, freios, dimensões e resistência estrutural), a ABNT NBR ISO 7176-15 ainda estabelece requisitos rigorosos de divulgação de informação e identificação, exigindo que cada equipamento forneça dados claros sobre sua massa total, comprimento e largura. Complementarmente, a segurança operacional é respaldada por normas como a ABNT NBR ISO 7176-1 (2018), que define testes de estabilidade estática, e a ABNT NBR ISO 7176-3:2015 (Cadeiras de rodas, Parte 3: Determinação da eficácia dos freios), assegurando que o equipamento responda adequadamente às diferentes inclinações do ambiente construído.

No contexto brasileiro, a ABNT NBR 9050 (2020), voltada à acessibilidade no ambiente construído, não classifica cadeiras de rodas em si, mas parte de dimensões referenciais para cadeiras manuais, motorizadas e esportivas a fim de definir módulos de referência, áreas de circulação e manobra. Ao incorporar explicitamente a existência de cadeiras esportivas e cambadas, a norma reconhece variações tipológicas relevantes para o dimensionamento de rotas acessíveis, plataformas elevatórias, assentos reservados e demais dispositivos do ambiente. Essa lógica se materializa no conceito de Módulo de Referência (MR), que projeta no plano horizontal a área ocupada por uma pessoa utilizando cadeira de rodas manual, com dimensões de 0,80 m × 1,20 m (Figura 2) que é a métrica base que orienta o desenho de rotas acessíveis e assegura que o espaço comporte a diversidade de modelos disponíveis no mercado.

Figura 2 - Módulo de Referência



Fonte: Extraído da norma ABNT NBR 9050 (2020).

A literatura científica reforça a relevância prática dessas distinções normativas. Bertolaccini *et al.* (2022), Flemmer e Flemmer (2016) e Dalsin *et al.* (2019) convergem ao demonstrar que diferenças estruturais entre tipologias de cadeiras de rodas, como configuração do quadro, posição do eixo traseiro, tipo de pneu e material construtivo, influenciam diretamente a eficiência da propulsão e o esforço muscular do usuário, podendo, no caso do público infantil, repercutir inclusive sobre a construção da identidade e da inclusão social. Tais evidências reforçam que uma classificação tipológica precisa não se limita ao interesse normativo: ela constitui subsídio indispensável para adaptações mais eficazes no ambiente construído.

Em síntese, a tipologia e a classificação das cadeiras de rodas, tal como estruturadas pelas normas ISO 9999 (2022), NBR ISO 7176 (ABNT, 2009) e NBR 9050 (ABNT, 2020), configuram um sistema técnico que organiza o mercado, orienta a prescrição profissional e informa políticas públicas, mas que só cumpre plenamente sua função quando dialoga com a experiência concreta do usuário e com os contextos reais de uso. Ao reconhecer cadeiras manuais, motorizadas, esportivas, pediátricas, monobloco ou dobráveis não apenas como categorias técnicas, mas como respostas diferenciadas a demandas funcionais e sociais, essa classificação torna-se ferramenta para personalizar a tecnologia assistiva e aproximá-la dos princípios do desenho universal (ABNT, 2020; Dalsin *et al.*, 2019; Bertolaccini *et al.*, 2022). Contudo, a literatura evidencia que persistem lacunas entre o ideal normativo e os produtos efetivamente disponíveis, sobretudo no que diz respeito à articulação entre critérios de desempenho, ergonomia e acessibilidade econômica (Flemmer; Flemmer, 2016; Dalsin *et al.*, 2019; Bertolaccini *et al.*, 2022).

O Quadro 2 sintetiza as diferenças centrais entre as normas ISO 9999:2022 e ABNT NBR 9050:2020:

Quadro 2 - Quadro comparativo entre as normas ISO 9999:2022 e a ABNT NBR 9050:2020

Tipologia	ISO 9999:2022	ABNT NBR 9050:2020
Classifica cadeiras de rodas?	Sim. Classificação funcional sistematizada, hierárquica em 3 níveis	Não. Apenas referencia tipologias para dimensionamento
Critério de tipificação	Forma de propulsão/acionamento (manual bimanual, unilateral, motorizada, por acompanhante etc.)	Geometria e dimensões (largura, comprimento, altura) e contexto de uso (esportiva, infantil)
Cadeira Manual	Subclasse 12 22, subdividida em mais de sete divisões por tipo de tração	Tipo único: manual aberta e manual fechada, sem subdivisões
Cadeira Motorizada	Subclasse 12 23, subdividida por posição da tração (dianteira, central, traseira) mais verticalização e escadas	Tipo único: motorizada, sem subdivisões
Cadeira Esportiva	Não tem código próprio	Tipologia específica: esportiva/cambada, com largura mínima de 1,00 m e impacto em vãos de portas
Cadeira Infantil	Não tem código próprio	Dimensões variáveis (menores que as de adulto)
Cadeira com verticalização	Divisão em manual e motorizada	Não menciona
Cadeira para subir escadas	Tem classificação tipológica	Não menciona
Cadeira operada por acompanhante	Tem classificação tipológica	Não menciona
Cadeira acionada pelos pés	Tem classificação tipológica	Não menciona
Cadeira de tração unilateral	Tem classificação tipológica	Não menciona
Scooter	Classifica como diversos veículos motorizados e está fora da classificação de cadeira de rodas	Existe uma citação dentro da norma, mas não há classificação
Acessórios	Existe uma subclasse dedicada de acessórios gerais e de assento	Não aborda
Parâmetros dimensionais	Não fornece dimensões, classifica apenas por função.	Foco central da norma, fornece as dimensões para projetos de espaços.

Fonte: Adaptado das Normas ISO 9999:2022 e ABNT NBR 9050:2020.

Em síntese, a norma ISO 9999:2022 concentra-se na classificação tipológica das cadeiras de rodas, organizando-as em categorias e subcategorias funcionais, enquanto a ABNT NBR 9050:2020 prioriza a definição de dimensões para adaptações no ambiente construído, sem ênfase em uma classificação tipológica propriamente dita.

2.2 VISÃO COMPUTACIONAL PARA DETECÇÃO DE OBJETOS

A visão computacional constitui um campo interdisciplinar que busca dotar máquinas da capacidade de interpretar o mundo visual, transformando imagens e vídeos em informações úteis em diferentes contextos. No âmbito desta pesquisa, ela

é empregada para transformar a captura bruta de sensores ópticos em dados inteligíveis sobre a presença e localização de cadeiras de rodas no ambiente construído. Conforme apontam Fernandes, de Santana e Martins Naves (2019), o campo é plural e em expansão, abrangendo desde métodos clássicos até o aprendizado profundo para interpretar cenas e apoiar decisões seguras. Historicamente, a visão computacional deixou de ser um problema de laboratório para integrar sistemas assistivos reais desde a década de 1980, operando ao lado de outros sensores e algoritmos para garantir a autonomia do usuário (Simpson, 2005).

2.2.1 Evolução da detecção: de métodos clássicos a *deep learning*

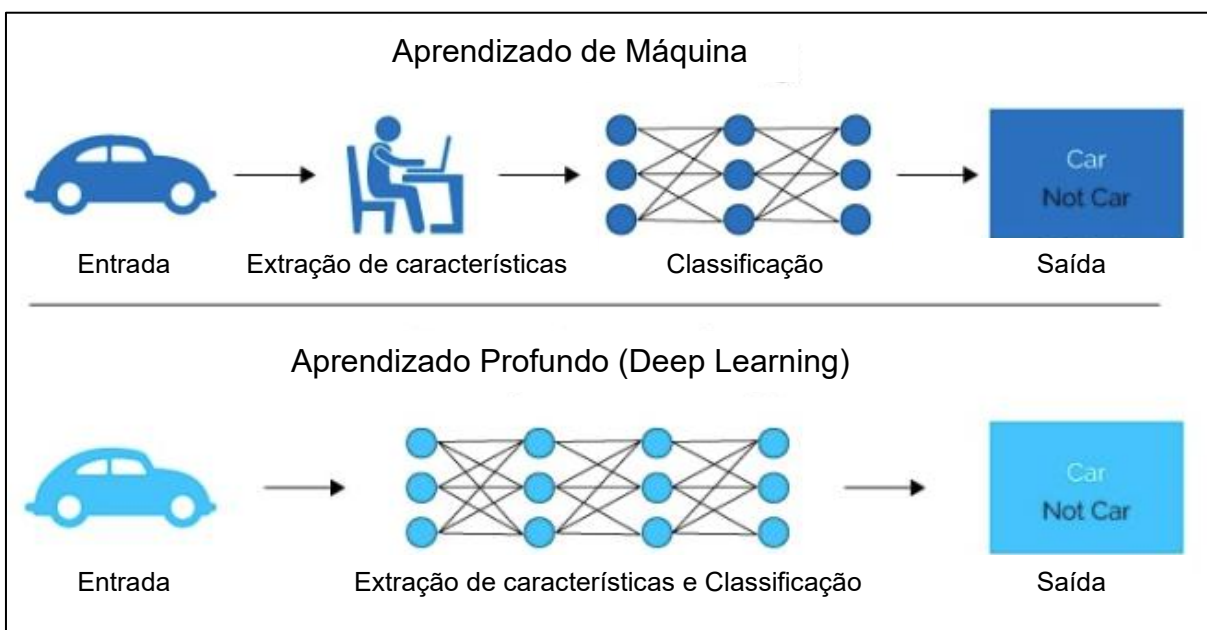
A trajetória da detecção de objetos revela uma transição de métodos baseados em regras geométricas manuais para modelos que aprendem representações diretamente dos dados. Durante as décadas de 1990 e 2000, consolidou-se a abordagem clássica, fundamentada na identificação de pontos característicos (bordas, cantos e contrastes) e no acompanhamento desses pontos em sequências de quadros (Bastos *et al.*, 2018). Esse processo era organizado em *pipelines* computacionais estáveis que envolviam detecção, correlação, estimativa de movimento e refinamento (Bastos *et al.*, 2018). Contudo, tais métodos apresentavam limitações estruturais em ambientes dinâmicos e sujeitos a oclusões.

Três limitações estruturais dos métodos clássicos justificam, em retrospecto, a transição massiva para arquiteturas baseadas em aprendizado profundo. A primeira é a sensibilidade a mudanças de iluminação, sombras e contrastes locais, que altera o conjunto de pontos característicos detectados em uma mesma cena e compromete a estabilidade da correlação entre quadros sucessivos. A segunda é a dependência de engenharia manual de atributos, em que o pesquisador precisa selecionar previamente quais descritores geométricos (bordas, cantos, gradientes, histogramas de orientação) serão extraídos da imagem, decisão que limita a generalização do sistema para classes não previstas no projeto inicial. A terceira é a fragilidade frente a deformações, variações de pose e oclusões parciais dos objetos detectados, condições recorrentes em ambientes reais como o ambiente construído. As redes neurais convolucionais superam essas três limitações ao aprender, por meio da retropropagação do erro, quais características visuais são mais discriminativas para a

tarefa em questão, sem necessidade de engenharia manual de atributos e com maior tolerância a variações de iluminação, oclusão e ponto de vista (Lecun *et al.*, 1998).

A Figura 3 ilustra essa mudança de paradigma, contrastando a necessidade de extração manual de atributos nos métodos tradicionais com a capacidade das Redes Neurais Convolucionais em aprender padrões complexos de forma automática e hierárquica.

Figura 3 - Comparação entre extração de atributos clássica e aprendizado profundo



Fonte: Adaptado de *Software Testing Help* (2024)

A partir da década de 2010, com o advento do algoritmo *YOLO* (*You Only Look Once*), a detecção passou a ser tratada como um único problema de regressão aprendido de ponta a ponta, permitindo um salto de velocidade para o tempo real (Redmon *et al.*, 2016). Para o reconhecimento de cadeiras de rodas, essa evolução é central: o sistema aprende a identificar as características estruturais do equipamento em vez de depender de filtros manuais, adaptando-se com robustez a condições de iluminação variável e oclusões inerentes ao ambiente construído.

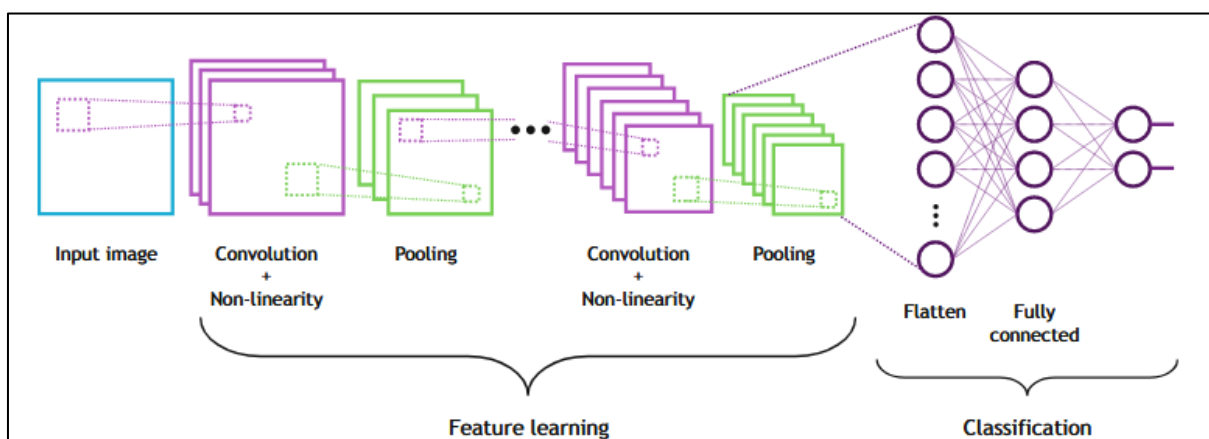
2.2.2 Arquiteturas baseadas em redes convolucionais

As Redes Neurais Convolucionais representam a espinha dorsal do aprendizado profundo aplicado à visão computacional, sendo projetadas para processar dados com topologia de grade, como imagens. A principal motivação para

o uso de arquiteturas convolucionais reside na necessidade de reconhecimento robusto de padrões que podem variar em posição, escala ou iluminação, o que é fundamental para a detecção de cadeiras de rodas em ambientes dinâmicos. Historicamente, o desenvolvimento dessas redes foi inspirado na fisiologia do sistema visual humano, especificamente no funcionamento das células simples e complexas do córtex visual, conceito introduzido por Fukushima (1980) ao propor o *Neocognitron*.

A estrutura fundamental de uma *CNN* é composta por uma sequência hierárquica de camadas que operam de forma integrada para realizar a aprendizagem de características (*feature learning*). A Figura 4 ilustra essa organização típica, evidenciando o percurso desde a entrada da imagem até a classificação final.

Figura 4 - Estrutura básica de uma Rede Neural Convolucional (*CNN*)



Fonte: Vakalopoulou *et al.* (2023)

Conforme ilustrado na Figura 4, o processo inicia-se com as camadas de convolução (*convolution*), responsáveis por aplicar filtros que extraem atributos elementares, como bordas e texturas, normalmente seguidas de uma função de ativação não linear (*ReLU*). Em seguida, as camadas de agrupamento (*pooling*) realizam a subamostragem dos mapas de características, reduzindo a dimensionalidade dos dados e conferindo ao modelo invariância a pequenos deslocamentos e ruídos. Ao final da rede, após sucessivas etapas de extração, os dados são convertidos em um vetor linear (*flatten*) para alimentar as camadas totalmente conectadas (*fully connected*), nas quais ocorre a tomada de decisão sobre a presença e a categoria do objeto detectado. A figura evidencia, ainda, a organização da arquitetura em dois grandes blocos funcionais: a extração de características (*feature learning*), à esquerda, e a classificação (*classification*), à direita, tornando

explícita a transição entre o aprendizado das representações visuais e a decisão final do modelo.

A transição dos métodos clássicos para as *CNNs* modernas, consolidada pelo trabalho de Lecun *et al.* (1998) com a rede *LeNet-5*, eliminou a necessidade de extratores de atributos manuais. Ao permitir que o próprio sistema aprenda, por meio da retropropagação do erro (*backpropagation*), quais características morfológicas são mais relevantes para identificar um dispositivo de mobilidade, as arquiteturas convolucionais garantem uma capacidade de generalização superior em relação aos métodos tradicionais. Essa autonomia justifica a adoção do *deep learning* nesta pesquisa, uma vez que a rede deve ser capaz de distinguir cadeiras de rodas sob condições complexas de oclusão e iluminação no ambiente construído, assegurando a precisão do sistema assistivo.

2.2.3 Métricas de Desempenho em Detecção de Objetos

Para validar a eficácia do sistema assistivo na detecção de cadeiras de rodas, utilizam-se métricas quantitativas que equilibram a precisão e a capacidade de localização espacial do equipamento. Os principais indicadores adotados, fundamentados nos critérios de Redmon *et al.* (2016), são:

- **Precisão (*Precision*):** Refere-se à proporção de acertos entre as detecções realizadas pelo modelo. No sistema proposto, uma alta precisão é vital para evitar acionamentos indevidos de mecanismos de acessibilidade por falsos positivos.
- **Revocação (*Recall*):** Mede a capacidade do sistema em identificar todos os objetos de interesse presentes na cena. No contexto assistivo, assegura que o equipamento de mobilidade não seja ignorado pelo sistema.
- **Interseção sobre União (do termo em inglês *Intersection over Union, IoU*):** Indicador que calcula o grau de sobreposição entre a caixa delimitadora prevista pelo modelo e a caixa real (*ground truth*).
- **precisão média (do termo em inglês *mean Average Precision, mAP*):** Métrica que expressa a precisão média das previsões entre todas as classes. Valores próximos a 1,0 indicam alto desempenho, conforme demonstrado em aplicações assistivas recentes que alcançaram *mAP* de 0,92 na identificação de obstáculos relevantes (Sarker *et al.*, 2022).

A Figura 5 apresenta a representação visual da métrica IoU, na qual se confrontam a caixa delimitadora prevista pelo modelo e a caixa de referência (*ground truth*) anotada manualmente. O indicador resulta da razão entre a área de intersecção (região de sobreposição entre as duas caixas) e a área de união, que corresponde à área total coberta por ambas: quanto maior a sobreposição, mais próximo de 1 é o valor, ao passo que detecções sem coincidência espacial tendem a zero. Na literatura de detecção de objetos, convencionou-se adotar IoU igual ou superior a 0,5 como limiar mínimo para considerar uma detecção válida, critério que governa a precisão da localização espacial dos objetos.

Figura 5 - Representação visual da métrica *IoU* em detecção de objetos



Fonte: Boesch (2024)

A análise integrada dessas quatro métricas é o que permite avaliar, de forma equilibrada, o desempenho de um sistema de detecção em cenários reais. Em sistemas assistivos embarcados como o aqui proposto, esse equilíbrio assume relevância adicional: privilegiar a Precisão minimiza acionamentos indevidos de dispositivos automatizados, enquanto manter Revocação e *IoU* adequados garante que nenhum usuário seja ignorado pelo sistema e que cada cadeira de rodas seja localizada com fidelidade suficiente para orientar adaptações ambientais coerentes com sua tipologia. As quatro métricas aqui apresentadas serão operacionalizadas com valores numéricos na Seção 3.6.2 e discutidas em detalhe nos resultados experimentais reportados no Capítulo 4.

2.3 A FAMÍLIA YOLO E SUAS PRINCIPAIS APLICAÇÕES EM AMBIENTES REAIS

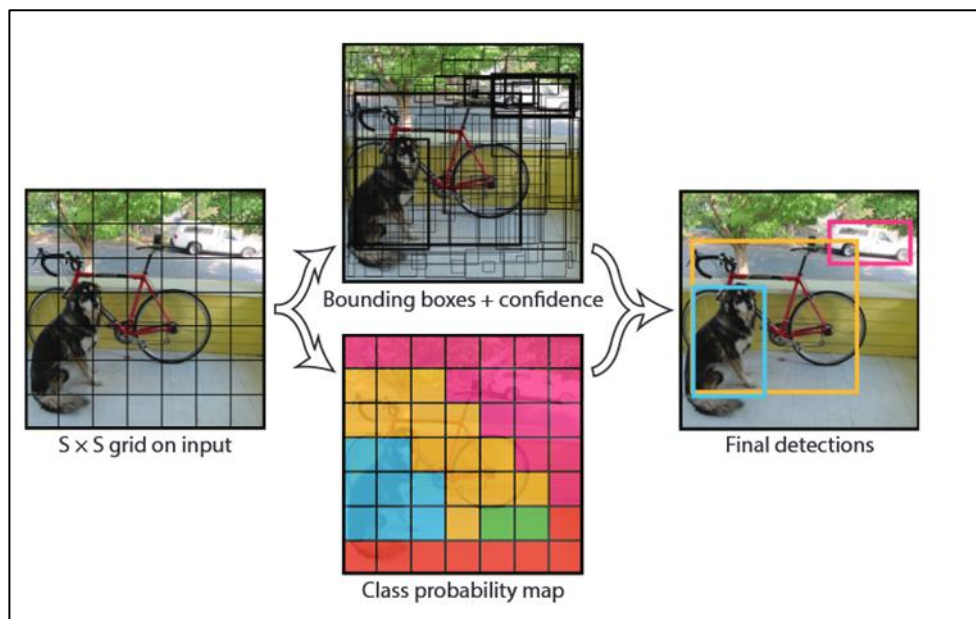
A arquitetura *YOLO* (*You Only Look Once*) consolidou-se como o paradigma predominante para tarefas de detecção de objetos em tempo real, diferenciando-se das abordagens tradicionais de múltiplos estágios. Enquanto detectores de dois estágios, como o *Faster R-CNN*, separam a proposta de região da classificação, o *YOLO* unifica estas etapas em um único passo computacional, o que resulta em baixa latência e elevada eficiência. Conforme destacam Ferrante *et al.* (2024), estudos recentes demonstram a robustez das versões contemporâneas do *YOLO* mesmo quando aplicadas em bases de dados reduzidas e em dispositivos com limitações de processamento. No contexto de tecnologias assistivas aplicadas ao ambiente construído, esta agilidade é determinante para que a detecção de cadeiras de rodas ocorra de forma fluida, permitindo o acionamento de dispositivos de acessibilidade sem interrupções perceptíveis.

2.3.1 Princípios de funcionamento do YOLO

O diferencial técnico do *YOLO* reside em tratar a detecção de objetos como um único problema de regressão, mapeando diretamente os pixels da imagem para as coordenadas das caixas delimitadoras (*bounding boxes*) e probabilidades de classe (Redmon *et al.*, 2016). O algoritmo divide a imagem de entrada em uma grade de $S \times S$ células; se o centro de um objeto incidir em uma célula específica, esta torna-se responsável pela sua detecção. Cada célula prevê um número fixo de caixas delimitadoras e os respectivos níveis de confiança, que refletem a probabilidade de a caixa conter o objeto e a precisão do ajuste ao equipamento assistivo.

A Figura 6 ilustra o sistema de detecção *YOLO*, evidenciando como o processamento unificado conduz da imagem bruta às predições finais por meio de uma única rede convolucional. A imagem de entrada é dividida em uma grade de $S \times S$ células e cada célula prevê, simultaneamente, um conjunto de caixas delimitadoras com seus respectivos níveis de confiança e um mapa de probabilidades de classe. A combinação dessas duas saídas, seguida da supressão das caixas redundantes, produz o conjunto final de detecções, no qual cada objeto aparece delimitado por uma caixa e associado a uma classe.

Figura 6 - O Sistema de Detecção YOLO



Fonte: Redmon *et al.* (2016)

Esse modelo de predição global permite que a rede aprenda o contexto completo da imagem, reduzindo erros de falso positivo em fundos visualmente complexos, isto é, cenas com múltiplos objetos, texturas e elementos de distração que poderiam ser confundidos com o alvo (Redmon et al., 2016).

2.3.2 Comparação entre versões (YOLOv5, YOLOv7 etc.)

Desde o lançamento da versão seminal em 2016, a família YOLO passou por sucessivas evoluções que equilibraram precisão e velocidade. O YOLOv3 consolidou o uso de redes internas mais profundas e a detecção em múltiplas escalas, o que aumentou a eficácia na identificação de objetos menores. Com o advento do YOLOv5, introduziu-se uma estrutura leve e de fácil implementação, tornando-se uma das versões mais utilizadas para *hardware* limitado devido à sua eficiência (Alruwaili *et al.*, 2024). Mais recentemente, o YOLOv8 demonstrou desempenho superior; estudos indicam que esta versão alcança uma precisão de até 0,998 especificamente na detecção de cadeiras de rodas (Ali; Zhang, 2024). O Quadro 3 apresenta uma síntese comparativa das principais características evolutivas da família YOLO, destacando as versões mais relevantes para a presente pesquisa.

Quadro 3 - Evolução e características das versões YOLO

Versão	Inovação Principal	Foco da Aplicação
YOLOv1	Primeira formulação de detecção de objetos em estágio único, transformando a tarefa em um único problema de regressão que prevê caixas e classes em um único “ <i>forward pass</i> ”; introduziu a abordagem de dividir a imagem em grade e prever <i>bounding boxes</i> e probabilidades de classe diretamente, permitindo detecção em tempo real com arquitetura relativamente simples de <i>CNN</i> .	Uso geral em detecção de objetos em tempo real (como veículos, pedestres etc.), servindo como base para adaptações posteriores em cenários específicos; em trabalhos citados no artigo sobre pessoas com deficiência, versões derivadas de <i>YOLOv1</i> são usadas como melhoria sobre métodos <i>R-CNN/Faster R-CNN</i> para detecção de pedestres com deficiência e seus dispositivos de mobilidade em vídeo de vigilância.
YOLOv5	Arquitetura leve baseada em <i>CSPNet</i> , com <i>backbone</i> e cabeça otimizados para melhor equilíbrio entre velocidade e precisão; uso extensivo de técnicas modernas de treinamento (<i>mosaic</i> , <i>CutMix</i> , <i>early stopping</i>) e suporte facilitado a quantização/poda, tornando-se um “padrão de fato” pela facilidade de uso e <i>deploy</i> em dispositivos com recursos limitados.	Aplicações em larga escala e em tempo real com restrições de <i>hardware</i> , como veículos autônomos, agricultura de precisão e inspeção industrial; no contexto de pessoas com deficiência, é usado como um dos <i>baselines</i> principais para detecção de indivíduos e seus auxílios de mobilidade, alcançando boa precisão, mas sendo superado por <i>YOLOv8</i> em métricas como precisão, <i>recall</i> , <i>mAP</i> e tempo de detecção no conjunto de imagens de pessoas com deficiência.
YOLOv8	Revisão da arquitetura com cabeça de detecção atualizada (incluindo camadas <i>C2f</i> , <i>SPPF</i> e melhorias na agregação de características), otimizações de <i>loss</i> e treinamento que aumentam <i>mAP</i> e reduzem tempo de inferência; modelo projetado para maior eficiência computacional mantendo alta acurácia, inclusive em objetos pequenos e cenários complexos.	Aplicações avançadas que exigem alta precisão e alta taxa de quadros em tempo real (saúde, veículos autônomos, monitoramento e vigilância inteligente); no artigo de detecção de pessoas com deficiência, <i>YOLOv8</i> é o modelo proposto, voltado especificamente à detecção e rastreamento de pessoas com diferentes dispositivos (cadeira de rodas, andador, muletas etc.), obtendo melhores resultados que <i>YOLOv5</i> e <i>YOLOv7</i> em precisão global, <i>recall</i> , <i>mAP@0.5</i> , <i>mAP@0.5:0.95</i> e <i>FPS</i> , sendo indicado como opção preferencial para sistemas assistivos e de monitoramento desse público.
Variantes (n / s / m / l / x)	Escalonamento arquitetural que oferece cinco variantes do mesmo modelo, com profundidades e larguras crescentes: <i>nano</i> (n), <i>small</i> (s), <i>medium</i> (m), <i>large</i> (l) e <i>xlarge</i> (x). Cada variante mantém a topologia geral da arquitetura YOLO, mas escala o número de parâmetros, do mais leve (<i>nano</i> , cerca de 3 milhões de parâmetros) ao mais robusto (<i>xlarge</i> , dezenas de milhões), oferecendo ao usuário um espectro explícito de	Para implementação em <i>hardware</i> embarcado de baixo custo, como o <i>Raspberry Pi</i> , as variantes <i>nano</i> e <i>small</i> são as mais indicadas, por oferecerem o melhor compromisso entre velocidade de inferência e precisão. As variantes <i>medium</i> , <i>large</i> e <i>xlarge</i> demandam recursos computacionais incompatíveis com operação em tempo real nesse tipo de plataforma e são, em regra, reservadas a aplicações em <i>GPU</i> dedicada ou em servidor. Nesta dissertação, foram

	escolha entre custo computacional e capacidade representacional, conforme o <i>hardware</i> disponível e os requisitos da aplicação.	avaliadas as variantes <i>nano</i> e <i>small</i> das versões <i>YOLOv5</i> e <i>YOLOv8</i> , totalizando quatro modelos comparados (<i>YOLOv5n</i> , <i>YOLOv5s</i> , <i>YOLOv8n</i> e <i>YOLOv8s</i>).
--	--	--

Fonte: Adaptado de Ali e Zhang (2024) e Alruwaili *et al.* (2024)

A trajetória evolutiva sintetizada no Quadro 3 justifica a opção metodológica desta dissertação por concentrar a análise comparativa nas versões *YOLOv5* e *YOLOv8*, que reúnem, respectivamente, o ponto de referência consolidado na literatura sobre detecção de equipamentos de mobilidade (Adorno *et al.*, 2023; Alruwaili *et al.*, 2024) e o estado da arte em precisão para a tarefa proposta (Ali; Zhang, 2024).

2.3.3 Uso do *YOLO* em dispositivos de baixa potência

A viabilidade de sistemas assistivos acessíveis depende da capacidade de executar modelos de IA em *hardware* de baixo custo, como o *Raspberry Pi*. Criado pela *Raspberry Pi Foundation*, este microcomputador possui arquitetura compacta baseada em processadores *ARM* e compatibilidade com sistemas operacionais de código aberto, permitindo a execução de algoritmos complexos de aprendizado de máquina (Paiva; Moreira, 2014).

A implementação do *YOLO* em dispositivos de baixa potência já demonstrou sucesso em aplicações assistivas para deficientes visuais e de mobilidade, utilizando as capacidades de processamento local para evitar a latência da *nuvem* (Yumang; Banguilan; Veneracion, 2021). Esta abordagem de *Edge Computing* também viabiliza a preservação da privacidade do usuário, conforme tratado na Seção 2.1.2. Tal convergência entre algoritmos eficientes e *hardware* acessível viabiliza o desenvolvimento de soluções portáteis, replicáveis e economicamente viáveis (Paiva; Moreira, 2014).

2.4 PROCESSOS DE TREINAMENTO DE MODELOS DE DETECÇÃO

O treinamento de modelos voltados à detecção de objetos representa uma etapa fundamental para assegurar que a arquitetura neural apresente níveis satisfatórios de acurácia e capacidade de generalização em cenários reais. Esse processo compreende um fluxo estruturado de operações que se inicia na organização dos dados de entrada e culmina no ajuste fino dos parâmetros

algorítmicos. No âmbito das tecnologias assistivas, a eficácia do sistema é diretamente dependente da qualidade do conjunto de dados e da precisão das anotações, elementos que permitem ao modelo distinguir padrões visuais complexos, como os de diferentes modelos de cadeiras de rodas, em ambientes dinâmicos. O algoritmo *YOLO* fundamenta essa tarefa ao tratar a detecção como um problema único de regressão, o que possibilita a otimização conjunta de todo o fluxo de processamento (Redmon *et al.*, 2016).

2.4.1 Preparação e anotação de bases de Imagens

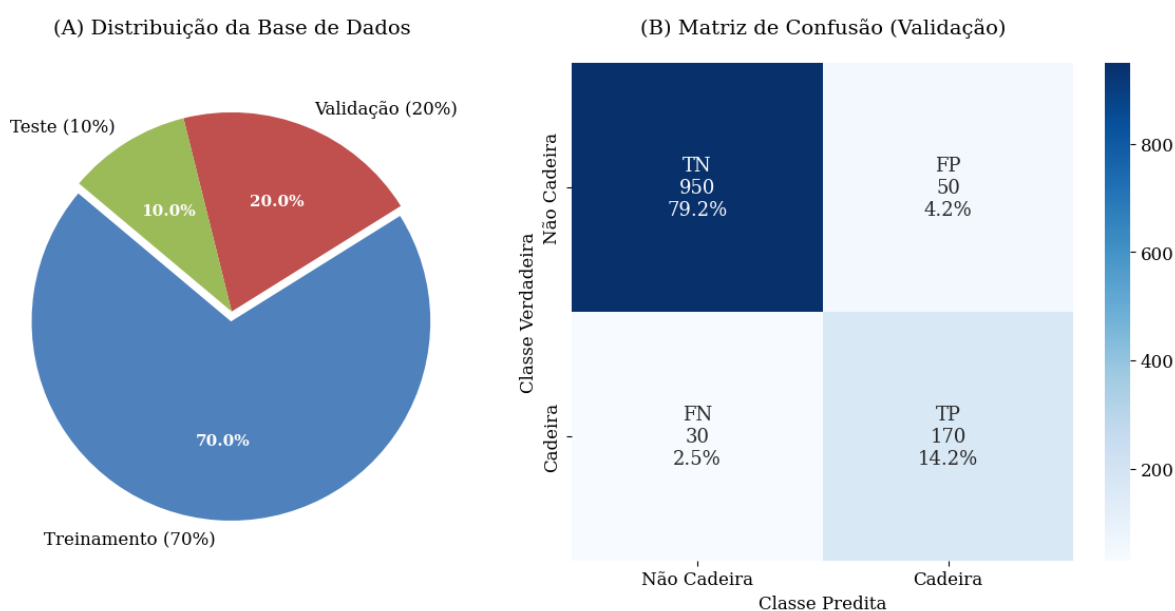
A construção de bases de dados para o treinamento de modelos de detecção de objetos fundamenta-se, predominantemente, na utilização de repositórios públicos e de código aberto, complementados, quando necessário, por buscas direcionadas voltadas à ampliação de classes sub-representadas. Para garantir a padronização técnica e a compatibilidade com as arquiteturas *YOLO*, as imagens são submetidas a critérios mínimos de resolução, comumente 640×480 pixels, antes de serem encaminhadas à anotação manual.

Nesta fase, adota-se o formato *YOLO TXT* para registrar as coordenadas das classes de interesse, no qual cada imagem possui um arquivo de texto homônimo contendo, para cada objeto detectável, o índice numérico da classe e as coordenadas normalizadas da caixa delimitadora (*bounding box*). Essa estrutura acomoda múltiplas classes em diferentes níveis de granularidade, desde problemas binários (presença ou ausência de um objeto), até classificações tipológicas mais refinadas, como a adotada nesta pesquisa, que distingue onze categorias morfofuncionais de cadeiras de rodas. De modo a conferir maior robustez ao modelo, aplicam-se técnicas de *data augmentation* (aumento de dados), tais como espelhamento, rotação e ajustes de brilho e contraste, conforme as recomendações de Ferrante *et al.* (2024) e Yumang, Banguilan e Veneracion (2021). Esses procedimentos são essenciais para reduzir falhas de detecção decorrentes de oclusões parciais ou variações abruptas de iluminação em cenários reais.

2.4.2 Estratégias de divisão de dados (*train/val/test*)

A integridade metodológica da avaliação do modelo exige uma segregação rigorosa do conjunto total de imagens coletadas. Seguindo os preceitos de Marconi e Lakatos (2017), os dados são distribuídos em três subconjuntos distintos para garantir a confiabilidade dos resultados experimentais e a validade científica da detecção. A Figura 7 ilustra tanto a organização quantitativa desta base quanto os resultados preliminares obtidos na fase de validação.

Figura 7 - Distribuição da base de dados e exemplo de matriz de confusão para validação da detecção de cadeiras de rodas



Fonte: Elaborado pelo autor (2026).

Conforme detalhado na Figura 7 (A), o conjunto de dados é dividido em treinamento (70%), validação (20%) e teste (10%). O subconjunto de treinamento é destinado ao ajuste dos pesos da rede neural, enquanto o de validação permite monitorar o desempenho durante o processo de treinamento, sendo fundamental para prevenir o sobreajuste (*overfitting*). O conjunto de teste é reservado exclusivamente para a validação experimental final em ambiente real, assegurando a capacidade de generalização do sistema em dados nunca apresentados ao modelo.

A Figura 7 (B) apresenta um exemplo de matriz de confusão, um dos principais instrumentos utilizados para a coleta e análise técnica nesta pesquisa. Através deste recurso, realiza-se a avaliação quantitativa de desempenho da rede neural YOLO no reconhecimento de cadeiras de rodas, confrontando as predições do modelo com os

dados reais observados. Na matriz, as linhas representam as classes reais e as colunas as classes previstas pelo modelo, de modo que os valores dispostos na diagonal principal correspondem aos acertos (verdadeiros positivos por classe) e os valores fora dela revelam as confusões entre tipologias; quando normalizada, cada célula exprime a proporção, em valores entre 0 e 1 (ou em percentuais), das instâncias de uma classe real atribuídas a cada classe prevista, o que permite identificar diretamente os pares de tipologias mais frequentemente confundidos. Os resultados obtidos na matriz, como os Verdadeiros Positivos (TP) e os Verdadeiros Negativos (TN), fundamentam o cálculo de métricas essenciais como precisão (*precision*), revocação (*recall*) e *F1-score*.

Esses indicadores permitem a análise comparativa com as métricas de referência encontradas em trabalhos correlatos, como as pesquisas de Ferrante *et al.* (2024) e Yumang, Banguilan e Veneracion (2021). A análise final segue a técnica de triangulação de dados, integrando os resultados quantitativos de desempenho com as observações qualitativas, de forma a garantir a credibilidade e a replicabilidade da solução assistiva proposta.

2.4.3 Parâmetros de treinamento e critérios de convergência

O treinamento é executado em plataformas de alto processamento gráfico para viabilizar a convergência do algoritmo dentro de um tempo satisfatório. Os hiperparâmetros são configurados estrategicamente, incluindo o tamanho do lote (*batch size*) entre 16 e 32 imagens e taxas de aprendizado (*learning rate*) adaptativas entre 0.001 e 0.01.

O ciclo de treinamento estende-se tipicamente por 100 a 200 épocas, sendo balizado por funções de perda que visam reduzir simultaneamente os erros de classificação e de localização das caixas delimitadoras, como a Binary Cross Entropy e a IoU Loss, introduzidas em versões posteriores da família YOLO a partir da formulação original (Redmon *et al.*, 2016).

Ao término do treinamento, o modelo é convertido para o formato *TensorFlow Lite* (.*tflite*) para garantir a compatibilidade com a arquitetura *ARM* do *Raspberry Pi*. Essa etapa de otimização é vital para que o sistema assistivo opere com baixo consumo energético e mantenha uma taxa de quadros por segundo (do termo em

inglês *Frames Per Second, FPS*) adequada para aplicações de detecção em tempo real em dispositivos de borda (*edge devices*).

2.5 IMPLEMENTAÇÃO EM *HARDWARE* EMBARCADO

A transição do modelo de detecção de um ambiente de desenvolvimento de alto desempenho para a aplicação em contextos reais exige a integração com plataformas de *hardware* de baixo custo e alta eficiência energética. Esta seção aborda a utilização do *Raspberry Pi* como unidade central de processamento para o sistema assistivo, discutindo suas especificidades técnicas, os processos de otimização de *softwares* necessários e o desempenho esperado para operações de detecção de cadeiras de rodas em tempo real. Ao integrar estes elementos, busca-se validar a viabilidade de uma solução tecnológica que alie rigor técnico à responsabilidade social.

2.5.1 Características e limitações do *Raspberry Pi*

O *Raspberry Pi* é um microcomputador de baixo custo concebido pela *Raspberry Pi Foundation* para democratizar o ensino da computação e viabilizar o desenvolvimento de soluções tecnológicas acessíveis (Paiva; Moreira, 2014). Devido às suas dimensões reduzidas e ao baixo consumo energético, o dispositivo apresenta elevada versatilidade de uso, sendo amplamente empregado em projetos de automação e tecnologia assistiva. Sua arquitetura, baseada em processadores *ARM Cortex-A* de múltiplos núcleos, oferece capacidade de processamento suficiente para a execução de algoritmos de aprendizado de máquina e visão computacional.

Apesar da versatilidade, o *hardware* apresenta limitações de desempenho quando comparado a unidades de processamento convencionais, o que exige um planejamento rigoroso quanto à complexidade dos modelos embarcados. Contudo, segundo Cox *et al.* (2014), experimentos com clusters de *Raspberry Pi* demonstram que o dispositivo mantém estabilidade térmica e resultados satisfatórios em tarefas de processamento paralelo, características essenciais para a operação contínua de sistemas de inteligência artificial (IA). Além disso, a presença de interfaces de entrada e saída de uso geral (do termo em inglês *General Purpose Input/Output, GPIO*) permite o controle direto de sensores e atuadores, facilitando a integração do sistema

de detecção de cadeiras de rodas com mecanismos físicos de acessibilidade, como portas automáticas e elevadores.

2.5.2 Conversão e otimização de modelos (*TensorFlow Lite*, quantização)

A implementação de redes neurais profundas em dispositivos com recursos limitados requer processos de conversão e otimização de modelos para garantir eficiência operacional. O *Raspberry Pi* oferece ampla compatibilidade com bibliotecas de IA de código aberto, como *TensorFlow Lite*, *OpenCV* e *PyTorch ARM*. O processo de conversão para formatos otimizados, como o *.tflite*, permite reduzir de forma expressiva o consumo de memória e energia sem comprometer a acurácia necessária para a detecção.

Técnicas de otimização, como a quantização, são fundamentais para adequar o modelo à arquitetura *ARM*, reduzindo a precisão numérica dos pesos da rede para diminuir a carga computacional. A escolha do algoritmo *YOLO* é estratégica neste contexto, pois, por realizar a detecção em uma única passagem pela rede (*single shot detection*), o sistema reduz significativamente o tempo de inferência e a carga sobre o *hardware* embarcado. Adicionalmente, o uso de *transfer learning* possibilita o treinamento de modelos de alta precisão mesmo com conjuntos de dados reduzidos (Ferrante *et al.*, 2024), facilitando a adaptação do sistema para a detecção específica de cadeiras de rodas em dispositivos de borda.

2.5.3 Desempenho Esperado em *Edge Computing* (FPS, latência, consumo)

A arquitetura de *Edge Computing* (computação de borda) possibilita que o processamento das imagens ocorra localmente no *Raspberry Pi*, garantindo a privacidade dos dados e a redução drástica da latência. Em tecnologias assistivas, a resposta imediata é um requisito crítico para a segurança do usuário, e a execução local permite que o sistema ofereça interações em tempo real com o ambiente. Modelos *YOLO* otimizados para dispositivos de borda são capazes de manter taxas de quadros por segundo (FPS) satisfatórias, aliando baixo custo computacional à eficiência técnica (Ferrante *et al.*, 2024).

A integração entre a detecção em tempo real e os mecanismos de controle automatizado amplia a autonomia de pessoas com mobilidade reduzida, permitindo o

acionamento de sinalizadores ou aberturas automáticas. Dessa forma, o desempenho esperado para o sistema fundamenta-se na capacidade de perceber, analisar e responder ao ambiente de forma integrada e autônoma. Ao unir baixo consumo energético e *hardware* acessível, a solução proposta reafirma o potencial transformador da tecnologia para promover cidades e comunidades mais inclusivas.

2.6 TRABALHOS RELACIONADOS

A literatura recente aponta para um avanço consistente no uso de técnicas de *deep learning*, em especial arquiteturas da família *YOLO*, para detecção de objetos relevantes à mobilidade e à acessibilidade em contextos urbanos e em sistemas assistivos embarcados. Esses estudos convergem com esta dissertação ao articular visão computacional, redes neurais convolucionais e implementação em plataformas computacionais restritas, embora com recortes distintos quanto ao foco na detecção de pessoas, cadeiras de rodas, elementos de infraestrutura urbana ou rotas acessíveis.

Em síntese, identificam-se três eixos principais: (i) detecção de pessoas com mobilidade reduzida e auxiliares de locomoção; (ii) cadeiras de rodas e sistemas assistivos inteligentes com reconhecimento visual embarcado; e (iii) aplicações em ambiente urbano voltadas à avaliação de acessibilidade e ao planejamento de rotas amigáveis para cadeirantes.

2.6.1 Detecção de pessoas e cadeirantes com *deep learning*

Adorno *et al.* (2023) apresentam um sistema de detecção automática de pessoas com mobilidade reduzida utilizando a rede *YOLOv5* aplicada ao conjunto de dados *MobilityAids*, contemplando classes como pedestres, usuários de cadeira de rodas, cadeiras auxiliadas, pessoas com muletas e andadores. Os autores exploram diferentes arquiteturas (*YOLOv5n* e *YOLOv5s*) e estratégias de redução da base de imagens, demonstrando que é possível manter *mAP* médio em torno de 0,99 com acurácias elevadas para cadeiras de rodas (*mAP* superior a 0,98), mesmo com apenas 18% das imagens originais, o que reforça a viabilidade de modelos compactos em cenários com restrições computacionais.

Do ponto de vista metodológico, o trabalho evidencia a importância da variabilidade dos dados em detrimento do mero volume, bem como o impacto da distância do objeto e do desbalanceamento de classes nas métricas de precisão e revocação. Embora o foco esteja na classificação de múltiplas categorias de mobilidade assistida para automatizar o controle de elevadores no Metrô de São Paulo, a abordagem é diretamente comparável à proposta desta dissertação ao empregar YOLO como núcleo de detecção e ao visar o acionamento automático de dispositivos físicos a partir da identificação de equipamentos associados à mobilidade reduzida.

Lecrosnier *et al.* (2020), por sua vez, desenvolvem um sistema de detecção, localização e rastreamento de portas e maçanetas para uma cadeira de rodas elétrica inteligente, combinando YOLOv3, câmera *Intel RealSense* e o algoritmo SORT (do termo em inglês *Simple Online and Realtime Tracking*) para gerar um mapa semântico 3D do entorno imediato do usuário. Embora o objetivo principal seja a navegação assistida da própria cadeira, o estudo demonstra o potencial das redes de detecção em tempo real para reconhecer alvos estruturais do ambiente construído e orientar o acionamento semiautônomo em direção a pontos de interesse, o que se aproxima, em nível conceitual, da ideia desta pesquisa de usar a detecção de cadeiras de rodas como gatilho para automação arquitetônica.

Os trabalhos discutidos nesta subseção evidenciam que arquiteturas YOLO já demonstraram maturidade suficiente para a detecção robusta de cadeiras de rodas e de outros elementos relacionados à mobilidade em cenários variados, fornecendo a base técnica sobre a qual esta dissertação se apoia. A próxima subseção amplia esse panorama, examinando como tais capacidades de detecção têm sido incorporadas a sistemas assistivos embarcados, em que a visão computacional deixa de operar como ferramenta isolada e passa a integrar *pipelines* de percepção, decisão e atuação voltados diretamente ao usuário com deficiência.

2.6.2 Sistemas assistivos com reconhecimento visual

No campo dos sistemas assistivos, Lecrosnier *et al.* (2020) aprofundam, no mesmo trabalho já apresentado na Seção 2.6.1 sob a ótica da detecção, a integração entre visão computacional e mobilidade ao propor um sistema avançado de assistência ao condutor (do termo em inglês *Advanced Driver Assistance System*,

ADAS) embarcado em uma cadeira de rodas, implementado em uma *Nvidia Jetson TX2* e baseado em *YOLOv3* para detecção, estimativa de profundidade e rastreamento de objetos relevantes ao deslocamento. O *pipeline* inclui a criação de um *dataset* específico de portas e maçanetas, o uso de câmeras *RGB-D* e o mapeamento semântico 3D, demonstrando que soluções de detecção em tempo real podem operar sob severas restrições de processamento e energia, o que é altamente pertinente para a adoção de *Raspberry Pi* como unidade embarcada nesta dissertação.

Em linha semelhante, Lecrosnier *et al.* (2021) descreve em outro artigo a arquitetura de uma cadeira de rodas robótica equipada com *YOLOv3*, câmera *RealSense* e rastreamento em 3D, salientando o papel do processamento embarcado e do sistema operacional para robôs (do termo em inglês *Robot Operating System, ROS*) na integração entre percepção visual, planejamento de movimento e controle de motores. Esses trabalhos consolidam um paradigma em que a visão computacional não se limita à detecção de obstáculos, mas atua como camada de percepção de alto nível, capaz de alimentar decisões autônomas sobre o ambiente e sobre a interação com o usuário, modelo conceitual que inspira a utilização da detecção de cadeiras de rodas, nesta dissertação, como interface entre o equipamento assistivo e sistemas de automação predial.

Além dos dispositivos centrados no usuário, revisões recentes sobre sistemas assistivos baseados em visão, como o estudo de Kathiria *et al.* (2024) sobre tecnologias para pessoas com deficiência visual, evidenciam o uso recorrente de arquiteturas *YOLO* e de redes convolucionais em dispositivos portáteis e vestíveis, com forte ênfase na execução em borda para redução de latência e proteção da privacidade. Essa prioridade pelo processamento local e pelo uso de dados voláteis converge diretamente com a opção adotada nesta pesquisa de realizar toda a inferência no *Raspberry Pi* e descartar as imagens imediatamente após a detecção da cadeira de rodas, conforme discutido no Capítulo 1.

Em conjunto, esses estudos delineiam um movimento progressivo da visão computacional, do reconhecimento pontual de objetos para a constituição de camadas de percepção integradas a dispositivos assistivos embarcados, com forte ênfase em processamento local e proteção de dados. A subseção seguinte desloca o foco do dispositivo para o ambiente, analisando aplicações em que a detecção visual é

mobilizada na escala do espaço público e da infraestrutura urbana, dimensão que se aproxima diretamente do escopo arquitetônico desta pesquisa.

2.6.3 Aplicações em ambientes públicos e acessibilidade

Macuácuca *et al.* (2024) investigam a detecção automática de elementos de infraestrutura urbana, postes de energia, iluminação pública, faixas de pedestres e rampas de acessibilidade a partir de imagens terrestres, utilizando a rede YOLOv5 para apoiar o mapeamento e a gestão de espaços públicos. Os resultados indicam *mAP* em torno de 78 a 79% para classes como postes e faixas, mas desempenho inferior (61 a 64%) para elementos de acessibilidade, em função de baixa resolução e variação visual, evidenciando tanto o potencial quanto os desafios da detecção de objetos diretamente relacionados à acessibilidade em ambientes reais.

Em escala urbana, Martínez-Chao *et al.* (2024) propõem um método para avaliação da acessibilidade de rotas de pedestres com foco em usuários de cadeira de rodas, combinando detecção de faixas de pedestres com YOLOv8 em ortoimagens aéreas, modelos digitais de elevação de alta resolução e análise de redes em Sistema de Informação Geográfica (SIG). A abordagem permite classificar travessias como acessíveis ou não acessíveis com base em declividades e descontinuidades geométricas, e calcular rotas “*wheelchair-friendly*” (amigáveis para cadeira de rodas) entre pontos de interesse, demonstrando como técnicas de detecção de objetos podem ser integradas a fluxos de planejamento urbano voltados à mobilidade inclusiva.

A síntese comparativa entre os trabalhos descritos nas subseções anteriores e a proposta desta dissertação permite situar a contribuição original em quatro contrastes objetivos. Adorno *et al.* (2023) detectam pessoas com mobilidade reduzida e seus auxílios, mas tratam a cadeira de rodas como classe única, sem distinguir tipologias funcionais; o desempenho elevado relatado pelos autores é, portanto, indicador de detecção genérica do equipamento, e não de classificação tipológica. Lecrosnier *et al.* (2020, 2021) embarcam visão computacional na própria cadeira de rodas, com câmera RGB-D e mapeamento semântico tridimensional, voltada à navegação autônoma do equipamento, e não à infraestrutura do ambiente, configurando uma topologia de aplicação distinta da que aqui se propõe. Macuácuca *et al.* (2024) e Martínez-Chao *et al.* (2024) operam em escala urbana, sobre

infraestrutura ou rotas pedonais, sem se ocuparem do equipamento de mobilidade em si; o desempenho comparativamente reduzido reportado por Macuácuá *et al.* (2024) para elementos de acessibilidade evidencia a dificuldade de generalização quando o objeto de interesse é minoritário em relação a outras classes do mesmo *dataset*, lacuna que a presente pesquisa busca evitar pela construção de um *corpus* dedicado e pela anotação criteriosa em onze tipologias. Em conjunto, esses contrastes posicionam esta dissertação como, até onde se pôde verificar na literatura revisada, o primeiro trabalho a estruturar a detecção computacional de cadeiras de rodas em torno de tipologias morfológicamente distintas, em *hardware* de baixo custo dimensionado para servir de gatilho de automação no ambiente construído, e sob princípio explícito de volatilidade dos dados em tempo de inferência.

O conjunto desses trabalhos demarca o estado da arte sobre o qual esta dissertação se constrói. As decisões metodológicas que materializam a contribuição aqui anunciada, da constituição do *corpus* à implementação no dispositivo embarcado, são detalhadas no Capítulo 3.

2.7 SÍNTESE E CONSIDERAÇÕES PARCIAIS

Este capítulo reuniu os fundamentos que sustentam a convergência entre tecnologia assistiva e visão computacional perseguida pela pesquisa. Discutiu-se, primeiro, o deslocamento conceitual do modelo médico para o modelo biopsicossocial e para o paradigma tecnológico da tecnologia assistiva, que reposiciona a deficiência como resultado da interação entre o sujeito e o ambiente e fundamenta a atuação sobre o espaço construído; nesse marco, consolidou-se a classificação tipológica das cadeiras de rodas a partir das normas ISO 9999:2022 e ABNT NBR 9050:2020 e dos catálogos de fabricantes. Em seguida, percorreu-se a evolução da detecção de objetos, dos métodos clássicos às redes neurais convolucionais, com as métricas que a avaliam (precisão, revocação, IoU e mAP), e a família YOLO foi caracterizada quanto a seus princípios, à comparação entre versões e à viabilidade em dispositivos de baixa potência. Por fim, examinaram-se os processos de treinamento, a implementação em *hardware* embarcado (com ênfase no *Raspberry Pi*, na conversão e na quantização de modelos e no desempenho esperado em *edge computing*) e os trabalhos relacionados, organizados em três eixos: detecção de pessoas e cadeirantes, sistemas assistivos com reconhecimento visual e aplicações em

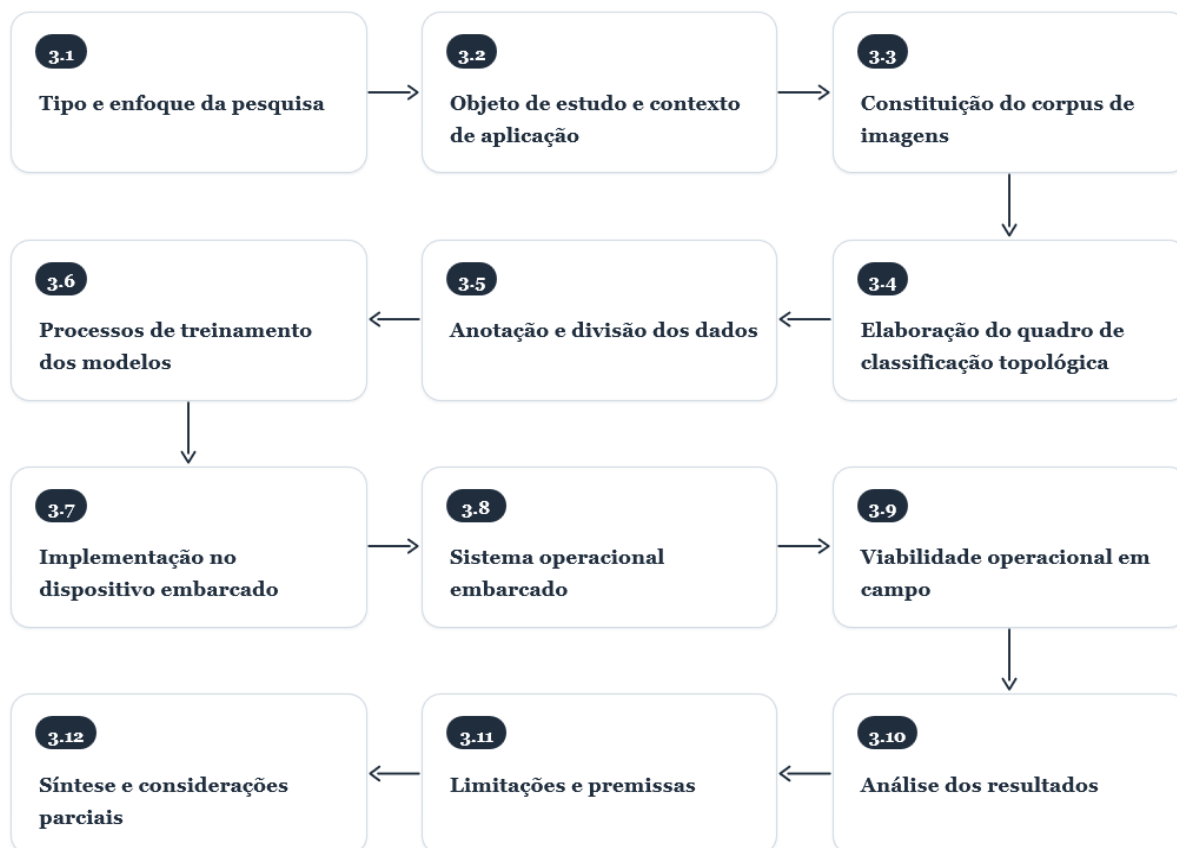
ambientes públicos. O confronto desses trabalhos evidenciou a lacuna que esta dissertação busca preencher: enquanto a literatura trata a cadeira de rodas como classe única ou, no máximo, distingue situações de uso, nenhum dos estudos revisados estrutura a detecção em torno de tipologias morfológicamente distintas, em *hardware* de baixo custo e sob princípio explícito de volatilidade dos dados. É a partir desse diagnóstico que o Capítulo 3 detalha as decisões metodológicas que materializam a proposta.

3 METODOLOGIA

A metodologia adotada neste trabalho visa o desenvolvimento e validação experimental de um sistema assistivo capaz de detectar e classificar tipologicamente cadeiras de rodas em tempo real por meio das redes neurais *YOLOv5* e *YOLOv8* implementadas em um microcomputador *Raspberry Pi*, priorizando o uso de tecnologias de baixo custo e código aberto. O estudo segue uma abordagem aplicada e exploratória, fundamentada no método hipotético-dedutivo, conforme as diretrizes de Marconi e Lakatos (2017), que orientam a formulação de hipóteses, a verificação empírica e a análise sistemática dos resultados obtidos, garantindo reprodutibilidade por meio de protocolos padronizados, códigos-fonte públicos e bancos de dados acessíveis.

A Figura 8 sintetiza, em alto nível, as etapas que estruturam o percurso metodológico desta pesquisa, da constituição do *corpus* à validação do sistema embarcado, e funciona como mapa de leitura para as seções que compõem este capítulo.

Figura 8 - Fluxograma das etapas metodológicas



Fonte: Elaborado pelo autor (2026).

3.1 TIPO E ENFOQUE DA PESQUISA

Esta dissertação é uma pesquisa aplicada de enfoque exploratório. É aplicada por sua finalidade prática: não busca apenas descrever ou interpretar um fenômeno, mas oferecer uma solução técnica concreta para um problema real do ambiente construído, qual seja a indistinção com que os sistemas prediais automatizados hoje reconhecem, ou deixam de reconhecer, as diferentes tipologias de cadeiras de rodas que circulam por seus espaços. É exploratória porque investiga um arranjo pouco documentado na literatura revisada, conforme demonstrado na Seção 2.6, no qual arquiteturas YOLO das famílias YOLOv5 e YOLOv8 são adaptadas para operar em *hardware* embarcado de baixo custo, com vistas à classificação morfológica do equipamento de mobilidade em tempo real.

A condução do trabalho segue o método hipotético-dedutivo, conforme as orientações de Marconi e Lakatos (2017), que organiza o percurso da pesquisa desde a formulação da hipótese até a validação dos resultados. O ponto de partida é a

hipótese operacional enunciada em termos qualitativos na Seção 1.2 e aqui formalizada em critérios quantitativos: um modelo *YOLO* devidamente treinado e otimizado deve alcançar precisão média ($mAP@0.5$) igual ou superior a 0,85 e taxa de inferência (FPS) igual ou superior a 15 quadros por segundo quando executado em um *Raspberry Pi 5*. Essa hipótese é confrontada, em seguida, com a evidência empírica produzida pela experimentação controlada, que cobre o treinamento dos modelos, a conversão para a arquitetura *ARM* e o *benchmark* sistemático do desempenho em diferentes condições de operação. A validação final é quantitativa, ancorada nas métricas de desempenho consolidadas na Seção 3.6.2.

Cabe um esclarecimento adicional sobre a estrutura dessa hipótese. A operação efetiva do sistema sob restrições de baixo custo e de preservação da privacidade do usuário não compõe o conjunto formal de critérios quantitativos, mas atua como premissa transversal do trabalho. Trata-se de uma condição assumida desde o início, que orienta as decisões tomadas em cada etapa metodológica e sem a qual os dois critérios formais perderiam sentido prático. Esse arranjo será retomado de modo explícito na Seção 4.6, no momento da discussão do atingimento dos critérios.

3.2 OBJETO DE ESTUDO E CONTEXTO DE APLICAÇÃO

O objeto de estudo desta pesquisa é um sistema embarcado de visão computacional, projetado para detectar e classificar tipologicamente cadeiras de rodas em tempo real. O sistema apoia-se na classificação de onze categorias funcionais consolidada na Seção 3.4, a partir do diálogo entre as normas ISO 9999:2022 e ABNT NBR 9050:2020 e os catálogos de quatro fabricantes nacionais e internacionais. Toda a inferência ocorre localmente, em uma plataforma *Raspberry Pi 5*, sobre o sinal visual capturado por câmera fixa, sem qualquer transmissão a servidores externos e sem retenção das imagens após a inferência.

O contexto de aplicação privilegiado é o dos ambientes institucionais do ambiente construído, como escolas, centros de reabilitação, terminais de transporte público e edifícios de uso coletivo. Nesses contextos, a presença de usuários de cadeira de rodas é frequente, e as adaptações ambientais que poderiam acolhê-los de modo mais preciso ainda raramente consideram as diferenças entre as tipologias do equipamento. O sistema proposto pretende oferecer à arquitetura desses espaços

um gatilho tecnológico sensível a essa diversidade, alinhado ao paradigma do desenho universal preconizado pela ABNT NBR 9050:2020 e à categoria de Sistemas de Controle de Ambiente prevista na taxonomia do Comitê de Ajudas Técnicas (Brasil, 2009).

3.3 CONSTITUIÇÃO DO *CORPUS* DE IMAGENS

A constituição do *corpus* de imagens utilizado no treinamento e na validação dos modelos foi conduzida em fevereiro de 2026, em fluxo de trabalho local na estação do pesquisador, em conformidade com o princípio de baixo custo e replicabilidade que orienta esta pesquisa. A estratégia adotada combinou a consulta a quatro repositórios públicos amplamente empregados pela comunidade de visão computacional com buscas direcionadas voltadas às tipologias sub-representadas, totalizando 19.452 imagens brutas que, após etapas sucessivas de filtragem por resolução, deduplicação e triagem visual, foram consolidadas em um *corpus* final de exatamente 2.000 imagens.

A primeira fonte consultada foi o *Wheelchair Dataset* (Sum, 2021), disponibilizado na plataforma *Kaggle* sob licença *MIT*, do qual foram incorporadas 1.266 imagens. Embora o repositório original contemplasse anotações em três classes genéricas, optou-se pelo descarte integral das marcações preexistentes e pela reanotação completa das imagens segundo as 11 tipologias da classificação consolidada (Quadro 5), garantindo a homogeneidade do *corpus* diante do nível de detalhamento tipológico proposto. A segunda fonte, *Open Images Dataset V7* (Kuznetsova et al., 2022), publicada pela *Google LLC* sob licença *Creative Commons Attribution 4.0 International*, foi acessada por meio da biblioteca *openimages*, com filtro pela rotulagem "*Wheelchair*"; o repositório retornou somente 131 imagens disponíveis (73 de treinamento, 20 de validação e 38 de teste), abaixo do limite de 500 inicialmente configurado, evidenciando a cobertura limitada da classe nesse banco. A terceira fonte prevista, o *Common Objects in Context* (COCO; Lin et al., 2014), em sua versão de 2017, foi descartada após verificação técnica: a tentativa de filtragem automatizada via *fiftyone* retornou o erro "*Unsupported classes: ['wheelchair']*", uma vez que a categoria não consta entre as 80 classes oficiais do *dataset*, aparecendo apenas incidentalmente como elemento de fundo em imagens de outras classes. A exclusão do COCO foi formalmente registrada e motivou a adoção de buscas direcionadas no

serviço *Bing Image Search* (Microsoft Corporation, 2026), operacionalizadas por meio da biblioteca *icrawler* (Hu, 2017), como estratégia substitutiva. A quarta fonte foi o *MobilityAids Dataset*, do grupo *Autonomous Intelligent Systems* da Universidade de Freiburg (Vasquez et al., 2017), do qual foram extraídas 17.079 imagens brutas a partir do arquivo *Images_RGB.zip* (12 GB), com anotações em formato *YAML*. Um *script Python* percorreu os 13.409 arquivos *.yml* disponíveis, identificando 5.366 imagens contendo cadeiras de rodas e 8.043 sem ocorrência da classe; em razão da elevada redundância entre frames consecutivos extraídos de sequências de vídeo, aplicou-se amostragem temporal de uma a cada cinco imagens nas positivas, resultando em 1.074 exemplares, complementados por 250 imagens negativas selecionadas aleatoriamente (semente 42) para servirem como amostras de fundo, conforme recomendação da documentação oficial do *framework YOLO* (Jocher; Chaurasia; Qiu, 2023).

Como complemento às bases públicas, foram conduzidas duas rodadas de busca direcionada na ferramenta *Bing Image Search*, mediante a biblioteca *icrawler* (*BingImageCrawler*), motivadas pela sub-representação das tipologias pediátrica, bariátrica, postural, *standing* e anfíbia nos repositórios genéricos. A primeira rodada (490 imagens) empregou termos em inglês específicos por tipologia (por exemplo, *manual folding wheelchair*, *standing wheelchair power stand*, *beach wheelchair amphibious sand*); a segunda rodada (486 imagens) utilizou termos em português e variações complementares em inglês (cadeira de rodas postural reclinável, *tilt in space wheelchair seating*, *extra wide bariatric wheelchair*), totalizando 976 imagens adicionais. Tentativas paralelas com as bibliotecas *bing_image_downloader* (incompatível com *Python* 3.13 em virtude da remoção do módulo *imgHDR*), *simple_image_download* (desatualizada, sem retorno de resultados) e *GoogleImageCrawler* (bloqueada pelo *Google*) foram registradas e descartadas por inviabilidade técnica. As imagens obtidas via *Bing* passaram por triagem manual posterior pelo pesquisador para verificação de qualidade e adequação tipológica, em consonância com o protocolo de anotação descrito na Seção 3.5.1.

O Quadro 4 sintetiza as fontes consultadas, os volumes brutos e incorporados e o status de cada repositório no *corpus* final.

Quadro 4 - Fontes de imagens, volumes e status de inclusão no *corpus*

Fonte	Brutas	Selecionadas	Status
<i>Wheelchair Dataset (Kaggle)</i>	1.266	1.266	Incorporado
<i>Open Images V7</i>	131	131	Incorporado
COCO 2017	Não se aplica	0	Descartado
<i>MobilityAids Freiburg</i>	17.079	1.324	Amostrado
<i>Bing Image Search</i> (rodada 1, termos em inglês)	490	490	Incorporado
<i>Bing Image Search</i> (rodada 2, termos em português)	486	486	Incorporado
Subtotal selecionado das fontes	19.452	3.697	Não se aplica
Corpus final após filtragem	Não se aplica	2.000	Não se aplica

Fonte: Elaborado pelo Autor (2026).

A heterogeneidade observada nos volumes incorporados de cada fonte reflete uma lógica de aproveitamento controlado, e não a aplicação de uma cota fixa por repositório. As fontes *Wheelchair Dataset (Kaggle)*, com 1.266 imagens, e *Open Images V7*, com 131 imagens, foram incorporadas integralmente, uma vez que constituem bases temáticas relativamente compactas e todo o conteúdo nelas disponível foi consumido após as etapas de filtragem por resolução, deduplicação e adequação tipológica estabelecidas no protocolo de anotação (Seção 3.5.1). A base *MobilityAids Freiburg*, ao contrário, com 17.079 imagens brutas, foi submetida a uma estratégia de amostragem que selecionou 1.324 imagens consideradas relevantes, segundo o critério de variabilidade tipológica priorizado neste *corpus*. As duas rodadas de busca direcionada na ferramenta *Bing Image Search*, por sua vez, foram dimensionadas *a posteriori* para suprir a sub-representação observada nas tipologias pediátrica, bariátrica, postural, *standing* e anfíbia após a triagem das bases públicas, e por isso operaram em volumes mais modestos (490 imagens com termos em inglês e 486 imagens com termos em português). O alvo agregado de 2.000 imagens efetivamente anotadas, ao qual o *corpus* convergiu após as etapas de filtragem descritas adiante, foi calibrado em equilíbrio com a viabilidade da anotação manual conduzida integralmente pelo pesquisador.

A opção por um *corpus* de escala moderada, porém heterogêneo, fundamenta-se na evidência empírica de que a variabilidade das imagens é mais determinante para o aprendizado do modelo do que o volume bruto. Adorno *et al.* (2023) reduziram o conjunto *MobilityAids* de 17.000 para 3.000 imagens (18% do total original), por meio de amostragem aleatória estratificada e proporcional, e obtiveram *acurácia* de 99%, resultado superior ao alcançado com a base completa. Os autores destacam que

conjuntos compostos por frames altamente similares induzem ao sobreajuste e produzem estimativas otimistas da qualidade do aprendizado, fenômeno observado quando o modelo memoriza padrões repetidos e falha diante de variações reais. A redução proporcionou ainda diminuição de 80% no tempo de treinamento, sem prejuízo das métricas de precisão e revocação (Adorno *et al.*, 2023). A abordagem adotada nesta pesquisa alinha-se a essa orientação ao priorizar a diversidade de fontes, a representatividade tipológica e a qualidade da anotação em detrimento do volume agregado.

Após a consolidação das fontes, todas as 19.452 imagens brutas foram preservadas em diretório de *backup* denominado *banco_completo*, garantindo a auditabilidade integral do processo e possibilitando o reaproveitamento em pesquisas futuras. Em seguida, três etapas sequenciais de filtragem produziram o *corpus* operacional: verificação automatizada da resolução mínima de 640 × 480 pixels, mediante *script Python* com a biblioteca *Pillow*, conforme requisito definido na Seção 2.4.1; remoção de duplicatas por meio de *hash perceptual*; e triagem visual manual, na qual o pesquisador descartou exemplares borrados, com oclusão excessiva, fora do escopo tipológico ou de qualidade insuficiente para anotação. Resolveu-se ainda um conflito de nomenclatura entre as imagens das duas rodadas do *Bing* por meio de um script de renomeação que adicionou prefixos identificadores da fonte e da tipologia. Ao término do processo, o *corpus* final foi consolidado em exatamente 2.000 imagens, renomeadas sequencialmente de 0000 a 1999 e armazenadas em diretório único, prontas para a etapa de anotação descrita na Seção 3.5. A relação completa dos scripts utilizados nessas etapas, com a respectiva descrição, integra o Apêndice A e será disponibilizada em repositório público para fins de reprodutibilidade.

3.4 ELABORAÇÃO DO QUADRO DE CLASSIFICAÇÃO TIPOLÓGICA

A construção do quadro de classificação tipológica adotado nesta pesquisa partiu da premissa de que nenhuma fonte isolada, normativa ou comercial, contempla de forma suficiente a diversidade real de cadeiras de rodas em circulação. Por esse motivo, optou-se por uma abordagem integrativa que articulou três eixos complementares de classificação. O primeiro eixo corresponde às normas internacionais e nacionais: a ISO 9999 (2022), que organiza os produtos de tecnologia assistiva em classes e subclasses funcionais, e a ABNT NBR 9050 (2020), que,

embora não classifique as cadeiras de rodas diretamente, estabelece dimensões referenciais distintas para cadeiras manuais, motorizadas e esportivas, reconhecendo implicitamente diferenças tipológicas relevantes para o trabalho. O segundo eixo incorpora as categorias adotadas pelos dois principais fabricantes nacionais, Ortobras e Jaguaribe, cujos portfólios refletem a realidade do mercado brasileiro e abrangem linhas que vão de cadeiras manuais dobráveis a modelos posturais, infantis, bariátricos e anfíbios (Ortobras, 2026; Jaguaribe, 2026). O terceiro eixo contempla as tipologias praticadas por dois dos maiores fabricantes globais, Sunrise Medical e Invacare Corporation, que introduzem categorias de alta especialização, como cadeiras do tipo *standing* (verticalização) e modelos esportivos por modalidade, ampliando o espectro tipológico para além do que as normas brasileiras preveem (Sunrise Medical, 2026; Invacare, 2026). A partir do cruzamento dessas fontes, foram eliminadas as categorias redundantes ou com sobreposição funcional, preservando-se apenas as tipologias genuinamente distintas em termos de finalidade, público-alvo, forma construtiva e contexto de uso. O resultado dessa síntese é apresentado no Quadro 5, que constitui a base taxonômica para a anotação das classes de detecção no treinamento dos modelos *YOLO* empregados nesta pesquisa.

Além de fundamentar o protocolo de anotação descrito na Seção 3.5, a tipologia consolidada no Quadro 5 orientou diretamente a estratégia de constituição do *corpus* apresentada na Seção 3.3, motivando, em particular, as duas rodadas de busca direcionada via *Bing Image Search* voltadas às classes (pediátrica, postural, bariátrica, *standing* e anfíbia) sub-representadas nos repositórios públicos consultados.

Quadro 5 - Classificação Tipológica Consolidada de Cadeiras de Rodas

Tipologia	Dica Visual Chave	Descrição	Imagem de Referência
Manual Dobrável	Mecanismo em “X” cruzado entre as longarinas do quadro; encosto e assento em tecido.	Cadeira de rodas com estrutura dobrável (sistema em X), propulsão manual pelo próprio usuário ou por acompanhante. Fabricada em aço ou alumínio, é o modelo mais difundido para uso cotidiano e transporte.	Figura A 1 no Anexo I
Manual Ativa (Monobloco/Rígida)	Quadro rígido contínuo, sem mecanismo em X; rodas traseiras com câmbio negativo (inclinadas para dentro); ausência frequente de apoios de braço.	Cadeira com quadro rígido (não dobrável), ultraleve, projetada para usuários ativos com alta demanda de mobilidade. Estrutura em alumínio aeronáutico ou fibra de carbono, com eixo traseiro ajustável para maior eficiência de propulsão.	Figura A 2 no Anexo I

Motorizada (Elétrica)	Joystick lateral; bateria visível sob o assento; rodízios pequenos frontais e rodas motrizes maiores; ausência de aros de propulsão.	Cadeira de rodas com propulsão por motor elétrico, comandada por joystick ou controle adaptado. Dispõe de baterias recarregáveis e é indicada para usuários com limitações nos membros superiores ou que necessitam de maior autonomia em trajetos longos.	Figura A 3 no Anexo I
Postural (<i>Tilt-in-Space</i>/Reclinável)	Encosto alto reclinável; mecanismo de inclinação visível na base; apoio de cabeça acoplado; cintos posturais.	Cadeira dotada de mecanismo de inclinação (<i>tilt</i>) e/ou reclinção do encosto, projetada para adequação postural de usuários com comprometimento do controle de tronco. Permite redistribuição de pressão e posicionamento terapêutico personalizado.	Figura A 4 no Anexo I
Pediátrica/Infantil	Dimensões reduzidas em relação ao usuário; cores e estética infantil; sistemas de contenção e apoios laterais.	Cadeira dimensionada para crianças e adolescentes, com sistema de crescimento (ajuste de largura, profundidade e altura) para acompanhar o desenvolvimento do usuário. Apresenta elementos anatômicos e estética diferenciada.	Figura A 5 no Anexo I
Bariátrica (<i>Heavy Duty</i>)	Estrutura visivelmente mais larga e robusta; assento ampliado; reforço duplo no quadro.	Cadeira com estrutura reforçada (aço carbono ou alumínio de alta resistência), assento ampliado e capacidade de carga elevada (acima de 150 kg, podendo atingir 250 kg ou mais). Projetada para usuários obesos ou com grande porte corporal.	Figura A 6 no Anexo I
Esportiva	Câmbor acentuado nas rodas traseiras; quadro extremamente baixo; ausência de empunhaduras; proteções laterais e roda frontal antitombamento.	Cadeira com <i>design</i> específico para a prática de modalidades paradesportivas (basquete, tênis, rugby, corrida, entre outras). Possui quadro rígido ultraleve, cambagem nas rodas traseiras para maior agilidade e estabilidade, e proteções laterais.	Figura A 7 no Anexo I
Banho/Higiênica	Assento sanitário (com abertura central); rodízios pequenos em todas as posições; estrutura tubular em alumínio ou PVC.	Cadeira fabricada em materiais resistentes à corrosão (alumínio com pintura epóxi ou PVC), com assento sanitário e rodízios adequados para ambientes molhados. Destinada ao uso em banheiros para higiene pessoal e banho, podendo ser acoplada ao vaso sanitário.	Figura A 8 no Anexo I
Hospitalar/Transporte	Quatro rodas pequenas (sem aros de propulsão); empunhaduras altas para acompanhante; cor neutra padronizada (cinza, branco, azul).	Cadeira de rodas padronizada para uso em ambiente clínico e hospitalar, destinada ao transporte interno de pacientes. Estrutura robusta e de fácil higienização, geralmente com rodas menores para condução por acompanhante e apoios de pés eleváveis.	Figura A 9 no Anexo I
<i>Standing</i> (Verticalização)	Mecanismo de elevação vertical visível; apoio de joelhos e cinto	Cadeira motorizada com sistema integrado de verticalização (<i>standing</i>), que permite ao usuário assumir a	Figura A 10 no Anexo I

	torácico; usuário em postura ereta ou intermediária.	posição ereta a partir da posição sentada. Indicada para benefícios fisiológicos, funcionais e sociais associados à postura em pé.	
Anfibia/Praia	Rodas flutuantes de grande diâmetro com pneus largos; assento em malha drenante; estrutura em PVC ou alumínio anodizado.	Cadeira projetada para ambientes aquáticos e terrenos arenosos (praias, piscinas). Possui estrutura anticorrosiva, rodas flutuantes de grande diâmetro e materiais impermeáveis, permitindo o acesso a ambientes de lazer antes inacessíveis.	Figura A 11 no Anexo I

Fonte: Adaptado de Ortobras (2026); Jaguaribe (2026); Sunrise Medical (2026); Invacare (2026); ISO 9999 (2022, classe 12 22 03); ABNT NBR 9050 (2020).

A composição do Quadro 5 articula contribuições específicas de cada uma das fontes consultadas. Da norma ISO 9999 (2022) derivam as classes funcionais 12 22 03 (cadeiras manuais) e 12 22 06 (cadeiras motorizadas), que fundamentam, respectivamente, a distinção entre as tipologias Manual Dobrável e Manual Ativa, de um lado, e a tipologia Motorizada, de outro. A ABNT NBR 9050 (2020), ao estabelecer dimensões referenciais distintas para cadeiras manuais, motorizadas e esportivas, sustenta o reconhecimento dessas três categorias e ampara, de modo complementar, a tipologia Hospitalar/Transporte por meio do módulo de referência aplicável ao trânsito interno em estabelecimentos de saúde.

No plano comercial, a Ortobras (2026) embasa as tipologias Manual Ativa (linhas *Star Lite*, M3 e M6), Motorizada (linhas E3, E4, E6, E9 e E20), Postural (linhas *Conforma Tilt*, *Conforma Tilt Reclinável* e T1), Pediátrica/Infantil (linhas *Genesys*, *Mini K*, *Mini Star Lite* e *Kinder*), Banho/Higiênica (linhas H1, H2, H3, *Hygienika* e *Flex*) e Anfíbia/Praia (modelo Ipanema). A Jaguaribe (2026) contribui com as tipologias Manual Ativa (linha *Speed*), Postural (linhas *Star Postural* e *KR Reclinável*), Pediátrica/Infantil (linhas *Ágile Baby* e *Star Kids*), Bariátrica (linhas *BIG 160 kg*, *Poty 150 kg* e *Super Big Monobloco 250 kg*), Banho/Higiênica (linhas *Banho RG*, *Banho Pop*, *Banho Big* e *Banho Alumínio Reclinável*) e Hospitalar/Transporte (linhas *Ágile Hospitalar*, *Big Hospitalar* e *Poty Hospitalar 150 kg*). Ambos os fabricantes nacionais sustentam, ainda, a tipologia Manual Dobrável, predominante em seus catálogos.

No plano internacional, a Sunrise Medical (2026) embasa as tipologias Manual Ativa (marca *Quickie*, linha *Ultralight Rigid*), Motorizada (*Quickie Power Wheelchairs*), Postural (*Quickie IRIS Tilt-in-Space*), Pediátrica/Infantil (marca *Zippie*), Bariátrica (*Quickie Heavy Duty Manual Wheelchairs*), Esportiva (marca *RGK Sports Wheelchairs*), Banho/Higiênica (*Bathroom Wheelchairs*), Hospitalar/Transporte

(*Transport Wheelchairs*), *Standing* (*Quickie Q700-UP M Standing Wheelchair*) e Anfíbia/Praia (*Beach Wheelchairs*). A Invacare (2026) contribui com as tipologias Manual Ativa (*série 9000 XT*), Motorizada (*séries TDX, FDX e Storm*), Postural (*séries Tilt/Recline, Fórmula CG*), Pediátrica/Infantil (linha pediátrica dedicada), Bariátrica (*séries Tracer IV Bariatric e Action Ampla*), Esportiva (*linha Sport Wheelchairs*), Hospitalar/Transporte (*linhas Standard e Transport*) e *Standing* (*AVIVA FX, MPS Maxx Standing System*).

As tipologias apresentadas resultam da consolidação das classificações normativas (ISO 9999:2022 e ABNT NBR 9050:2020) com as categorias adotadas pelos principais fabricantes nacionais (Ortobras e Jaguaribe) e internacionais (*Sunrise Medical e Invacare*). Foram eliminadas sobreposições e mantidas apenas as categorias funcionalmente distintas em termos de finalidade, público-alvo, forma construtiva e contexto de uso, constituindo a base taxonômica que orientará a anotação descrita na Seção 3.5.1.

3.5 ANOTAÇÃO E DIVISÃO DOS DADOS

A etapa de anotação manual e divisão estratificada do *corpus* constitui o alicerce metodológico sobre o qual se assenta toda a cadeia de treinamento e validação dos modelos de detecção. No contexto desta pesquisa, o procedimento foi integralmente conduzido em estação de trabalho pessoal do pesquisador, o que reforça a viabilidade de replicação do método em ambientes computacionais convencionais, sem a necessidade de infraestruturas laboratoriais especializadas para a fase de preparação dos dados. A escolha por um fluxo de trabalho local alinha-se ao princípio norteador desta dissertação de desenvolver uma solução assistiva acessível e de baixo custo em todas as suas etapas.

O processo foi estruturado em duas fases complementares e sequenciais: (i) a anotação das 2.000 imagens conforme protocolo padronizado, com identificação das onze tipologias consolidadas no Quadro 5; e (ii) a divisão estratificada do conjunto em subconjuntos de treinamento, validação e teste, na proporção 70/20/10. Ambas as fases seguem critérios documentados e reproduzíveis, de modo a garantir a transparência metodológica e a possibilidade de auditoria e replicação por pesquisadores independentes, conforme preconizado por Marconi e Lakatos (2017).

O *corpus* de 2.000 imagens (consolidado conforme descrito na Seção 3.3 a partir das fontes *Wheelchair Dataset (Kaggle)*, *Open Images V7*, *MobilityAids Freiburg* e duas rodadas complementares de busca direcionada via *Bing Image Search*) foi organizado em um diretório local de trabalho com nomenclatura sequencial padronizada, de 0000 a 1999, garantindo a unicidade dos identificadores e a compatibilidade com os *scripts* de partição e treinamento empregados nas etapas subsequentes. As subseções a seguir descrevem em detalhe o protocolo de anotação adotado (Seção 3.5.1) e os critérios de partição estratificada do conjunto em treinamento, validação e teste (Seção 3.5.2).

3.5.1 Protocolo de Anotação

O protocolo de anotação adotado nesta pesquisa foi desenvolvido com o objetivo de assegurar a consistência, a rastreabilidade e a reprodutibilidade das marcações realizadas sobre o banco de imagens. A anotação foi executada de forma manual pelo pesquisador, utilizando a ferramenta de código aberto *LabelImg* (versão 1.8.6 ou superior), amplamente empregada pela comunidade de visão computacional para tarefas de anotação de objetos em formato compatível com a arquitetura *YOLO* (Tzutalin, 2015). A escolha desta ferramenta justifica-se por sua interface intuitiva, compatibilidade multiplataforma e capacidade de exportação direta no formato *YOLO TXT*, eliminando a necessidade de conversões intermediárias que poderiam introduzir erros ou inconsistências nos dados.

A opção pela anotação restrita à estrutura do equipamento integra a estratégia ética desta pesquisa em duas frentes complementares. Em primeiro lugar, todas as imagens utilizadas no treinamento provêm de repositórios de acesso público disponibilizados sob licenças que autorizam o uso acadêmico e a redistribuição (*MIT*, *Creative Commons Attribution 4.0 International* e licença de pesquisa para o *MobilityAids*), nos termos descritos na Seção 3.3, de modo que as pessoas eventualmente retratadas nas cenas o foram em condições de consentimento prévio para uso e divulgação de sua imagem em contextos de pesquisa. Em segundo lugar, mesmo nesse cenário de licenciamento permissivo, a marcação manual circunscreve-se exclusivamente à estrutura da cadeira, evitando a captura de feições faciais ou de outros elementos identificadores e reforçando, no plano operacional, o princípio de

volatilidade de dados estabelecido no Capítulo 1 e operacionalizado no sistema embarcado descrito na Seção 3.7.

O uso do *Labellmg* na estação de trabalho exigiu, ainda, uma correção pontual em seu código-fonte para compatibilizá-lo com versões recentes do *Python*, sem a qual a ferramenta interrompia a execução ao redimensionar as caixas delimitadoras pelo mouse. O detalhamento dessa correção, mantido em nome da transparência metodológica e da reprodutibilidade, é apresentado no Apêndice A.

O formato de anotação adotado corresponde ao padrão *YOLO TXT*, no qual cada imagem possui um arquivo de texto homônimo (extensão *.txt*) contendo uma linha para cada objeto anotado na cena. Cada linha segue a estrutura:

<classe> <x_centro> <y_centro> <largura> <altura>

Na qual *<classe>* corresponde ao índice numérico inteiro da tipologia de cadeira de rodas (variando de 0 a 10, seguindo a convenção de indexação iniciada em zero dos *frameworks* YOLO e correspondentes às onze tipologias consolidadas no Quadro 5), e os quatro valores subsequentes representam, respectivamente, as coordenadas normalizadas do centro da caixa delimitadora (*bounding box*) e suas dimensões relativas à resolução da imagem, com valores no intervalo [0, 1]. Essa normalização garante a independência do formato em relação à resolução original da imagem, conferindo portabilidade ao banco de dados anotado.

O mapeamento entre os índices numéricos e as tipologias de cadeiras de rodas obedece à classificação consolidada no Quadro 5, sendo definido conforme a seguinte correspondência: 0 – Manual Dobrável; 1 – Manual Ativa (Monobloco/Rígida); 2 – Motorizada (Elétrica); 3 – Postural (*Tilt-in-Space*/Reclinável); 4 – Pediátrica/Infantil; 5 – Bariátrica (*Heavy Duty*); 6 – Esportiva; 7 – Banho/Higiênica; 8 – Hospitalar/Transporte; 9 – *Standing* (Verticalização); e 10 – Anfíbia/Praia. Esse mapeamento foi registrado em um arquivo de metadados denominado *classes.txt*, armazenado no diretório raiz do banco de imagens, assegurando a correspondência unívoca entre rótulos e categorias ao longo de todo o ciclo de treinamento e avaliação.

Por convenção de codificação dos *frameworks* YOLO, os nomes das tipologias registrados no arquivo *classes.txt* foram grafados em formato simplificado e sem acentos (por exemplo, *Manual Dobravel* em vez de Manual Dobrável), preservando-se o formato completo das denominações apenas no Quadro 5 e no corpo do texto.

Para garantir a padronização e a qualidade das anotações, foram estabelecidos os seguintes critérios operacionais, aplicados sistematicamente a cada imagem do banco de dados:

a) Enquadramento da caixa delimitadora: a *bounding box* deve envolver exclusivamente a estrutura da cadeira de rodas (quadro, rodas, apoios de braço e apoios de pés), mesmo nas imagens em que a pessoa usuária esteja presente. Em conformidade com o princípio ético adotado nesta pesquisa, a marcação se restringe ao equipamento assistivo e não inclui o rosto, partes do corpo identificáveis ou quaisquer elementos que possam permitir a identificação do indivíduo. Áreas excessivas de fundo também devem ser evitadas, mantendo-se a caixa ajustada aos contornos da estrutura. Em situações de oclusão parcial, nas quais parte da cadeira encontra-se obstruída por outros objetos ou pelo próprio usuário, a caixa delimitadora deve abranger apenas a porção visível da estrutura, desde que ao menos 30% dela permaneça identificável; abaixo desse limiar, a imagem é tratada como negativa, com arquivo de anotação vazio. A definição do limiar fundamenta-se nas recomendações da documentação oficial do *framework* YOLO (Jocher; Chaurasia; Qiu 2023), que orienta a exclusão de anotações excessivamente fragmentadas em razão de sua tendência a degradar a capacidade discriminativa do modelo, e foi consolidada ao longo da própria fase de anotação, na qual se verificou que recortes inferiores a esse limiar tornavam ambígua a identificação tipológica mesmo para o anotador humano.

b) Classificação tipológica: cada cadeira de rodas foi classificada de acordo com a tipologia predominante identificável visualmente, conforme os critérios morfológicos e funcionais descritos no Quadro 5 e no Anexo I desta dissertação. Em casos de ambiguidade tipológica, priorizou-se a característica estrutural mais distintiva do equipamento. Quando a tipologia não pôde ser determinada com confiança razoável a partir da imagem, o exemplar foi anotado como a classe mais provável.

c) Múltiplas instâncias por imagem: imagens contendo mais de uma cadeira de rodas foram anotadas com múltiplas linhas no arquivo de anotação correspondente, cada uma representando uma instância distinta com sua respectiva classe e coordenadas. Essa abordagem permite que o modelo aprenda a detectar múltiplos objetos em uma mesma cena, cenário recorrente em ambientes públicos e hospitalares.

d) Imagens negativas: foram incluídas no banco de dados imagens que não contêm cadeiras de rodas, denominadas imagens negativas ou de fundo (*background*

images), as quais possuem arquivos de anotação vazios (sem linhas). Essas imagens são essenciais para que o modelo aprenda a distinguir cenários sem o objeto de interesse, reduzindo a taxa de falsos positivos durante a inferência. Foram selecionadas 250 imagens negativas a partir do *MobilityAids Freiburg* (Vasquez et al., 2017), por meio de amostragem aleatória com semente fixa (*random seed* = 42); destas, 182 permaneceram no *corpus* final após a triagem visual descrita na Seção 3.3, correspondendo a 9,1% do total de 2.000 imagens, proporção próxima à recomendação da documentação oficial do *framework YOLO* de cerca de 10% (Jocher; Chaurasia; Qiu 2023).

e) Controle de qualidade: ao término da etapa de anotação, procedeu-se a uma revisão sistemática de 100% das marcações por meio de inspeção visual auxiliada pela função de visualização do próprio *Labellmg*, que permite sobrepor as caixas delimitadoras às imagens originais. Essa revisão objetivou identificar e corrigir erros de enquadramento, classificação tipológica equivocada e anotações ausentes. Adicionalmente, foi executado um *script* de validação em *Python* para verificar a integridade do banco de dados, confirmando que: (i) cada imagem possui um arquivo de anotação correspondente; (ii) os índices de classe estão dentro do intervalo esperado (0 a 10); (iii) os valores de coordenadas estão normalizados entre 0 e 1; e (iv) não há arquivos corrompidos ou com formatação inconsistente. O *script* de validação foi documentado e será disponibilizado como material suplementar para fins de replicação.

Cabe ressaltar que, por se tratar de anotação realizada por um único pesquisador, reconhece-se a limitação inerente à ausência de cálculo de concordância interanotadores (*inter-annotator agreement*). Para mitigar essa limitação, adotou-se a revisão integral descrita no item (e). Adicionalmente, o aplicativo de coleta iterativa descrito na Seção 3.8, que opera em campo gerando pseudoanotações revisadas manualmente, constitui uma estratégia complementar para o aprimoramento progressivo do banco de dados, possibilitando a verificação retrospectiva das classes mais sujeitas a ambiguidade. Em pesquisas subsequentes, recomenda-se a incorporação de um segundo anotador independente para a validação cruzada de ao menos uma amostra representativa do banco de dados, possibilitando o cálculo de métricas de concordância, como o coeficiente *Kappa de Cohen*.

3.5.2 Divisão dos Dados (Treinamento/Validação/Teste)

A integridade da avaliação de um modelo de aprendizado de máquina depende fundamentalmente da segregação rigorosa dos dados em subconjuntos funcionalmente independentes, de modo a evitar vazamento de informação (*data leakage*) entre as etapas de treinamento, ajuste de hiperparâmetros e avaliação final (Marconi; Lakatos, 2017). Nesta pesquisa, o banco de imagens anotado foi particionado em três subconjuntos mutuamente exclusivos, seguindo a proporção planejada de 70% para treinamento, 20% para validação e 10% para teste, conforme ilustrado na Figura 7 (A) e em consonância com as práticas consolidadas na literatura de visão computacional aplicada a tecnologias assistivas (Adorno *et al.*, 2023; Alruwaili *et al.*, 2024).

A divisão foi realizada por meio de um *script* automatizado em *Python*, que implementa uma estratégia de amostragem aleatória estratificada proporcional por tipologia. Esse procedimento assegura que a distribuição relativa das onze classes de cadeiras de rodas seja mantida de forma aproximadamente proporcional em cada um dos três subconjuntos, evitando vieses de representação que poderiam comprometer a capacidade de generalização do modelo para tipologias sub-representadas. O *script* recebe como entrada o diretório de imagens anotadas e produz como saída três diretórios distintos (*train*, *val* e *test*), cada qual contendo os pares de arquivos imagem-anotação correspondentes, organizados na estrutura esperada pelos *frameworks* YOLOv5 e YOLOv8.

O procedimento de divisão obedece à seguinte sequência operacional:

a) Inventário e catalogação: o *script* percorre o diretório de anotações e cataloga todas as imagens válidas, extraindo a classe predominante de cada exemplar a partir do arquivo de anotação correspondente. Para imagens com múltiplas instâncias de classes distintas, adota-se a classe com maior frequência de ocorrência na imagem como critério de estratificação.

b) Embaralhamento determinístico: antes da partição, o conjunto de dados é embaralhado utilizando uma semente aleatória fixa (*random seed* = 42), garantindo que a mesma divisão possa ser reproduzida integralmente em execuções subsequentes. Essa semente é documentada no código-fonte e nos metadados do experimento, assegurando a rastreabilidade do processo.

c) Partição estratificada: a função de divisão estratificada separa o banco de dados respeitando a proporção de 70/20/10 dentro de cada estrato tipológico. Para tipologias com número reduzido de exemplares, garante-se a presença de ao menos um representante em cada subconjunto, priorizando a representação no conjunto de treinamento. Essa medida é essencial para que o modelo tenha acesso à variabilidade tipológica completa durante a fase de aprendizado.

d) Verificação de integridade pós-divisão: após a partição, o *script* gera automaticamente um relatório de validação contendo: (i) a contagem absoluta e percentual de imagens em cada subconjunto; (ii) a distribuição de classes por subconjunto; (iii) a confirmação de que nenhuma imagem aparece em mais de um subconjunto (exclusividade mútua); e (iv) a verificação de que todos os pares imagem-anotação foram alocados sem perdas. Esse relatório é salvo em formato CSV no diretório raiz do projeto para fins de documentação e auditoria.

A estrutura final de diretórios resultante, compatível com os formatos esperados pelos *frameworks* YOLOv5 e YOLOv8, é apresentada em detalhe no Apêndice A.

O arquivo *data.yaml*, exigido pela API (do termo em inglês *Application Programming Interface, interface de programação de aplicações*) de treinamento das redes YOLO, foi gerado automaticamente pelo *script* de divisão e contém os caminhos absolutos para os diretórios de treinamento e validação, o número total de classes (onze) e a lista ordenada dos nomes das tipologias, garantindo a correspondência entre os índices numéricos das anotações e os rótulos textuais utilizados na avaliação dos resultados.

A aplicação efetiva do procedimento sobre o *corpus* de 2.000 imagens anotadas resultou em 1.394 imagens no conjunto de treinamento (69,7%), 394 no conjunto de validação (19,7%) e 212 no conjunto de teste (10,6%), proporções consistentes com o desenho planejado de 70/20/10 e com pequenas variações decorrentes do arredondamento por estrato tipológico. A escolha dessa proporção justifica-se por representar um equilíbrio adequado entre a necessidade de alimentar o modelo com volume suficiente de dados para o ajuste dos pesos (treinamento), a disponibilidade de um subconjunto robusto para monitoramento da convergência e prevenção de sobreajuste (validação), e a reserva de um conjunto inédito para a avaliação final imparcial do desempenho (teste). Essa proporção é amplamente adotada em pesquisas de detecção de objetos com bases de dados de escala

moderada (Ferrante *et al.*, 2024) e demonstrou-se eficaz em trabalhos diretamente correlatos à detecção de cadeiras de rodas (Adorno *et al.*, 2023; Alruwaili *et al.*, 2024).

A inspeção da distribuição pós-divisão revelou, contudo, um desbalanceamento expressivo entre as onze classes: enquanto a tipologia Manual Dobrável reúne 1.081 exemplares (refletindo sua predominância nos repositórios consultados e no próprio mercado de cadeiras de rodas), a tipologia Hospitalar/Transporte conta com apenas 14 imagens, e classes como Bariátrica (32), Manual Ativa (48) e Postural (55) apresentam volumes consideravelmente reduzidos. Esse desequilíbrio, decorrente da disponibilidade assimétrica de imagens nas fontes públicas, foi mantido intencionalmente por refletir a frequência relativa das tipologias em cenários reais e por integrar a estratégia de ampliar progressivamente o *corpus* por meio do ciclo de coleta iterativa descrito na Seção 3.8. Ressalta-se que a influência desse desbalanceamento sobre o desempenho por classe é discutida em detalhe no Capítulo 4, com as devidas ressalvas metodológicas, especialmente no que se refere às tipologias sub-representadas. Registre-se, para a leitura correta dos números ao longo do trabalho, a distinção entre imagens (arquivos do corpus) e instâncias (objetos anotados em cada partição): uma mesma imagem pode conter mais de uma instância, e os totais por partição correspondem a frações do corpus completo. Assim, por exemplo, a tipologia Hospitalar/Transporte reúne 14 imagens no corpus total e 10 instâncias no conjunto de treinamento, enquanto a Manual Ativa, com 48 imagens no corpus, contribui com 60 instâncias de treinamento, e a Esportiva, com 131 imagens, com 338, reflexo da presença frequente de múltiplas cadeiras em uma mesma fotografia nessas categorias (Seção 4.1).

Ressalta-se que as métricas de desempenho reportadas no Capítulo 4 foram calculadas sobre o subconjunto de validação, que, embora monitorado para a seleção do melhor checkpoint (*best.pt*) pelo mecanismo de *early stopping*, não participa da atualização dos pesos da rede e constitui, portanto, um conjunto independente do treinamento. A opção por reportar o desempenho sobre a validação, e não sobre o subconjunto de teste de 10%, decorre diretamente do desbalanceamento de cauda longa documentado nesta seção: classes severamente sub-representadas, como a Hospitalar/Transporte, com apenas 14 imagens no corpus total, restariam com somente uma a duas instâncias em uma partição de teste de 10%, volume insuficiente para uma estimativa estável de mAP por classe. O subconjunto de validação, com 394 imagens e 457 instâncias representando todas as onze tipologias, oferece a base

amostral mínima necessária para a análise de desempenho por classe conduzida na Seção 4.3, preservando, ao mesmo tempo, a independência em relação ao ajuste dos pesos. Essa segregação temporal é fundamental para assegurar que as métricas de desempenho reportadas no Capítulo 4 reflitam o desempenho do sistema sobre um conjunto independente do ajuste dos pesos, evitando estimativas otimistas decorrentes de vazamento de informação; o subconjunto de teste de 10% permanece reservado para a validação final em campo, etapa que extrapola o escopo temporal desta dissertação e é tratada como trabalho futuro no Capítulo 5. Todo o código-fonte utilizado nas etapas de divisão e validação dos dados será disponibilizado em repositório público para garantir a reprodutibilidade integral do experimento.

3.6 PROCESSOS DE TREINAMENTO DOS MODELOS

O treinamento dos modelos de detecção constitui a etapa central do fluxo experimental, na qual os padrões visuais das onze tipologias de cadeiras de rodas consolidadas no Quadro 5 são aprendidos pela rede neural a partir do *corpus* de 2.000 imagens anotado e particionado conforme descrito na Seção 3.5. Nesta pesquisa, o processo foi conduzido integralmente na estação de trabalho pessoal do pesquisador, em coerência com o princípio de baixo custo e replicabilidade que orienta o trabalho desde a constituição do *corpus*.

O delineamento experimental contemplou o treinamento e a avaliação comparativa de duas arquiteturas da família *YOLO* em duas variantes cada (*YOLOv5n*, *YOLOv5s*, *YOLOv8n* e *YOLOv8s*), totalizando quatro modelos. A *YOLOv5* foi incluída por sua consolidação em ambientes com restrições computacionais (Alruwaili *et al.*, 2024), enquanto a *YOLOv8* foi adotada por representar o estado da arte em precisão de detecção, com resultados superiores reportados na literatura para a identificação de cadeiras de rodas (Ali; Zhang, 2024). A opção pelas variantes nano (n) e *small* (s), em detrimento das versões *medium* (m), *large* (l) e *xlarge* (x) disponíveis em ambas as arquiteturas, fundamenta-se na exigência de viabilidade de execução em *hardware* embarcado de baixa potência (*Raspberry Pi 5*) na etapa subsequente da pesquisa, conforme detalhado na Seção 3.7. As variantes maiores, embora frequentemente associadas a métricas de precisão superiores em *hardware* dedicado, são incompatíveis com os requisitos de inferência em tempo real e baixo consumo energético próprios da plataforma embarcada.

3.6.1 Configurações dos Modelos e Hiperparâmetros

O treinamento dos quatro modelos foi executado em estação de trabalho pessoal do pesquisador, cuja configuração de *hardware* é detalhada no Quadro 6. A opção pelo treinamento local, em detrimento de plataformas de computação em *nuvem*, justifica-se por três fatores: (i) a eliminação de dependências externas que poderiam comprometer a reprodutibilidade dos experimentos em função de alterações na infraestrutura ou na disponibilidade dos serviços; (ii) o controle total sobre o ambiente de *software*, versões de bibliotecas e *drivers*, assegurando a estabilidade do *pipeline* de treinamento; e (iii) a coerência com a filosofia de baixo custo e acessibilidade que norteia esta dissertação, demonstrando que o desenvolvimento completo do sistema assistivo é viável com recursos computacionais domésticos.

Quadro 6 - Configuração de *hardware* da estação de trabalho utilizada para treinamento

Componente	Especificação
Processador (CPU)	<i>AMD Ryzen 7 7700X</i> , 8 núcleos / 16 <i>threads</i> , 4,5 GHz (turbo 5,4 GHz), <i>cache</i> 40 MB, <i>socket</i> AM5
Placa-mãe	<i>Gigabyte B650 Eagle AX</i> , chipset <i>AMD B650</i> , <i>socket</i> AM5, <i>DDR5</i>
Memória RAM	32 GB (2 × 16 GB) <i>DDR5</i> 6000 MHz, <i>Team Group</i> T-Force Vulcan A, CL38, dual-channel
Armazenamento	<i>SSD Kingston Fury Renegade</i> , 1 TB, <i>M.2 2280</i> , <i>PCIe 4.0 NVMe</i> , leitura 7.300 MB/s, gravação 6.000 MB/s
Placa de vídeo (GPU)	<i>ASRock AMD Radeon RX 7600 Challenger OC</i> , 8 GB <i>GDDR6</i> , arquitetura <i>RDNA 3</i>
Sistema operacional	<i>Windows 11</i> com suporte a <i>DirectML</i>

Fonte: Elaborado pelo Autor (2026).

A estação de trabalho dispõe de uma GPU (do termo em inglês *Graphics Processing Unit*, unidade de processamento gráfico) AMD Radeon RX 7600 com 8 GB de memória dedicada *GDDR6*, baseada na arquitetura *RDNA 3*. Embora o planejamento inicial previsse o uso desta GPU para a aceleração das operações de treinamento por meio da plataforma *AMD ROCm* (*Radeon Open Compute*), constatou-se, no início da fase experimental, que o *ROCm* não possui suporte oficial no sistema operacional *Windows*, restringindo-se a distribuições *Linux*. Diante da indisponibilidade de ambiente *Linux* nativo no *hardware* em uso e da decisão metodológica de manter o fluxo de trabalho integralmente local, optou-se por conduzir o treinamento exclusivamente em CPU (do termo em inglês *Central Processing Unit*,

unidade central de processamento), aproveitando os 8 núcleos físicos e 16 *threads* do processador *Ryzen 7 7700X*. As versões dos componentes de *software* adotadas foram *Python* 3.13.9 e *PyTorch* 2.10.0 (variante para *CPU*, instalada a partir do índice oficial do *PyTorch* dedicado a sistemas sem aceleração por *GPU*).

Esta restrição imposta pela ausência de compatibilidade entre *ROCm* e *Windows* traduziu-se em maior tempo de execução, mas não em prejuízo à qualidade dos modelos resultantes: o treinamento dos quatro modelos consumiu aproximadamente 41 horas de processamento em *CPU*, distribuídas em sessões noturnas ao longo de cinco dias consecutivos. Os tempos individuais variaram entre as variantes, cerca de 11 horas para o *YOLOv5n* e o *YOLOv5s*, 7 horas para o *YOLOv8n* (a variante mais eficiente do conjunto) e 12 horas para o *YOLOv8s*, conforme detalhado no Capítulo 4. Os pesos finais resultantes são funcionalmente independentes do dispositivo de processamento utilizado durante o treinamento, de modo que a opção pela *CPU*, embora mais lenta, não compromete a qualidade nem a comparabilidade dos modelos. Ressalta-se ainda que, no início da fase de treinamento, a biblioteca *Ultralytics* reportou ausência de *workers* de carregamento de dados (*dataloader workers*) no ambiente *Windows*, apesar da configuração padrão prever oito; trata-se de limitação conhecida do *framework* em sistemas *Windows* com multiprocessamento *Python*, registrada aqui em nome da transparência metodológica e sem impacto sobre a qualidade dos resultados, embora com efeito mensurável sobre o tempo total de execução.

O *armazenamento* em *SSD NVMe PCIe 4.0*, com taxas de leitura sequencial de até 7.300 MB/s, desempenhou papel relevante na etapa de treinamento ao eliminar gargalos de entrada e saída (I/O) durante o carregamento dos lotes de imagens pelo *dataloader*. A memória *RAM* de 32 GB *DDR5* a 6.000 MHz em configuração dual-channel assegurou largura de banda suficiente para o pré-processamento paralelo das imagens e o *armazenamento* dos *buffers* intermediários, evitando interrupções no fluxo de alimentação da *CPU* durante o ciclo de treinamento.

Previamente ao treinamento, os modelos foram inicializados com pesos pré-treinados no conjunto COCO, uma base de referência amplamente adotada para tarefas de detecção de objetos. Essa estratégia de *transfer learning* (aprendizado por transferência) permite que a rede aproveite representações visuais genéricas já aprendidas, como bordas, texturas e formas geométricas, acelerando a convergência e melhorando o desempenho em conjuntos de dados específicos e de menor escala

(Ferrante *et al.*, 2024). Os pesos pré-treinados utilizados correspondem aos arquivos *yolov5n.pt*, *yolov5s.pt*, *yolov8n.pt* e *yolov8s.pt*, disponibilizados oficialmente pela *Ultralytics* (Jocher; Chaurasia; Qiu, 2023). Cabe ressaltar que os modelos *YOLOv5* foram treinados a partir do repositório oficial original do *YOLOv5* (também mantido pela *Ultralytics*), enquanto os modelos *YOLOv8* utilizaram a biblioteca *Ultralytics* em sua versão mais recente, em conformidade com a recomendação atual de cada arquitetura.

Os hiperparâmetros de treinamento foram aplicados de forma idêntica aos quatro modelos, garantindo a comparabilidade direta dos resultados. O Quadro 7 sintetiza os principais parâmetros adotados.

Quadro 7 - Hiperparâmetros de treinamento dos modelos *YOLOv5* e *YOLOv8*

Hiperparâmetro	Valor Adotado	Justificativa
Resolução de entrada	640 × 640 pixels	Padrão das arquiteturas <i>YOLOv5/v8</i> para equilíbrio entre resolução e velocidade de inferência
Épocas (epochs)	150	Intervalo intermediário (100–200), suficiente para convergência sem sobreajuste excessivo
Batch size (tamanho do lote)	16	Valor padrão das implementações <i>YOLOv5/v8</i> , que equilibra a estabilidade da estimativa do gradiente e o consumo de memória; como o treinamento foi conduzido em CPU (Seção 3.6.1), a restrição relevante é a memória RAM de 32 GB da estação, amplamente suficiente para esse tamanho de lote
Learning rate inicial	0,01	Valor padrão recomendado pela <i>Ultralytics</i> para treinamentos com <i>transfer learning</i>
Learning rate final (lrf)	$0,01 \times lr0 = 0,0001$	Decaimento linear ao longo das épocas até 1% do valor inicial
Warm-up	3 épocas	Aumento progressivo da taxa de aprendizado nas primeiras épocas para estabilizar o treinamento a partir dos pesos pré-treinados
Otimizador	SGD (<i>Stochastic Gradient Descent</i>)	Padrão das implementações <i>YOLOv5/v8</i> , com <i>momentum</i> de 0,937 e <i>weight decay</i> de 0,0005
Data augmentation	Mosaic, espelhamento horizontal ($p = 0,5$), ajuste de HSV (matiz,	Técnicas nativas do <i>YOLO</i> para ampliação da variabilidade do

	saturação, brilho); espelhamento vertical desativado	conjunto; o <i>flip</i> vertical foi desativado por não corresponder a uma simetria plausível para cadeiras de rodas em uso
Early stopping (parada antecipada)	Paciência de 30 épocas	Interrompe o treinamento caso o <i>mAP@0.5</i> não apresente melhoria por 30 épocas consecutivas, prevenindo sobreajuste
Número de classes	11	Conforme classificação tipológica consolidada no Quadro 5
Random seed	42	Semente fixa para reprodutibilidade dos experimentos
Dispositivo de processamento	<i>CPU AMD Ryzen 7 7700X</i> (8 núcleos / 16 threads)	Imposto pela ausência de suporte oficial do <i>ROCm</i> para <i>Windows</i> ; impacto restrito ao tempo de treinamento, sem efeito sobre os pesos finais

Fonte: Elaborado pelo Autor (2026).

Os hiperparâmetros descritos no Quadro 7 foram auditados *a posteriori* nos arquivos *args.yaml* (YOLOv8) e *opt.yaml* (YOLOv5) gerados automaticamente em cada um dos quatro treinamentos, confirmando a aplicação uniforme dos mesmos valores ao longo de todas as variantes, em particular, o parâmetro *cos_lr* consta como desativado em todos os arquivos, validando a adoção do decaimento linear como política de ajuste da taxa de aprendizado nos quatro modelos.

A resolução de entrada de 640 × 640 pixels foi adotada como padrão por corresponder à configuração nativa das arquiteturas YOLOv5 e YOLOv8, garantindo compatibilidade com os pesos pré-treinados e com a estrutura de ancoragem dos modelos. As imagens do banco de dados são automaticamente redimensionadas para essa resolução durante o carregamento, com preservação da proporção original por meio de preenchimento (*padding*) nas bordas, procedimento gerenciado nativamente pelo *dataloader* das implementações *Ultralytics*.

As técnicas de aumento de dados (*data augmentation*) foram empregadas de forma nativa, utilizando os mecanismos integrados às implementações YOLOv5 e YOLOv8. A técnica *Mosaic*, proposta originalmente na YOLOv4, combina quatro imagens em uma única composição durante o treinamento, forçando o modelo a aprender a detectar objetos em diferentes escalas e contextos simultaneamente. O espelhamento horizontal (*horizontal flip*) com probabilidade de 0,5 amplia a variabilidade de orientação dos equipamentos, enquanto os ajustes de matiz,

saturação e brilho (espaço de cor HSV) simulam variações de iluminação comuns em ambientes reais do ambiente construído. Optou-se, contudo, pela desativação do espelhamento vertical (*vertical flip*), por não corresponder a uma simetria plausível para cadeiras de rodas em uso, uma vez que tal transformação geraria imagens com a estrutura invertida em relação ao chão, ausentes da distribuição real do objeto a ser detectado. Conforme discutido na Seção 2.4.1, essas técnicas são essenciais para conferir robustez ao modelo diante das condições dinâmicas de captura (Ferrante *et al.*, 2024; Yumang; Banguilan; Veneracion, 2021).

O mecanismo de parada antecipada (*early stopping*) foi configurado com paciência de 30 épocas, monitorando a métrica *mAP@0.5* no subconjunto de validação. Caso o modelo não apresentasse melhoria nessa métrica por 30 épocas consecutivas, o treinamento seria interrompido automaticamente e os pesos correspondentes à melhor época preservados no arquivo *best.pt*. Essa estratégia previne o sobreajuste (*overfitting*), fenômeno no qual o modelo memoriza peculiaridades do conjunto de treinamento e perde capacidade de generalização para dados inéditos, conforme alertado por Adorno *et al.* (2023) em relação a bases com frames altamente similares.

As funções de perda (*loss functions*) utilizadas compreendem três componentes, operando de forma conjunta para otimizar simultaneamente a localização e a classificação dos objetos detectados: (i) a *Box Loss* (baseada na *Complete Intersection over Union, CIoU*), que penaliza desvios na posição e nas dimensões das caixas delimitadoras previstas em relação às anotações de referência (*ground truth*); (ii) a *Objectness Loss* (baseada em *Binary Cross-Entropy*), que avalia a confiança da rede na presença efetiva de um objeto dentro de cada região candidata; e (iii) a *Classification Loss* (igualmente baseada em *Binary Cross-Entropy*), responsável pela correta atribuição da classe tipológica ao objeto detectado. A combinação ponderada dessas três componentes constitui a perda total do modelo, conforme formulação original de Redmon *et al.* (2016) e aprimoramentos subsequentes implementados nas versões YOLOv5 e YOLOv8.

Ao término de cada treinamento, os artefatos gerados foram automaticamente armazenados em diretórios estruturados no formato *runs/detect/<modelo>_wheelchair/* (para os modelos YOLOv8) e *yolov5/runs/train/<modelo>_wheelchair/* (para os modelos YOLOv5), e incluem: (i) o arquivo de pesos do melhor modelo (*best.pt*); (ii) o arquivo de pesos da última época

(*last.pt*); (iii) as curvas de perda e métricas de treinamento e validação por época; (iv) a matriz de confusão do subconjunto de validação; e (v) os *logs* completos de treinamento. O conjunto desses artefatos, organizado por modelo e variante, constitui o acervo documental que sustenta a análise comparativa apresentada no Capítulo 4.

Para garantir a reprodutibilidade integral dos experimentos, todos os comandos de instalação do ambiente, configuração de dependências, treinamento e exportação dos modelos foram registrados em *scripts Python* documentados, que serão disponibilizados em repositório público. Adicionalmente, todos os hiperparâmetros efetivamente utilizados em cada treinamento foram registrados automaticamente nos arquivos *args.yaml* (YOLOv8) e *opt.yaml* (YOLOv5) gerados pelas implementações de cada arquitetura, assegurando a rastreabilidade integral de eventuais ajustes pontuais. As versões exatas de todas as bibliotecas e *drivers* utilizados foram documentadas por meio do comando *pip freeze* e *armazenadas* em arquivo *requirements.txt*, permitindo a reconstrução idêntica do ambiente de treinamento por pesquisadores independentes.

3.6.2 Critérios de Avaliação e Métricas

A avaliação do desempenho dos modelos treinados fundamenta-se em um conjunto de métricas quantitativas consolidadas na literatura de detecção de objetos, selecionadas por sua capacidade de capturar dimensões complementares da qualidade do sistema assistivo: a acurácia da classificação tipológica, a precisão da localização espacial do equipamento e a eficiência computacional em *hardware* embarcado. A adoção de métricas padronizadas assegura a comparabilidade dos resultados tanto entre os quatro modelos testados nesta pesquisa (YOLOv5n, YOLOv5s, YOLOv8n e YOLOv8s) quanto em relação aos trabalhos correlatos discutidos na Seção 2.6.

As métricas de detecção empregadas, fundamentadas nos critérios estabelecidos por Redmon *et al.* (2016) e adotadas como padrão pela comunidade YOLO, compreendem:

a) Precisão (*Precision*): representa a proporção de detecções corretas em relação ao total de detecções positivas emitidas pelo modelo. No contexto do sistema assistivo proposto, uma elevada precisão é essencial para minimizar falsos positivos, evitando que o sistema acione indevidamente mecanismos de acessibilidade, como

portas automáticas ou elevadores, na ausência de uma cadeira de rodas. Formalmente, a precisão é calculada como:

$$Precision = TP / (TP + FP)$$

em que *TP* (*True Positives*) representa as detecções corretas e *FP* (*False Positives*) as detecções incorretas.

b) *Revocação (Recall)*: mede a proporção de objetos de interesse efetivamente identificados pelo modelo em relação ao total de objetos presentes na cena. No sistema assistivo, uma alta revocação garante que o equipamento de mobilidade não seja ignorado, assegurando que todos os usuários de cadeira de rodas sejam contemplados pela automação do ambiente construído. É calculada como:

$$Recall = TP / (TP + FN)$$

em que *FN* (*False Negatives*) representa os objetos presentes na cena que o modelo falhou em detectar.

c) *F1-Score*: corresponde à média harmônica entre a precisão e a revocação, oferecendo uma medida sintética que equilibra ambos os indicadores. Essa métrica é particularmente relevante quando se deseja avaliar o desempenho global do sistema em cenários nos quais tanto os falsos positivos quanto os falsos negativos acarretam consequências indesejáveis para a experiência do usuário. É calculada como:

$$F1 = 2 \times (Precision \times Recall) / (Precision + Recall)$$

d) *Intersection over Union (IoU)*: indicador que quantifica o grau de sobreposição entre a caixa delimitadora prevista pelo modelo e a caixa de referência (*ground truth*) anotada manualmente. Valores de *IoU* superiores a 0,5 são convencionalmente considerados detecções válidas, enquanto limiares mais exigentes ($IoU \geq 0,75$) indicam localização espacial de alta qualidade, conforme ilustrado na Figura 5. No sistema proposto, a precisão da localização é determinante para que o gatilho de automação atue na zona espacial correta do ambiente construído.

e) *mAP@0.5* (*mean Average Precision a IoU $\geq 0,5$*): métrica-padrão para avaliação de detectores de objetos, calculada como a média das áreas sob as curvas de Precisão $\times$ Revocação de todas as classes. Nesta pesquisa, o *mAP@0.5* é a métrica primária de avaliação, por expressar de forma integrada a capacidade do modelo em detectar e classificar corretamente as onze tipologias de cadeiras de rodas. O limiar mínimo definido *a priori* para validação da hipótese operacional é de $mAP@0.5 \geq 0,85$, conforme estabelecido na Seção 3.1; o atendimento integral, parcial

ou não atendimento desse limiar pelos modelos avaliados é objeto de discussão no Capítulo 4.

f) $mAP@0.5:0.95$: versão mais rigorosa do mAP , calculada como a média do mAP em dez limiares de IoU (de 0,5 a 0,95 com incrementos de 0,05). Essa métrica penaliza modelos que localizam os objetos de forma imprecisa, fornecendo uma avaliação mais abrangente da qualidade da detecção. Embora não constitua critério primário de validação da hipótese, o $mAP@0.5:0.95$ é reportado para fins de comparabilidade com a literatura, especialmente com os resultados de Alruwaili *et al.* (2024) e Ali e Zhang (2024).

Além das métricas de detecção, a avaliação contempla indicadores de eficiência computacional, fundamentais para aferir a viabilidade da implantação no *Raspberry Pi*.

g) Velocidade de inferência (FPS, Frames Per Second): mede o número de quadros processados por segundo pelo modelo durante a fase de inferência. O limiar mínimo definido *a priori* é de 15 FPS no *Raspberry Pi*, conforme estabelecido na hipótese operacional (Seção 3.1), valor considerado adequado para aplicações de detecção em tempo real com latência imperceptível ao acionamento de dispositivos de acessibilidade. A medição de FPS foi realizada tanto na estação de trabalho do pesquisador quanto no dispositivo embarcado, permitindo a quantificação da degradação de desempenho decorrente da portabilidade do modelo para a arquitetura ARM.

h) Tempo de inferência por imagem (ms): indicador complementar ao FPS, que expressa a latência individual de cada predição em milissegundos. Essa métrica é particularmente relevante para avaliar a responsividade do sistema assistivo em cenários de acionamento pontual, nos quais uma única detecção bem-sucedida é suficiente para disparar o gatilho de automação.

i) Tamanho do modelo (MB): registra o tamanho do arquivo de pesos do modelo nos formatos *PyTorch* (.pt) e nos formatos otimizados para a arquitetura ARM gerados na fase de implantação no embarcado, fornecendo um indicador da demanda de *armazenamento* e da complexidade computacional. Modelos menores são preferíveis para dispositivos embarcados com restrições de memória e *armazenamento*, como o *Raspberry Pi*.

Considerando o desbalanceamento expressivo entre as onze classes do *corpus*, documentado na Seção 3.5.2, adotou-se nesta pesquisa uma estratégia

complementar de análise: além das métricas agregadas, o desempenho dos modelos foi reportado individualmente por classe, permitindo distinguir tipologias com volume amostral robusto, como Manual Dobrável, com 1.081 exemplares, de tipologias sub-representadas, como Hospitalar/Transporte, com apenas 14. Essa segmentação é essencial para evitar que o desempenho médio mascare deficiências localizadas em classes específicas, prática consolidada em pesquisas de detecção de objetos com distribuição assimétrica entre categorias (Adorno *et al.*, 2023; Alruwaili *et al.*, 2024).

A matriz de confusão, exemplificada na Figura 7 (B), constitui o instrumento central para a análise qualitativa das predições do modelo, permitindo a identificação de padrões de confusão entre tipologias morfológicamente similares, como cadeiras manuais dobráveis e hospitalares, ou cadeiras motorizadas e posturais. Essa análise por classe é fundamental para orientar eventuais refinamentos no banco de dados ou nos parâmetros de treinamento, e dialoga diretamente com o ciclo de coleta iterativa descrito na Seção 3.8, voltado à ampliação progressiva das classes sub-representadas.

O Quadro 8 sintetiza o conjunto de métricas adotadas, seus respectivos limiares de referência e a função de cada indicador na avaliação do sistema:

Quadro 8 - Síntese das métricas de avaliação

Métrica	Limiar Mínimo	Dimensão Avaliada	Função no Sistema Assistivo
Precision	Não se aplica	Acurácia	Minimizar acionamentos indevidos (falsos positivos)
Recall	Não se aplica	Cobertura	Garantir que nenhum equipamento seja ignorado
F1-Score	Não se aplica	Equilíbrio global	Avaliar o desempenho integrado entre precisão e revocação
IoU	$\geq 0,50$	Localização espacial	Assegurar precisão do posicionamento da detecção
mAP@0.5	$\geq 0,85$	Desempenho global	Critério primário de validação da hipótese operacional
mAP@0.5:0.95	Não se aplica	Desempenho rigoroso	Comparabilidade com literatura e avaliação da qualidade de localização
FPS	≥ 15	Eficiência computacional	Garantir detecção em tempo real no <i>Raspberry Pi</i>
Tempo de inferência	Não se aplica	Latência	Avaliar responsividade do acionamento de automação
Tamanho do modelo	Não se aplica	Portabilidade	Viabilizar <i>armazenamento</i> e execução no dispositivo embarcado

Fonte: Elaborado pelo Autor (2026).

A análise comparativa entre os modelos foi conduzida em duas etapas. Na primeira etapa, os quatro modelos treinados (YOLOv5n, YOLOv5s, YOLOv8n e YOLOv8s) foram avaliados no subconjunto de validação mediante o cálculo das métricas de detecção (*Precision*, *Recall*, *F1-Score*, *mAP@0.5* e *mAP@0.5:0.95*) e da análise de desempenho por classe, utilizando os *scripts* de avaliação nativos das respectivas implementações, executados na estação de trabalho do pesquisador. Na segunda etapa, o modelo identificado como o de melhor relação entre acurácia agregada, desempenho por classe e eficiência computacional foi submetido aos procedimentos de conversão e otimização para a arquitetura *ARM* e a testes de inferência no *Raspberry Pi*, nos quais foram medidos os indicadores de eficiência computacional (*FPS*, latência e tamanho do modelo), conforme detalhado na Seção 3.7.

Essa abordagem em duas etapas permite identificar, de forma sistemática, o modelo que oferece o melhor equilíbrio entre acurácia de detecção e viabilidade computacional para execução em dispositivos embarcados, atendendo simultaneamente aos requisitos de precisão do sistema assistivo e às restrições de *hardware* do *Raspberry Pi*. A condução integral do processo em ambiente local, com *hardware* de uso doméstico, válida a premissa de que a cadeia completa de desenvolvimento, do treinamento à implantação, pode ser replicada por pesquisadores e desenvolvedores sem acesso a infraestruturas computacionais de alto custo, em consonância com o princípio de baixo custo e acessibilidade que orienta o trabalho.

3.7 IMPLEMENTAÇÃO NO DISPOSITIVO EMBARCADO

A implementação do modelo de detecção em *hardware* embarcado constitui a etapa de validação experimental do sistema assistivo proposto, na qual a viabilidade técnica de execução em tempo real, sob restrições reais de processamento e energia, é confrontada com os requisitos definidos *a priori* para a operação no ambiente construído. A plataforma escolhida foi o microcomputador *Raspberry Pi 5*, sucessor da família introduzida em 2012 pela *Raspberry Pi Foundation*, cuja proposta original era de democratizar o acesso à computação por meio de placa de baixo custo, formato compacto e arquitetura aberta, alinha-se diretamente ao princípio que orienta esta dissertação: o de que soluções assistivas eficazes não exigem, necessariamente,

infraestrutura computacional dispendiosa (Paiva; Moreira, 2014). A escolha do *Pi 5*, em detrimento das versões anteriores *Pi 3* e *Pi 4*, deve-se a três avanços decisivos para uma aplicação de visão computacional em tempo real: o processador *Broadcom BCM2712*, com quatro núcleos *ARM Cortex-A76* 64 bits a 2,4 GHz, representa um salto significativo de desempenho por núcleo em relação ao *Pi 4*; a presença de instruções *NEON FP16* (extensões vetoriais do conjunto *ARM* otimizadas para operações em ponto flutuante de meia precisão) viabiliza, quando exploradas por *frameworks* compatíveis, a inferência de redes neurais com latência reduzida e sem dependência de aceleradores externos; e a configuração de memória *LPDDR4X-4267* amplia a margem para o pré-processamento intensivo de imagens demandado pelos *pipelines* das arquiteturas *YOLO*.

Esta seção apresenta, em três subseções, a operacionalização completa da implantação. A Seção 3.7.1 descreve a plataforma embarcada e a configuração do ambiente de *software* adotado. A Seção 3.7.2 detalha o procedimento de conversão dos modelos treinados para formatos compatíveis com a arquitetura *ARM*, incluindo o percalço técnico da exportação inicial em precisão plena (*FP32*) e sua correção mediante reexportação em meia precisão (*FP16*). A Seção 3.7.3 apresenta os protocolos de *benchmark* empregados, os resultados comparativos entre os formatos avaliados e as otimizações incrementais a adoção do *framework* NCNN (*otimizado para processadores ARM, desenvolvido pela Tencent*) e *overclock* controlado do processador, que viabilizaram a operação do sistema dentro do limiar de 15 *FPS* estabelecido na hipótese operacional.

3.7.1 Plataforma Embarcada e Configuração do Ambiente

A plataforma adotada para a implantação do sistema assistivo foi o *Raspberry Pi 5* (Figura 9), adquirido em formato de *kit* comercial (*Kit Raspberry Pi 5 Essential, RoboCore*), que inclui a placa principal, fonte oficial *USB-C* de 27 W, case com *cooler* ativo e cartão *microSD* de 64 GB. Para a captura de imagens em tempo real, foi utilizada uma *webcam Logitech* com resolução máxima de 720p, conectada via *USB* (Figura 10). Como interface de operação no ambiente construído, foi acoplada uma tela *IPS touchscreen* de 7 polegadas com resolução nativa de 1024 × 600 pixels (Figura 11), que permite tanto a visualização das detecções em tempo real quanto a

interação direta com o sistema. O Quadro 9 sintetiza as especificações de *hardware* e do ambiente de *software* adotados.

Quadro 9 - Configuração do *Raspberry Pi 5* e do ambiente de *software*

Categoria	Componente	Especificação
Hardware	Plataforma	<i>Raspberry Pi 5 (kit Essential RoboCore)</i>
Hardware	Processador	<i>Broadcom BCM2712, quad-core ARM Cortex-A76 64 bits a 2,4 GHz (com <i>overclock</i> controlado para 2,8 GHz)</i>
Hardware	Memória RAM	4 GB LPDDR4X-4267
Hardware	Armazenamento	<i>microSD SanDisk 64 GB</i>
Hardware	Refrigeração	Cooler ativo oficial
Hardware	Captura de vídeo	<i>Webcam Logitech 720p (USB)</i>
Hardware	Display	Tela IPS <i>touchscreen</i> de 7", 1024 × 600 pixels
Hardware	Alimentação	Fonte oficial USB-C PD, 5 V / 5 A (27 W)
Software	Sistema operacional	<i>Raspberry Pi OS Bookworm 64 bits (kernel 6.12)</i>
Software	Linguagem	<i>Python 3.13.5</i>
Software	Frameworks de inferência	<i>NCNN 1.0 (FP16) e TensorFlow Lite via ai-edge-litert 2.1.3 (INT8)</i>
Software	Framework auxiliar	<i>Ultralytics 8.4.27</i>
Software	Bibliotecas auxiliares	<i>OpenCV headless, NumPy, Pillow, PyYAML, threading</i>

Fonte: Elaborado pelo Autor (2026).

A configuração inicial do dispositivo seguiu um fluxo padronizado que assegurou a reprodutibilidade integral do procedimento, conduzido integralmente por acesso remoto, sem necessidade de monitor, teclado ou mouse acoplados ao Pi. O detalhamento desse fluxo, gravação da imagem do sistema, configuração de acesso remoto e atualização inicial das bibliotecas, consta no Apêndice A.5.

Figura 9 - *Raspberry Pi 5*



Fonte: Acervo Pessoal Autor (2026).

Figura 10 - Webcam Logitech 720p



Fonte: Acervo Pessoal Autor (2026).

Figura 11 - Tela IPS *touchscreen* de 7 Polegadas 1024x600 pixels



Fonte: Acervo Pessoal Autor (2026).

A fim de isolar as bibliotecas Python do sistema assistivo em relação às bibliotecas globais do Raspberry Pi OS, criou-se um ambiente virtual dedicado, no qual foram instaladas, em variante compatível com a arquitetura ARM e sem aceleração por GPU, as dependências necessárias à execução dos modelos nos formatos TFLite e NCNN. A relação completa das dependências e dos comandos de instalação consta do Apêndice A.5.

Como última etapa da preparação da plataforma, o processador foi submetido a um *overclock* controlado, elevando a frequência nominal de 2,4 GHz para 2,8 GHz mediante edição do arquivo de configuração de inicialização `/boot/firmware/config.txt`

com os parâmetros *ARM_freq=2800* e *over_voltage=4*. A escolha por essa frequência específica, e não por valores mais agressivos, fundamenta-se na busca por um equilíbrio entre ganho de desempenho e estabilidade térmica sustentada no contexto operacional 24/7, estabilidade essa que foi posteriormente verificada por meio do teste de uma hora descrito na Seção 3.7.3, no qual a temperatura permaneceu dentro da faixa segura sem ativação sustentada de *throttling* (mecanismo de redução automática de frequência por temperatura). O *cooler* ativo oficial incluído no *kit*, em conjunto com a janela térmica do BCM2712, mostrou-se suficiente para manter a operação do sistema dentro dos limites projetados ao longo de toda a fase experimental.

3.7.2 Conversão dos Modelos para Arquitetura ARM

A operação eficiente de redes neurais convolucionais em dispositivos embarcados depende da conversão dos modelos treinados (originalmente *armazenados* em formato *PyTorch* (.pt)), para representações otimizadas para a arquitetura ARM. Essa transposição não constitui mera mudança de extensão de arquivo: envolve a reorganização interna da topologia da rede para uma representação numérica adequada às instruções nativas do processador, a eventual quantização dos pesos para precisão reduzida (de 32 bits para 16 ou 8 bits) e a serialização em estrutura binária compatível com o *framework* de inferência alvo. Nesta pesquisa, foram avaliados dois caminhos de conversão complementares: o formato *TensorFlow Lite* (.tflite), suportado nativamente pelas implementações YOLOv5 e YOLOv8 e amplamente utilizado em dispositivos móveis e embarcados; e o formato *NCNN*, *framework* de inferência desenvolvido pela *Tencent* e otimizado para processadores ARM com suporte às operações em ponto flutuante de meia precisão (FP16) por meio das instruções vetoriais NEON, conforme apresentado na Seção 3.7.1.

Por decisão metodológica de robustez, a conversão dos modelos para o formato *TFLite* foi conduzida na estação de trabalho do pesquisador, e não diretamente no *Raspberry Pi*. Essa escolha justifica-se pelas incompatibilidades entre as versões mais recentes do *TensorFlow* disponíveis no *Pi* (2.21.0) e as bibliotecas auxiliares de conversão *tf_keras* e *onnx2tf*, que apresentaram conflitos de dependência durante os testes preliminares de exportação. A estação de trabalho, por dispor de versões mais maduras e estáveis do *TensorFlow* (2.19.0) e do *Python*

(3.10.11) em ambiente virtual dedicado à conversão, distinto daquele empregado no treinamento (Python 3.13.9, Seção 3.6.1), executou de forma confiável os *pipelines* de exportação. Os arquivos resultantes foram, em seguida, transferidos para o *Pi* por meio do protocolo SCP (*Secure Copy Protocol*) sobre a rede local, prática que assegurou tanto a integridade dos arquivos quanto o registro automático de *checksums* para verificação posterior. A conversão para *NCNN*, em contraste, foi conduzida diretamente no próprio *Pi*, conforme descrito mais adiante. Os comandos exatos empregados em cada exportação estão reunidos no Apêndice A.

Procedimento de exportação dos modelos YOLOv8

Para os modelos *YOLOv8n* e *YOLOv8s*, treinados a partir da biblioteca *Ultralytics*, a exportação para *TFLite* foi realizada pelo método nativo `export()` da própria biblioteca, que executa internamente um *pipeline* de conversão composto por três etapas sucessivas: a transposição inicial do formato *PyTorch* para o *ONNX* (formato intermediário interoperável entre *frameworks* de aprendizado profundo), a posterior conversão para o *TensorFlow SavedModel* e, por fim, a serialização final em *TFLite*. Esse encadeamento é gerenciado automaticamente pela *Ultralytics*, eliminando a necessidade de intervenção manual em cada estágio. Para cada modelo, foram geradas duas variantes:

a) *TFLite FP32* (precisão plena): preserva a precisão numérica original dos pesos em ponto flutuante de 32 bits, servindo como referência para avaliação de eventual perda de precisão imposta pela quantização.

b) *TFLite INT8* (quantização integral pós-treinamento): reduz a representação dos pesos de 32 para 8 bits inteiros, com ganho expressivo de velocidade de inferência ao custo de uma potencial perda de precisão. A calibração da quantização foi realizada automaticamente pelo *Ultralytics*, que utiliza um conjunto interno de 20 imagens em resolução 128×128 pixels para mapear as faixas dinâmicas das ativações da rede ao intervalo de inteiros disponível.

Procedimento de exportação dos modelos YOLOv5

Os modelos *YOLOv5n* e *YOLOv5s*, treinados a partir do repositório oficial original (não da biblioteca *Ultralytics* consolidada para o *YOLOv8*), exigiram procedimento distinto, dada a incompatibilidade entre os formatos internos de pesos das duas implementações. A exportação foi conduzida por meio do script `export.py` nativo do repositório *YOLOv5*, nas variantes *FP32* e *INT8*. O conjunto de calibração empregado pelo *YOLOv5* nesse procedimento, distinto do utilizado pela *Ultralytics* no

caminho do *YOLOv8*, viria a se mostrar fonte de degradação severa da precisão da variante *INT8*, conforme detalhado na análise dos resultados apresentada na Seção 3.7.3.

Exportação para o formato *NCNN*

Após a primeira rodada de *benchmark* dos modelos *TFLite* no *Raspberry Pi* (cujos resultados serão apresentados na Seção 3.7.3) ter identificado o *YOLOv8n* como candidato preferencial para a implantação operacional, este modelo foi adicionalmente exportado para o formato *NCNN*, em meia precisão (FP16). A exportação para *NCNN* foi conduzida diretamente no *Raspberry Pi* (em contraste com a conversão para *TFLite*, que exigiu a estação de trabalho), porque o *NCNN* é distribuído como pacote *Python* autocontido, instalável via *pip*, e dispensa as bibliotecas auxiliares conflitantes do *pipeline TensorFlow*. O procedimento gerou um par de arquivos característico do *framework*: um arquivo *.param*, contendo a topologia da rede, e um arquivo *.bin*, contendo os pesos quantizados em meia precisão. O Quadro 10 sintetiza a totalidade dos modelos gerados nesta etapa de conversão, registrando, para cada um, o formato e a variante de precisão e o tamanho do arquivo resultante. Esta última coluna é relevante porque expressa, de forma imediata, o efeito da quantização sobre a compactação do modelo: a redução do número de bits empregado na representação dos pesos diminui o espaço de armazenamento exigido, fator crítico para a implantação em um dispositivo embarcado de memória limitada como o *Raspberry Pi*.

Quadro 10 - Modelos exportados para a arquitetura *ARM*

Modelo de Origem	Formato e variante	Tamanho do arquivo	Observação
<i>YOLOv5n</i>	<i>TFLite FP32</i>	3,5 MB	Referência de precisão
<i>YOLOv5n</i>	<i>TFLite INT8</i>	1,9 MB	Quantização integral pós-treinamento
<i>YOLOv5s</i>	<i>TFLite FP32</i>	13,5 MB	Referência de precisão
<i>YOLOv5s</i>	<i>TFLite INT8</i>	7,1 MB	Quantização integral pós-treinamento
<i>YOLOv8n</i>	<i>TFLite FP32</i>	11,7 MB	Referência de precisão
<i>YOLOv8n</i>	<i>TFLite INT8</i>	3,2 MB	Quantização integral pós-treinamento
<i>YOLOv8s</i>	<i>TFLite FP32</i>	42,7 MB	Referência de precisão
<i>YOLOv8s</i>	<i>TFLite INT8</i>	11,0 MB	Quantização integral pós-treinamento
<i>YOLOv8n</i>	<i>NCNN FP16</i>	5,9 MB	Modelo selecionado para implantação operacional

Fonte: Elaborado pelo Autor (2026).

Durante a validação do sistema embarcado, observou-se uma discrepância entre o desempenho de inferência nos *benchmarks* sintéticos do NCNN (cerca de 14 FPS) e o desempenho real com a webcam ativa (8,5 FPS), sem causa aparente. A investigação diagnóstica identificou que o modelo havia sido exportado silenciosamente em precisão plena (FP32), e não em meia precisão (FP16), em razão da omissão do parâmetro explícito de exportação; a reexportação correta reduziu o arquivo de 12 MB para 5,9 MB e restaurou a latência média a cerca de 71 ms (14 FPS). O episódio fica registrado como alerta metodológico: a precisão de qualquer exportação para formatos otimizados deve ser verificada explicitamente, seja pela inspeção dos metadados, seja pela conferência do tamanho do arquivo gerado. O roteiro diagnóstico completo consta do Apêndice A.6.

3.7.3 Protocolos de Teste no *Raspberry Pi*

A avaliação experimental do desempenho dos modelos no *Raspberry Pi* foi conduzida por meio de um conjunto estruturado de protocolos de *benchmark*, projetados para isolar progressivamente cada variável que poderia influenciar a velocidade real de inferência em condições operacionais. O processo foi organizado em três níveis sucessivos de aproximação ao cenário de uso real: *benchmarks* sintéticos com imagens artificiais, para caracterizar o desempenho intrínseco de cada modelo; *benchmarks* com captura de vídeo em tempo real, para mensurar o impacto da latência da câmera e da concorrência entre *threads*; e teste de estabilidade térmica de longa duração, para verificar a sustentabilidade da configuração selecionada em regime de operação contínua. Todos os procedimentos foram conduzidos no *Raspberry Pi 5* descrito na Seção 3.7.1, sob as mesmas condições ambientais, com o *cooler* ativo permanentemente ligado e o *governor* da *CPU* configurado em modo *performance*.

Protocolo de mensuração de latência e FPS

O *benchmark* sintético foi adotado como ponto de partida por sua reprodutibilidade controlada: a entrada do modelo é constituída por uma imagem gerada artificialmente, com dimensões correspondentes à resolução nativa esperada (geralmente $1 \times 640 \times 640 \times 3$) e tipo de dado adequado a cada formato (*float32* para variantes *FP32* e *FP16*, *uint8* para variantes *INT8*). O protocolo, aplicado de forma idêntica a todos os modelos avaliados, compreendeu cinco etapas sequenciais.

Inicialmente, o modelo foi carregado em memória e os tensores internos foram alocados pelo *runtime* de inferência. Em seguida, foram executadas cinco inferências de aquecimento (*warm-up*), cujos tempos foram descartados das medições, com o objetivo de estabilizar os caches do processador e as otimizações dinâmicas do *runtime*. Após o aquecimento, foram realizadas 50 inferências cronometradas individualmente por meio da função *time.perf_counter()* da biblioteca padrão do *Python*, que oferece resolução de nanossegundos no *Raspberry Pi OS*. A latência média foi obtida pela média aritmética das 50 medições, e a taxa de quadros por segundo (*FPS*) foi calculada como o inverso da latência ($FPS = 1000 / \text{latência_média_ms}$). Para os modelos *TFLite*, o *runtime* utilizado foi o *ai-edge-litert 2.1.3* com o *delegate XNNPACK*, que aplica otimizações específicas para *CPUs ARM*. Para o modelo *NCNN*, o *benchmark* foi conduzido por meio da interface da própria biblioteca *Ultralytics*, que dialoga diretamente com o *NCNN* como *backend* de inferência.

Protocolo de captura de vídeo em tempo real

Após a caracterização sintética, foi necessário verificar se os ganhos de desempenho observados em condições controladas se manteriam quando o modelo operasse com captura real da *webcam Logitech 720p*, cenário em que a latência de aquisição de cada quadro se soma à latência de inferência. Foram implementadas duas abordagens distintas para essa avaliação. Na primeira, denominada captura sequencial, a aquisição de quadros pela *webcam* e a inferência sobre cada quadro ocorrem no mesmo *thread* de execução, de modo que cada nova predição só pode iniciar após a conclusão da captura anterior. Na segunda, denominada captura paralela, a aquisição é delegada a um *thread* dedicado (instanciado via *threading.Thread* da biblioteca padrão do *Python*), enquanto o *thread* principal executa a inferência sobre o último quadro disponibilizado pelo *thread* de captura. Essa segunda configuração permite que os tempos de captura e inferência se sobreponham temporalmente, eliminando a soma direta das duas latências e elevando significativamente o *FPS* efetivo do sistema, conforme detalhado na análise dos resultados.

Protocolo de teste com *display* ativo

Para simular as condições reais de operação do sistema acoplado a uma interface visual de 7 polegadas (1024 × 600 pixels), foi adicionada ao protocolo a renderização das *bounding boxes* sobre cada quadro processado, por meio da função

cv2.imshow do *OpenCV*. Essa etapa, embora pareça apenas decorativa, introduz uma carga computacional adicional não desprezível, pois envolve a composição gráfica do quadro processado e a sua transferência para o *buffer* de vídeo do sistema operacional. Para isolar essa variável de forma controlada, foi utilizado um *display* virtual mediante a ferramenta *Xvfb* (*X Virtual Framebuffer*), configurada para emular uma tela de 1024 × 600 pixels com 24 bits de profundidade de cor, equivalente em recursos à tela física de produção. A introdução do *display* permitiu identificar uma contenção entre o *thread* de inferência e o compositor gráfico do sistema operacional, cuja análise é apresentada adiante nesta subseção.

Protocolo de mensuração de precisão (*mAP@0.5*)

A avaliação de precisão dos modelos quantizados, expressa pelo *mAP@0.5*, foi conduzida na estação de trabalho do pesquisador (e não no *Raspberry Pi*), utilizando o conjunto de validação descrito na Seção 3.5.2 (394 imagens, 457 instâncias anotadas, distribuídas nas 11 classes tipológicas). A escolha pela estação de trabalho como ambiente de cálculo do *mAP* é metodologicamente equivalente à mensuração no dispositivo embarcado, uma vez que a precisão de detecção depende exclusivamente dos pesos do modelo e do formato numérico da representação, e não da plataforma de execução. As métricas foram calculadas pelas rotinas de validação nativas do *Ultralytics* (para os modelos *YOLOv8*) e do repositório oficial do *YOLOv5* (para a família *YOLOv5*), garantindo a comparabilidade direta com os resultados obtidos durante o treinamento (Seção 3.6) e relatados nos arquivos *results.csv* de cada execução.

Resultados da primeira rodada de *benchmark* (*TFLite*)

Aplicado o protocolo descrito acima sobre os oito modelos *TFLite* gerados na Seção 3.7.2, foram obtidos os resultados sintetizados no Quadro 11. As medições foram realizadas com o processador operando em sua frequência nominal (2,4 GHz), antes da aplicação do *overclock*, para que o *baseline* de cada formato pudesse ser caracterizado em condições padrão.

Quadro 11 - *Benchmark* de inferência *TFLite* no *Raspberry Pi 5* (CPU, *XNNPACK*, 2,4 GHz)

Modelo	Variante	Tamanho	Latência média	FPS	mAP@0.5
<i>YOLOv5n</i>	<i>FP32</i>	3,5 MB	174,4 ms	5,7	0,599
<i>YOLOv5n</i>	<i>INT8</i>	1,9 MB	48,0 ms	20,9	0,001*
<i>YOLOv5s</i>	<i>FP32</i>	13,5 MB	575,2 ms	1,7	0,638
<i>YOLOv5s</i>	<i>INT8</i>	7,1 MB	143,0 ms	7,0	0,001*
<i>YOLOv8n</i>	<i>FP32</i>	11,7 MB	290,3 ms	3,4	0,670
<i>YOLOv8n</i>	<i>INT8</i>	3,2 MB	164,6 ms	6,1	0,670
<i>YOLOv8s</i>	<i>FP32</i>	42,7 MB	958,4 ms	1,0	0,652
<i>YOLOv8s</i>	<i>INT8</i>	11,0 MB	403,8 ms	2,5	0,651

Fonte: Elaborado pelo Autor (2026).

(*): modelos com colapso de precisão após quantização *INT8*.

A análise dos resultados revelou três comportamentos distintos. O primeiro foi o colapso da precisão das variantes *INT8* dos modelos *YOLOv5*, cujo *mAP@0.5* caiu para valores próximos a zero (0,001), embora a velocidade de inferência tenha se elevado expressivamente (20,9 *FPS* no *YOLOv5n* e 7,0 *FPS* no *YOLOv5s*, contra 5,7 e 1,7 *FPS* nas variantes *FP32* correspondentes). Essa combinação de alta taxa de quadros e precisão praticamente nula tornou os dois modelos inviáveis para uso prático, uma vez que, embora as inferências ocorressem com baixa latência, a rede deixou de produzir detecções com confiança suficiente para ser considerada funcional. A degradação está possivelmente associada à combinação de dois fatores: o conjunto de calibração interno utilizado pelo *script export.py* do *YOLOv5*, composto por imagens genéricas que podem não capturar adequadamente a distribuição de ativações de uma rede treinada para classes específicas como cadeiras de rodas; e a sensibilidade da arquitetura *YOLOv5* às transformações numéricas da quantização integral, particularmente nas camadas finais responsáveis pela predição das caixas delimitadoras e das probabilidades de classe. Os modelos *YOLOv8*, em contraste, preservaram a precisão com perda marginal após a quantização *INT8* (de 0,670 para 0,670 no *YOLOv8n* e de 0,652 para 0,651 no *YOLOv8s*, perdas inferiores a 0,2%), comportamento que se atribui ao *pipeline* de exportação adotado pela biblioteca *Ultralytics*, que opera em conjunto com as bibliotecas auxiliares *onnx* e *onnx2tf* e que, na prática, produziu maior estabilidade numérica ao longo da conversão.

O segundo comportamento foi a ineficiência computacional dos modelos *YOLOv8s* mesmo em sua variante *INT8*, que atingiu apenas 2,5 *FPS*, valor demasiadamente distante do limiar de 15 *FPS* estabelecido na hipótese operacional.

Esse resultado, em conjunto com o colapso *INT8* dos *YOLOv5*, deixou apenas uma alternativa viável entre os oito modelos *TFLite* avaliados: o *YOLOv8n INT8*, com 6,1 *FPS* e *mAP* de 0,670 sem degradação. Esse modelo, contudo, continuava abaixo do limiar de 15 *FPS*, indicando que a quantização *INT8* isoladamente não seria suficiente para atender ao requisito de operação em tempo real.

O terceiro comportamento, mais sutil e ainda assim relevante, foi a confirmação de que a redução do tamanho do arquivo proporcionada pela quantização (de 11,7 MB para 3,2 MB no *YOLOv8n* e de 42,7 MB para 11,0 MB no *YOLOv8s*, reduções respectivas de 72,6% e 74,2%) é benéfica para o *armazenamento* e a portabilidade do sistema, ainda que a velocidade de inferência permaneça insuficiente. Esse achado reforçou a necessidade de buscar alternativas adicionais ao *TFLite* como mecanismo de aceleração da inferência.

Otimizações incrementais sobre o *YOLOv8n*

Diante da limitação observada, o modelo *YOLOv8n* (selecionado por reunir a melhor combinação de *mAP@0.5* e tamanho compacto) foi submetido a duas otimizações incrementais sucessivas. A primeira consistiu na substituição do *runtime* de inferência do *TFLite* com quantização *INT8* para o *NCNN* com pesos em *FP16*, conforme descrito na Seção 3.7.2. A segunda consistiu no *overclock* controlado da *CPU*, elevando a frequência de operação de 2,4 GHz para 2,8 GHz, conforme descrito na Seção 3.7.1. Os resultados sintetizados no Quadro 12 demonstram o ganho cumulativo dessas duas intervenções.

Quadro 12 - Evolução do desempenho do *YOLOv8n* com otimizações incrementais (*benchmark* sintético, 640 × 640)

Configuração	<i>Runtime</i>	Frequência da <i>CPU</i>	Latência média	<i>FPS</i>	Ganho relativo
Linha de base	<i>TFLite INT8</i>	2,4 GHz	164,6 ms	6,1	referência
Otimização 1	<i>NCNN FP16</i>	2,4 GHz	75,6 ms	13,2	2,2 vezes
Otimização 2	<i>NCNN FP16</i>	2,8 GHz	66,1 ms	15,1	2,5 vezes

Fonte: Elaborado pelo Autor (2026).

A primeira otimização, isoladamente, foi responsável pelo maior ganho individual de desempenho registrado no estudo (2,2 vezes em relação à linha de base; o ganho de 2,5 vezes registrado no Quadro 12 para a Otimização 2 é cumulativo, e não um ganho isolado), elevando o *FPS* de 6,1 para 13,2 sem qualquer alteração de *hardware*. Esse resultado decorre do fato de o *NCNN* explorar nativamente as instruções vetoriais *NEON FP16* do *Cortex-A76*, executando operações de

convolução e de produto escalar em paralelo sobre múltiplos elementos de 16 bits, enquanto o *TFLite* com *XNNPACK*, embora otimizado para *ARM*, opera predominantemente em 32 bits nas variantes *FP32* ou em 8 bits nas variantes *INT8* (com sobrecargas de quantização e desquantização nas bordas das camadas). A diferença arquitetural entre os dois caminhos de inferência traduziu-se diretamente em vantagem mensurável de *throughput*. A segunda otimização, o *overclock* para 2,8 GHz, contribuiu com um ganho adicional de aproximadamente 14% (de 13,2 para 15,1 *FPS*), suficiente para ultrapassar o limiar mínimo de 15 *FPS* estabelecido na hipótese operacional, ainda que por margem estreita.

Resultados em cenários reais com câmera e *display*

Os resultados da otimização final foram, em seguida, validados em condições mais próximas da operação real. O Quadro 13 apresenta o desempenho do *YOLOv8n NCNN FP16* a 2,8 GHz em três configurações progressivamente mais realistas: captura de imagem por *webcam* sem paralelismo, captura por *webcam* com *thread* dedicado, e o cenário completo com *thread* de captura combinada à renderização em *display*.

Quadro 13 - Desempenho do *YOLOv8n NCNN FP16* com câmera real e *display* (640×640, 2,8 GHz)

Configuração	Tempo de captura	Latência de inferência	Tempo total por quadro	<i>FPS</i> efetivo
Câmera sequencial (sem <i>thread</i>)	66,7 ms	68,3 ms	134,9 ms	7,4
Câmera paralela (com <i>thread</i>)	(sobreposto)	69,3 ms	69,3 ms	14,4
Câmera paralela com <i>display</i>	(sobreposto)	75,3 ms	75,3 ms	13,3

Fonte: Elaborado pelo Autor (2026).

A captura sequencial revelou uma queda expressiva do *FPS* efetivo (de 15,1 para 7,4), uma vez que o tempo de aquisição da *webcam* (cerca de 67 ms) somava-se diretamente à latência de inferência, dobrando o tempo total por quadro. A simples introdução de um *thread* dedicado à captura, executando em paralelo com a inferência, restaurou o desempenho a 14,4 *FPS*, valor próximo ao máximo teórico do *benchmark* sintético. A adição da renderização em *display* virtual reduziu o *FPS* para 13,3, queda de aproximadamente 7% atribuída à sobrecarga de composição gráfica no compositor do *X server*. Esse último valor, embora ligeiramente abaixo do limiar

formal de 15 *FPS*, foi considerado adequado à aplicação proposta, pois corresponde a uma latência por quadro de cerca de 75 ms, amplamente compatível com a velocidade de deslocamento de usuários de cadeira de rodas em uso cotidiano, mensurada por Pinnock Branford *et al.* (2025) em $0,43 \pm 0,16$ m/s, e com o tempo de resposta esperado para o acionamento de dispositivos de acessibilidade como portas automáticas e elevadores.

Durante a validação com o display ativo, observou-se um comportamento contraintuitivo: a abertura da janela de visualização do OpenCV elevava a latência de inferência e reduzia a taxa para cerca de 10 *FPS*. Um experimento controlado, variando o número de *threads* do OpenMP com a janela ativa e inativa, revelou que restringir o paralelismo a apenas dois núcleos preserva a latência mesmo com o display ativo, ao liberar os núcleos remanescentes do Cortex-A76 para o compositor gráfico do sistema. Consolidou-se, assim, como padrão dos três aplicativos (Seção 3.8), a limitação programática dos pools de paralelismo nativo a duas *threads* antes de qualquer importação dependente, o que restaurou a taxa sustentada de aproximadamente 14 *FPS* com a interface gráfica ativa. Os valores do experimento e as variáveis de ambiente ajustadas constam do Apêndice A.6.

Estabilidade térmica em operação contínua

Para validar a viabilidade da configuração selecionada em regime de operação 24 horas por dia, sete dias por semana (24/7), foi conduzido um teste de estresse de 60 minutos com inferência *YOLO* ininterrupta no *Raspberry Pi 5* a 2,8 GHz, utilizando a configuração final do sistema (*YOLOv8n NCNN FP16* com captura *threaded* e *OMP_NUM_THREADS=2*). Os resultados são apresentados no Quadro 14.

Quadro 14 - Resultados do teste de estabilidade térmica do sistema embarcado (60 minutos de inferência contínua)

Parâmetro	Resultado
Duração do teste	60 minutos
Total de inferências realizadas	53.133
Temperatura média da <i>CPU</i>	80–82 °C
Temperatura máxima registrada	82,3 °C
Frequência sustentada da <i>CPU</i>	2.800 MHz
<i>Throttling</i> térmico ativo	Nenhum sustentado
<i>FPS</i> médio mantido	14,7 a 14,9

Fonte: Elaborado pelo Autor (2026).

A temperatura atingiu equilíbrio térmico nos primeiros 15 a 20 minutos do teste, estabilizando entre 80 °C e 82 °C, valor situado abaixo do limiar de 85 °C a partir do

qual o BCM2712 ativa a redução automática de frequência (*throttling*). Houve uma única ocorrência pontual de redução breve do *clock* (para 2.766 MHz por instantes no minuto 37,5, registrada com a *flag 0x80008* do utilitário *vcgencmd*), com recuperação imediata e sem impacto sobre o *FPS* médio sustentado. A *flag 0x80000*, que indica que o limite de temperatura foi atingido em algum momento da execução, manteve-se ativada ao longo do teste, mas a ausência de redução sustentada de desempenho atesta a adequação do sistema de refrigeração ativo do *kit* comercial. A estabilização da temperatura em regime permanente permite inferir, por extrapolação termodinâmica, que o sistema manterá o mesmo desempenho em operação contínua de longo prazo, uma vez que as condições observadas após o equilíbrio térmico são representativas do estado estacionário.

Configuração final selecionada

O conjunto de procedimentos descritos nesta seção culminou na seleção da configuração operacional do sistema assistivo embarcado, sintetizada como segue: o modelo *YOLOv8n* exportado para *NCNN* com pesos em *FP16* e resolução de entrada de 640×640 pixels; o processador operando em 2,8 GHz com *cooler* ativo e *governor* em modo *performance*; a captura de vídeo conduzida em *thread* paralelo dedicado, dissociada da *thread* de inferência; e os pools de paralelismo nativo (*OpenMP*, *MKL*, *OpenBLAS*) restringidos a duas *threads* simultâneas para mitigar a contenção com o compositor gráfico. Essa configuração entrega *FPS* sustentado entre 13,3 (com *display* ativo) e 14,4 (sem *display*), com *mAP@0.5* preservado em 0,708 (ante 0,711 do modelo *PyTorch* original, queda de apenas 0,003 ponto absoluto, indicando ausência de perda significativa de precisão na exportação para *NCNN FP16*), e operação termicamente estável em ambiente ininterrupto. O atendimento ao limiar formal de 15 *FPS* estabelecido na hipótese operacional foi alcançado em condições de *benchmark* sintético (15,1 *FPS*), e o ligeiro déficit observado em cenário real (13,3 *FPS* com *display*) é discutido em detalhe no Capítulo 4, juntamente com as implicações funcionais para a aplicação no ambiente construído.

3.8 SISTEMA OPERACIONAL EMBARCADO

A implantação do modelo de detecção descrita na Seção 3.7 representa apenas uma parte do sistema assistivo embarcado. Para que a inferência cumpra sua função no ambiente construído, isto é, gere dados utilizáveis para automação, demonstre a

superioridade do modelo treinado em relação a soluções genéricas, e alimente um ciclo de aprimoramento iterativo do próprio sistema, é necessário um conjunto de aplicativos que organizem a operação cotidiana do dispositivo. Esta seção descreve a arquitetura modular dos três aplicativos desenvolvidos para o *Raspberry Pi 5*: o Aplicativo 1, voltado à operação de produção, com registro estruturado de eventos; o Aplicativo 2, que executa o modelo *YOLOv8n* pré-treinado no conjunto COCO como referência comparativa para fins demonstrativos; e o Aplicativo 3, que implementa um ciclo de coleta iterativa para retreinamento progressivo do modelo. A separação em três aplicativos independentes constitui uma decisão metodológica deliberada, articulando uma estratégia de triangulação no sentido proposto por Marconi e Lakatos (2017): a operação de produção do Aplicativo 1 fornece a dimensão quantitativa da validação, expressa em séries temporais de detecções, latências e tipologias registradas em arquivos estruturados; o Aplicativo 2 oferece a contraparte comparativa qualitativa, ao colocar lado a lado o desempenho do modelo especialista e o de uma referência genérica amplamente difundida na literatura; e o Aplicativo 3 introduz a dimensão iterativa, na qual os dados gerados em operação real retroalimentam progressivamente o próprio sistema, expandindo o *corpus* de treinamento e elevando, em iterações futuras, a precisão do modelo. Essa tríade isola as funções de produção, demonstração e coleta, evita o acoplamento indesejado entre comportamentos com finalidades distintas, e permite a evolução independente de cada componente ao longo da continuidade da pesquisa.

3.8.1 Arquitetura modular e o módulo comum

Os três aplicativos compartilham um módulo *Python* comum, denominado *detector_core.py*, que concentra as rotinas operacionais reutilizáveis: o carregamento de modelos no formato *NCNN* ou *PyTorch*, a abertura e a configuração da *webcam* por meio do *OpenCV*, o desenho das *bounding boxes* e dos rótulos de classe sobre cada quadro, a renderização de um HUD (do termo em inglês *Head-Up Display*, sobreposição informativa não intrusiva à imagem), e a contagem incremental de *FPS*. Esse módulo implementa também as configurações de paralelismo descritas na Seção 3.7.3, definindo as variáveis de ambiente *OMP_NUM_THREADS*, *MKL_NUM_THREADS* e *OPENBLAS_NUM_THREADS* para o valor 2 antes de qualquer importação dependente, mitigando assim a contenção entre o *pool* de

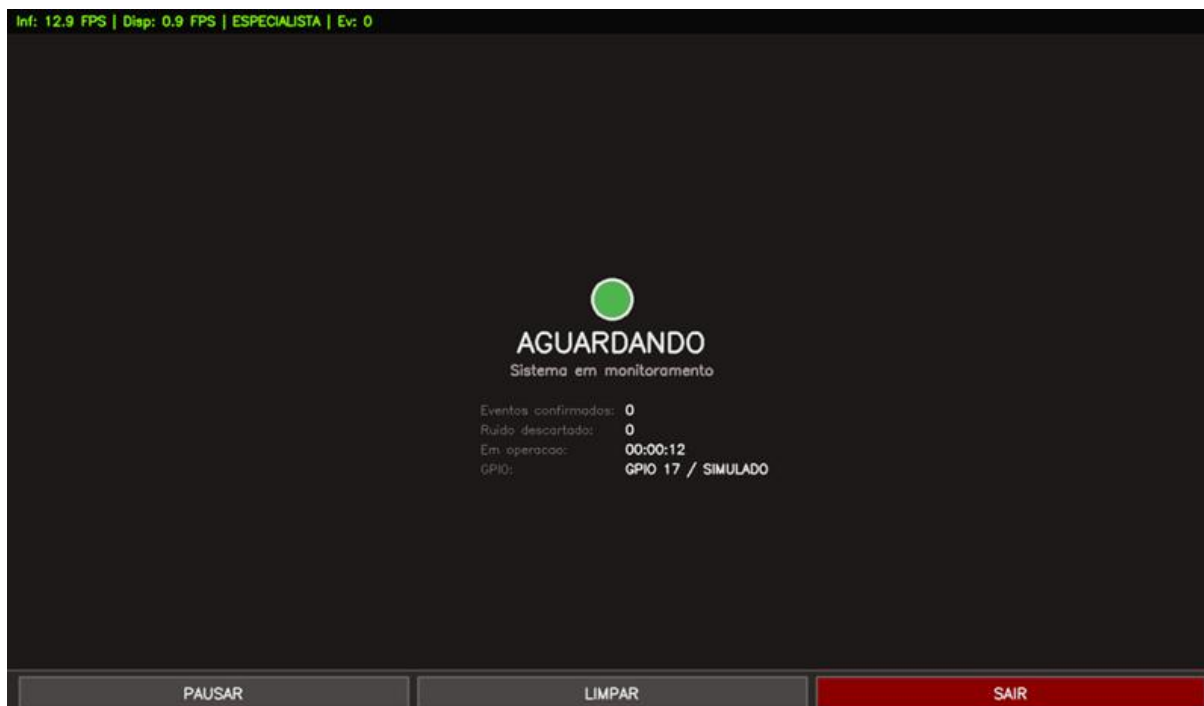
threads do *OpenMP* e o compositor gráfico do *Raspberry Pi OS*, conforme detalhado na subseção referida.

A modularização atende a três objetivos complementares. Em primeiro lugar, reduz a duplicação de código entre os aplicativos, simplificando a manutenção: correções e melhorias no carregamento do modelo, na captura da *webcam* ou na renderização do *HUD* propagam-se automaticamente aos três programas, sem necessidade de alteração em múltiplos arquivos. Em segundo lugar, explicita a separação entre lógica comum e lógica específica: cada aplicativo concentra-se exclusivamente nas decisões que o diferenciam dos demais, mantendo o código mais legível e auditável. Em terceiro lugar, facilita a expansão futura do sistema com novos aplicativos especializados, que podem ser construídos a partir do mesmo módulo de base, herdando todas as garantias operacionais já validadas para os três originais.

3.8.2 Aplicativo 1: Detector Especialista com Registro de Eventos

O Aplicativo 1 constitui o módulo operacional de produção do sistema assistivo. Executa o modelo *YOLOv8n NCNN FP16*, selecionado e validado nas Seções 3.7.2 e 3.7.3, sobre o fluxo de imagens capturado em tempo real pela *webcam Logitech*, a uma taxa sustentada de aproximadamente 14 *FPS* no *Raspberry Pi 5* com *overclock* para 2,8 GHz. As detecções são apresentadas na tela touchscreen de 7 polegadas (1024 × 600 *pixels*) por meio do desenho das *bounding boxes* e dos rótulos das tipologias detectadas, oferecendo retorno visual imediato ao operador. A Figura 12 ilustra a interface do Aplicativo 1 em operação, no estado de espera (*AGUARDANDO*), em que o sistema monitora continuamente a cena sem detecções ativas. A barra superior exibe as taxas de inferência e de exibição, o modo especialista e o contador de eventos, enquanto o painel central reúne os indicadores de eventos confirmados, ruído descartado e tempo em operação, além do estado da saída GPIO em modo simulado.

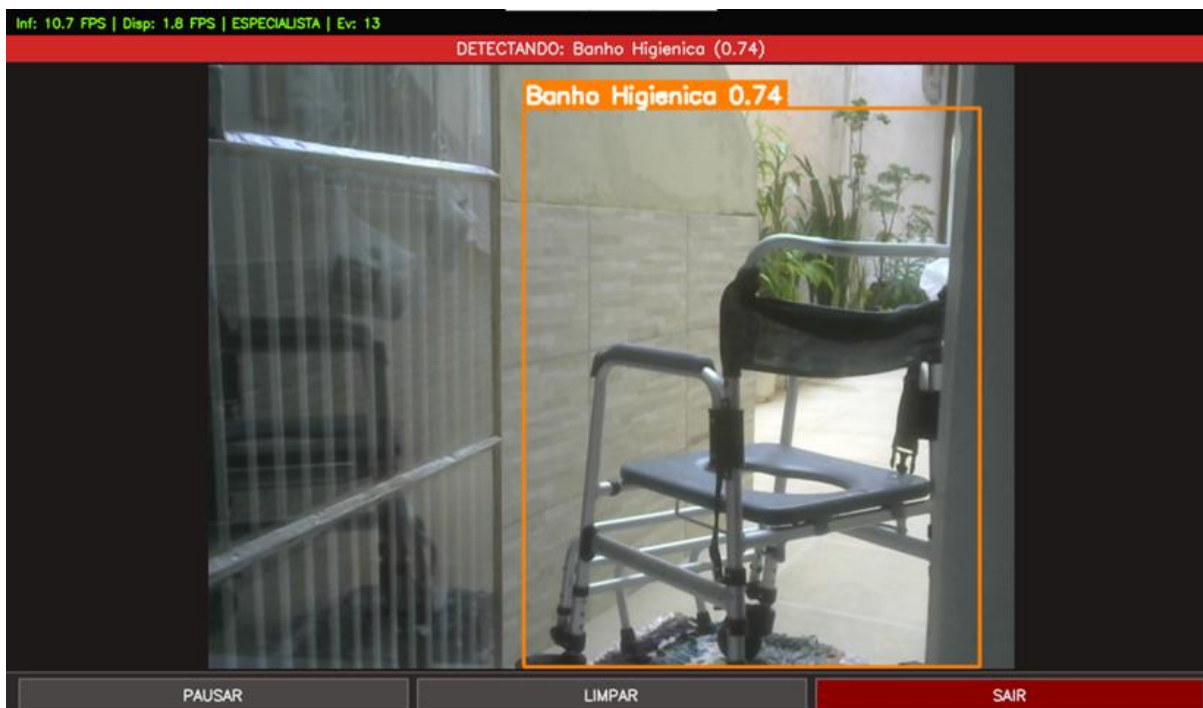
Figura 12 - Aplicativo Especialista em modo “Aguardando”.



Fonte: Acervo Pessoal Autor (2026).

Para conciliar a alta taxa de inferência com a renderização gráfica em tempo real, o aplicativo adota uma arquitetura orientada a eventos (*event-driven*) com desacoplamento entre dois *threads* principais. O *thread* de inferência opera continuamente sobre os quadros capturados pela *webcam*, executando o modelo a aproximadamente 14 *FPS*, enquanto o *thread* de exibição atualiza a tela apenas durante eventos ativos de detecção, a uma frequência reduzida de aproximadamente 2 *FPS*. Essa arquitetura elimina a contenção observada entre inferência e compositor gráfico, descrita na Seção 3.7.3, e garante que a velocidade de detecção, variável crítica para o acionamento de dispositivos de acessibilidade, não seja sacrificada em nome da fluidez visual da interface, variável menos sensível ao tempo de resposta humana. Na Figura 13 pode-se observar o Aplicativo Especialista funcionando e realizando a detecção de uma cadeira de banho.

Figura 13 - Aplicativo Especialista em modo de Detecção.



Fonte: Acervo Pessoal Autor (2026).

O sistema de registro de operação foi projetado para permitir auditoria detalhada do funcionamento do aplicativo sem violar o princípio ético de volatilidade de dados estabelecido nos capítulos anteriores. Em vez de persistir os quadros capturados, o que armazenaria imagens identificáveis dos usuários e contrariaria o princípio adotado, o aplicativo extrai os metadados de cada detecção e descarta o *buffer* de quadro imediatamente após a inferência. Os metadados são gravados em dois arquivos no formato CSV gerados diariamente, com granularidades complementares: um arquivo nomeado *frames_AAAA-MM-DD.csv*, com granularidade quadro a quadro, contendo o *timestamp*, a classe detectada, a confiança e as coordenadas normalizadas da *bounding box* para cada inferência positiva; e um arquivo *eventos_AAAA-MM-DD.csv*, com granularidade agregada por evento, contendo o *timestamp* de início e fim, a classe, a duração total, a confiança média e máxima, e o número de quadros que compõem cada evento.

Define-se evento, no contexto deste aplicativo, como uma sequência contínua de detecções da mesma classe, com tolerância para lacunas de até 2 segundos (equivalentes a aproximadamente 28 quadros consecutivos a 14 FPS). Essa estrutura agregada produz uma série temporal sintética adequada à análise estatística posterior em ferramentas como pandas, R ou planilhas eletrônicas, sem sobrecarregar o

sistema com a persistência de imagens. Em conformidade com o requisito de privacidade descrito no Capítulo 1 e operacionalizado na Seção 3.5.1, nenhuma imagem capturada é gravada em disco em nenhum momento da execução do aplicativo.

O Aplicativo 1 foi adicionalmente preparado, no nível de *software*, para emitir sinais por meio das pinagens de *GPIO* (*General Purpose Input/Output*) do *Raspberry Pi*, viabilizando a integração futura com mecanismos físicos de automação do ambiente construído, tais como relés para portas automáticas, controles de iluminação ou atuadores de elevadores. No estágio atual da pesquisa, esses sinais são apenas registrados no *logbook* do aplicativo, sem efetivação de ações físicas. Essa decisão metodológica separa a validação da capacidade de detecção da efetivação de ações no ambiente, permitindo que o sistema seja avaliado em diferentes cenários sem o risco de acionamentos involuntários durante a fase de validação experimental.

3.8.3 Aplicativo 2: Detector Genérico para Validação Comparativa

O Aplicativo 2 executa o modelo *YOLOv8n* pré-treinado no conjunto COCO, com 80 classes genéricas (Lin *et al.*, 2017), e cumpre função estritamente demonstrativa e comparativa. Sua finalidade é evidenciar, em contexto de apresentação ou de defesa, a vantagem do modelo especialista treinado nesta pesquisa sobre uma aplicação direta de modelo pré-treinado genérico ao mesmo problema.

Em testes operacionais conduzidos durante o desenvolvimento, observou-se que o modelo COCO genérico reconhece cadeiras de rodas predominantemente como objetos das classes "*person*" ou "*chair*", sem qualquer granularidade tipológica. Essa observação reforça a contribuição científica da classificação consolidada no Quadro 5: a tarefa de detecção e classificação tipológica de cadeiras de rodas requer tanto um *corpus* específico, com anotações orientadas por uma taxonomia robusta, quanto um treinamento dirigido, condições ambas atendidas pelo modelo especialista do Aplicativo 1. A coexistência dos dois aplicativos no mesmo *Raspberry Pi 5* permite a alternância simples entre os modelos pelo operador e possibilita demonstrações visuais comparativas da diferença de comportamento, ilustrando para audiências não especializadas o valor agregado pela classificação tipológica desenvolvida na

pesquisa. A Figura 14 exemplifica o funcionamento do Aplicativo Detector Genérico no Raspberry Pi.

Figura 14 - Aplicativo Detector Genérico.



Fonte: Acervo Pessoal Autor (2026).

Do ponto de vista de implementação, o Aplicativo 2 difere do Aplicativo 1 apenas na constante que define o caminho do modelo carregado. Enquanto o primeiro aponta para o modelo *NCNN* especialista (*/home/flavio/yolov8n_best_ncnn_model*), o segundo carrega o arquivo *PyTorch* original *yolov8n.pt*, distribuído pela *Ultralytics* como modelo de referência treinado no *COCO*. Toda a infraestrutura restante, incluindo o módulo *detector_core.py*, a captura de *webcam*, a renderização das *bounding boxes* e o tratamento de paralelismo, é compartilhada entre os dois aplicativos.

3.8.4 Aplicativo 3: Coletor para Retreinamento Iterativo

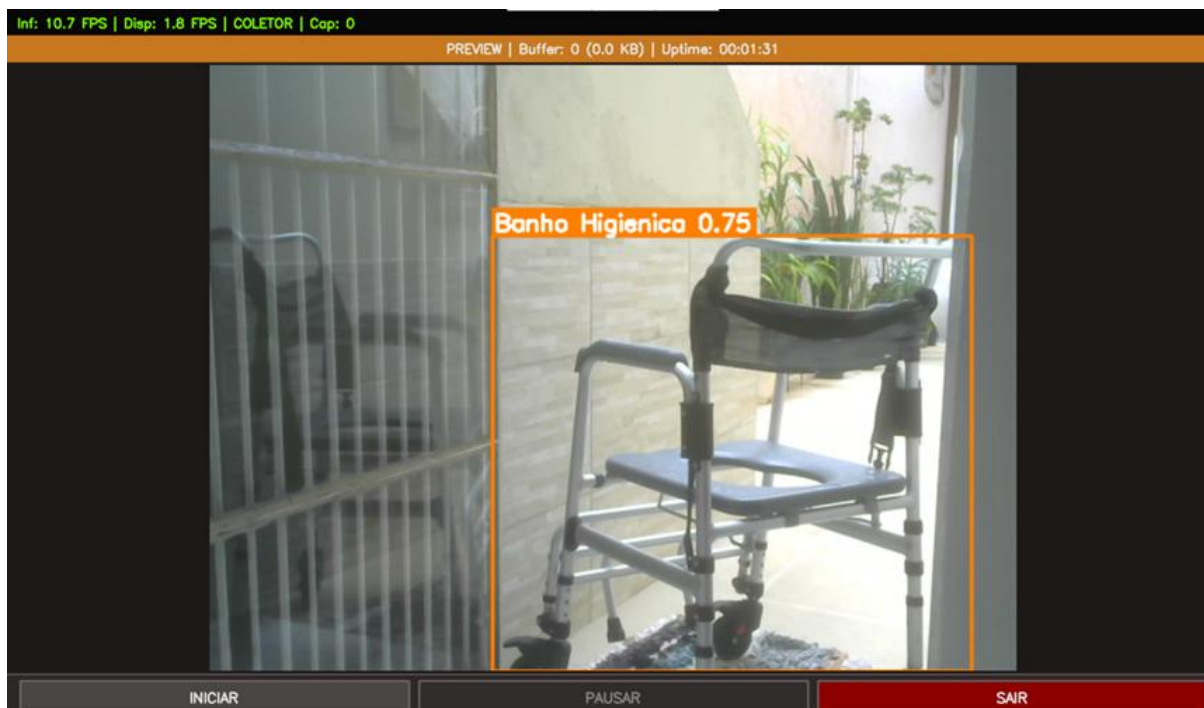
O Aplicativo 3 implementa um ciclo de aprendizado ativo (*active learning*) e pseudo-anotação (*pseudo-labeling*), estratégia bem estabelecida na literatura de aprendizado de máquina para a expansão progressiva de conjuntos de dados em domínios específicos. Diferentemente do Aplicativo 1, cuja função é detectar e registrar metadados, o Aplicativo 3 captura periodicamente o quadro bruto da *webcam*

(a cada 1 segundo, em formato *JPEG* e resolução de 640×480 pixels) e *armazena*, em paralelo, um arquivo de texto contendo as predições do modelo especialista no formato *YOLO TXT*, idêntico ao padrão de anotação descrito na Seção 3.5.1.

O fluxo previsto consiste em quatro etapas iterativas. Em primeiro lugar, o aplicativo opera em ambiente real, capturando quadros e gerando pseudo-anotações automáticas com base nas predições do modelo. Em segundo lugar, as pseudo-anotações são revisadas manualmente pelo pesquisador no *Labellmg*, com correção dos erros de classe ou de *bounding box* detectados visualmente. Em terceiro lugar, os pares quadro-anotação corrigidos são incorporados ao *corpus* original (Seção 3.3), expandindo o número de exemplos disponíveis para treinamento, em particular nas classes sub-representadas. Em quarto lugar, o modelo é retreinado sobre o *corpus* ampliado, gerando uma nova versão potencialmente mais precisa, especialmente nas tipologias com volume amostral originalmente reduzido. Esse ciclo pode ser repetido sucessivamente, configurando uma estratégia de aprimoramento contínuo do sistema.

Uma decisão técnica relevante do Aplicativo 3 (Figura 15) refere-se ao limiar de confiança aplicado às predições durante a captura. Enquanto o Aplicativo 1, voltado à operação de produção, utiliza um limiar de 0,5 (apenas detecções com confiança igual ou superior a 50% são consideradas válidas), o Aplicativo 3 adota um limiar mais permissivo de 0,25, com o objetivo de maximizar o volume de pseudo-anotações disponíveis para revisão. A justificativa para essa escolha é operacional: na fase de revisão no *Labellmg*, é mais eficiente para o pesquisador corrigir uma anotação automática imprecisa, movendo a *bounding box* ou alterando a classe atribuída, do que anotar do zero um caso que o modelo deixou de detectar, o que exigiria desenhar a *bounding box* manualmente desde o início. A literatura de aprendizado ativo apoia essa intuição: a correção é, em termos de esforço humano, menos custosa do que a anotação completa.

Figura 15 - Aplicativo Coletor



Fonte: Acervo Pessoal Autor (2026).

A coleta iterativa do Aplicativo 3 tem como foco preferencial as classes de menor volume amostral no *corpus* inicial, conforme analisado na Seção 3.5.2: Hospitalar/Transporte (com 14 exemplares no *corpus* original), Bariátrica (32), Manual Ativa (48) e Postural (55). A expansão progressiva dessas classes constitui a estratégia central para a superação do desbalanceamento estrutural do *corpus* e para a aproximação do $mAP@0.5$ agregado ao limiar de 0,85 estabelecido na hipótese operacional. As iterações futuras desse ciclo, que ultrapassam o escopo temporal desta dissertação, são tratadas no Capítulo 5 como linha de continuidade da pesquisa.

O Quadro 15 sintetiza as características distintivas dos três aplicativos do sistema operacional embarcado.

Quadro 15 - Síntese dos três aplicativos do sistema operacional embarcado

Aplicativo	Modelo Carregado	Limiar de confiança	Função primária	Persistência de dados
Aplicativo 1 (Detector Especialista)	YOLOv8n <i>NCNN</i> FP16 (11 classes tipológicas)	0,5	Operação de produção com registro de eventos	Somente metadados em CSV; imagens descartadas
Aplicativo 2 (Detector Genérico)	YOLOv8n <i>PyTorch</i> (COCO, 80 classes)	0,5	Demonstração comparativa em apresentações	Nenhuma persistência
Aplicativo 3 (Coletor Iterativo)	YOLOv8n <i>NCNN</i> FP16 (11 classes tipológicas)	0,25	Coleta de pseudo- anotações para retreinamento	Quadros em JPEG e pseudo-anotações em TXT, revisados manualmente

Fonte: Elaborado pelo Autor (2026).

O Quadro 15 evidencia que os três aplicativos, embora compartilhem a mesma infraestrutura de inferência fornecida pelo módulo *detector_core.py*, diferem deliberadamente em três dimensões que definem o uso operacional de cada um: o modelo carregado, o limiar de confiança aplicado às detecções e a política de persistência dos dados produzidos. Essa diferenciação operacional concretiza, no plano da implementação, a triangulação metodológica anunciada no início desta seção: o Aplicativo 1 sustenta a dimensão quantitativa por meio de séries temporais auditáveis sem comprometer a privacidade do usuário; o Aplicativo 2 viabiliza a dimensão comparativa qualitativa pelo contraste direto entre o modelo especialista e o modelo COCO genérico; e o Aplicativo 3 inaugura a dimensão iterativa, ao transformar a operação cotidiana do sistema em fonte deliberada de novos dados para o ciclo subsequente de treinamento. A arquitetura descrita nesta seção valida, portanto, a viabilidade do sistema do ponto de vista da operação computacional cotidiana. Resta ainda, contudo, um requisito complementar para que a solução seja efetivamente implantável em cenários reais do ambiente construído: a sua viabilidade sob restrições de energia e mobilidade, cuja avaliação experimental é objeto da seção subsequente.

3.9 VIABILIDADE OPERACIONAL EM CAMPO

A operação do sistema assistivo em ambientes reais do ambiente construído impõe um requisito que não se reduz ao desempenho computacional avaliado nas seções anteriores: a capacidade de funcionamento sustentado em locais onde a

disponibilidade de alimentação elétrica pode ser intermitente ou ausente. Embora a avaliação inicial da plataforma embarcada (Seção 3.7) e a operacionalização dos três aplicativos (Seção 3.8) tenham sido conduzidas com alimentação direta pela fonte oficial *USB-C* de 27 W do *Raspberry Pi 5*, a ampliação do uso do sistema para cenários institucionais como escolas, centros de reabilitação e terminais de transporte público, em particular para a operação do Aplicativo 3 de coleta iterativa, tornou necessária a investigação de soluções de alimentação portáteis. Esta seção apresenta a análise experimental conduzida para essa investigação, incluindo o teste inicial fracassado com bateria externa *USB-C*, a avaliação comparativa de quatro alternativas, a configuração final selecionada com base no módulo UPS (do termo em inglês *Uninterruptible Power Supply*, fonte de alimentação ininterrupta) Geekworm X1202 e o protocolo de conexão decorrente da decisão.

A primeira hipótese de alimentação portátil avaliada foi um *powerbank* *USB-C* com tecnologia Power Delivery (PD), aparentemente suficiente por admitir o fornecimento dos 27 W nominais do dispositivo. Os testes, contudo, revelaram comportamento oposto ao esperado: mesmo com cabos certificados para corrente elevada, o sistema acusou subtensão sustentada na fonte, levando o *firmware* a reduzir a frequência da CPU à metade da configurada (de 2.800 para 1.500 MHz) e a degradar o FPS de 14,4 para cerca de 4. A temperatura permaneceu baixa durante o teste (48,8 °C), confirmando tratar-se de limitação exclusivamente elétrica, decorrente de corrente sustentada insuficiente, e não térmica. Esse resultado descartou o *powerbank* *USB-C* convencional para a aplicação proposta; as medições detalhadas constam do Apêndice A.6.

Diante do impasse, foram avaliadas quatro alternativas adicionais para a alimentação portátil do sistema: bateria recarregável de parafusadeira de uso geral acoplada a um conversor *buck* para regulação de tensão; módulo *UPS HAT Waveshare*, solução comercial específica para *Raspberry Pi*; módulo *UPS HAT Geekworm X1202*, também específico para *Raspberry Pi*, mas com topologia distinta de alimentação; e o próprio *powerbank USB-C* com adaptador de corrente ampliada. A análise comparativa considerou três critérios principais. O primeiro foi a capacidade energética disponível, mensurada em *watt-hora* (Wh) e correlacionada com a autonomia esperada sob carga de inferência. O segundo foi a topologia de alimentação adotada pelo módulo, em particular se ele libera ou ocupa a entrada *USB-C* do *Pi*, considerando que essa porta é a única fonte oficialmente suportada pelo

firmware do dispositivo. O terceiro foi a presença de função UPS, isto é, a capacidade de transição automática entre alimentação externa e bateria interna sem interrupção do funcionamento do sistema, requisito essencial para uso em campo onde quedas eventuais de energia podem ocorrer. A configuração que melhor atendeu aos três critérios simultaneamente foi o módulo *Geekworm X1202*.

O *Geekworm X1202* (Figura 16) é um módulo de expansão (*HAT*, sigla para *Hardware Attached on Top*) projetado especificamente para o *Raspberry Pi 5*, com formato compatível e instalação por encaixe direto sobre a placa principal. A configuração adotada nesta pesquisa utilizou quatro células 18650 do tipo *Samsung 30Q* (3.000 *mAh* cada, com capacidade nominal combinada de aproximadamente 43 Wh, $4 \times 3,0 \text{ Ah} \times 3,6 \text{ V}$, dos quais cerca de 30 a 38 Wh efetivamente utilizáveis dependendo do estado de carga, da temperatura de operação e do corte de subtensão) em montagem *flat-top*. A topologia de alimentação do *X1202* difere fundamentalmente da adotada pelos powerbanks convencionais: em vez de alimentar o *Pi* pela entrada USB-C, o módulo entrega energia diretamente aos pinos GPIO de 5 V por meio de *pogo pins*, contatos de mola que se acoplam aos pinos da placa quando o *HAT* é instalado, liberando a entrada *USB-C* do *Pi* para a conexão da fonte externa de carregamento. Essa topologia elimina a necessidade de partilhar o barramento *USB-C* entre alimentação externa e bateria interna, e habilita a função *UPS* que constituiu critério decisivo da escolha: o sistema continua operando ininterruptamente em caso de queda de energia externa, com transição automática para a bateria sem reinicialização. A solução foi adquirida via plataforma *AliExpress*, ao custo aproximado de 35 dólares para o módulo (excluídas as células 18650, adquiridas separadamente), valor coerente com o princípio de baixo custo que orienta o trabalho.

Figura 16 - Módulo *UPS Geekworm X1202*

Fonte: Acervo Pessoal Autor (2026).

Os testes operacionais conduzidos em modo bateria confirmaram a viabilidade da configuração para o uso em campo. Sob carga de inferência contínua com o Aplicativo 1 ativo, descrito na Seção 3.8.2, o sistema manteve a frequência sustentada de 2.800 MHz e o *FPS* entre 13 e 14,4, equivalentes aos valores reportados na Seção 3.7.3 com alimentação pela fonte oficial. Não foi registrada qualquer ativação da *flag 0x50000*, demonstrando que o módulo entrega a potência adequada à demanda real do dispositivo. A autonomia mensurada situou-se entre 4 e 5 horas de operação ininterrupta com as quatro células totalmente carregadas, intervalo compatível com sessões típicas de coleta de campo em ambientes institucionais. O Quadro 16 sintetiza a análise comparativa entre as configurações de alimentação efetivamente testadas.

Quadro 16 - Comparação experimental entre as configurações de alimentação avaliadas para o *Raspberry Pi 5*

Configuração	Frequência sustentada pela CPU	Status Elétrico	FPS sustentado	Adequação ao uso em campo
Fonte oficial 27 W (5 V / 5 A)	2.800 MHz	Operação plena (flag 0x0)	14,4	Referência para uso fixo (não portátil)
Powerbank USB-C, cabo 1	1.500 MHz	Subtensão severa (flag 0x50000)	4,1	Inviável
Powerbank USB-C, cabo 2	1.500 MHz	Subtensão severa (flag 0x50000)	4,2	Inviável
UPS Geekworm X1202 (4 × 18650 Samsung 30Q)	2.800 MHz	Operação plena (flag 0x0)	13 a 14,4	Adequada para até 5 horas de operação em campo

Fonte: Elaborado pelo Autor (2026).

A adoção do módulo *X1202* trouxe consigo um detalhe operacional que merece registro explícito como protocolo de conexão. Em razão de o módulo adicionar uma segunda fonte ativa de tensão (a bateria interna) ao mesmo barramento de 5 V do *Raspberry Pi*, a conexão simultânea da fonte oficial *USB-C* na entrada do *Pi* e do módulo *X1202* nos pinos *GPIO* criaria um conflito potencial entre duas fontes de tensão concorrentes, situação capaz de provocar instabilidade elétrica e, em casos extremos, danos aos circuitos do dispositivo. O protocolo correto, conforme as instruções do fabricante e validado nos testes desta pesquisa, exige que o carregador *USB-C* oficial seja conectado exclusivamente à entrada *USB-C* do próprio módulo *X1202*, e nunca diretamente ao *Pi*. Dessa forma, o módulo passa a atuar como ponto único de gerenciamento de energia: quando há alimentação externa presente, ele conduz a corrente para o *Pi* e simultaneamente carrega as células 18650; quando a alimentação externa é interrompida, ele transiciona automaticamente para o modo de descarga da bateria, sem interromper o funcionamento do sistema. Essa configuração foi padronizada como procedimento operacional padrão para todos os cenários de uso em campo previstos na pesquisa.

A configuração de alimentação aqui descrita complementa a arquitetura consolidada nas Seções 3.7 e 3.8, encerrando o conjunto de decisões metodológicas necessárias para a operação efetiva do sistema assistivo em ambientes reais. O sistema, no estado em que foi entregue por esta pesquisa, é capaz de operar com autonomia de até 5 horas em modo bateria, com a mesma taxa de inferência e com as mesmas garantias de privacidade e estabilidade demonstradas em operação com fonte fixa. As limitações que ainda subsistem em relação ao escopo total da operação

contínua de longo prazo (incluindo a necessidade de troca periódica de células, a impossibilidade de operação prolongada sem acesso à rede elétrica para recarga das baterias, e a sensibilidade do conjunto às variações de temperatura ambiente típicas de espaços externos) são tratadas, junto com as demais premissas e restrições do trabalho, na Seção 3.11 como parte do conjunto consolidado de limitações da pesquisa.

3.10 ANÁLISE DOS RESULTADOS

A análise dos resultados experimentais foi estruturada em três dimensões complementares, articuladas de modo a fornecer uma avaliação simultaneamente quantitativa, comparativa e interpretativa do sistema desenvolvido. Essas dimensões são aqui apresentadas em seu plano metodológico, antes de serem efetivamente operacionalizadas no Capítulo 4 a partir dos dados experimentais coletados ao longo do trabalho.

A primeira dimensão compreende a avaliação quantitativa do desempenho dos quatro modelos treinados (*YOLOv5n*, *YOLOv5s*, *YOLOv8n* e *YOLOv8s*), com base nas métricas de detecção definidas na Seção 3.6.2: *Precision*, *Recall*, *F1-Score*, *mAP@0.5* e *mAP@0.5:0.95*. A análise contempla tanto os valores agregados sobre o conjunto de validação quanto o desempenho individual por classe tipológica, abordagem cuja relevância foi antecipada na Seção 3.6.2 em razão do desbalanceamento expressivo documentado na Seção 3.5.2 entre as onze categorias do *corpus*. A análise por classe é particularmente importante para distinguir tipologias com desempenho satisfatório, candidatas à operação imediata do sistema, das tipologias sub-representadas que demandam expansão do *corpus* por meio do ciclo iterativo do Aplicativo 3 descrito na Seção 3.8.4. As matrizes de confusão geradas pela rotina de validação de cada modelo constituem o instrumento auxiliar central dessa análise por classe, ao revelar os padrões de confusão entre tipologias morfologicamente similares (por exemplo, entre Manual Dobrável e Hospitalar/Transporte, ou entre Motorizada e Postural) e ao identificar fontes recorrentes de erro de classificação que orientarão futuras intervenções no protocolo de anotação ou na composição do *corpus*.

A segunda dimensão abrange a análise comparativa de eficiência computacional entre as configurações de inferência avaliadas no *Raspberry Pi 5*,

confrontando os indicadores de *FPS*, latência por inferência e tamanho do modelo com os limiares operacionais estabelecidos *a priori* na hipótese de pesquisa (Seção 3.1): $mAP@0.5 \geq 0,85$ e $FPS \geq 15$ no dispositivo embarcado. A comparação inclui as variantes geradas na Seção 3.7.2 (*TensorFlow Lite FP32*, *TensorFlow Lite INT8* e *NCNN FP16*), as otimizações incrementais documentadas na Seção 3.7.3 (substituição do runtime de inferência e *overclock* controlado da *CPU*), o desempenho em cenários reais de operação com captura por *webcam* e renderização em *display*, e a estabilidade térmica e elétrica em regime sustentado, esta última documentada também na Seção 3.9 sob diferentes configurações de alimentação. Essa dimensão é responsável por verificar se a implementação efetivamente entregue atende aos requisitos de tempo real previstos para a operação no ambiente construído, ou se o atendimento ocorre apenas em parte, caso em que a discussão das implicações funcionais para o sistema assistivo passa a ser parte integrante da análise, e não mero registro de não conformidade.

A terceira dimensão promove a triangulação metodológica entre os dados quantitativos coletados nas duas dimensões anteriores e as observações qualitativas oriundas dos testes em cenário operacional, conforme preconizado por Marconi e Lakatos (2017). Essa triangulação cumpre dupla função epistemológica. Por um lado, ela permite confrontar valores numéricos isolados (por exemplo, um $mAP@0.5$ agregado em determinado patamar ou um *FPS* sustentado em outro) com o significado prático que esses valores assumem em cenários de uso, situando o desempenho do sistema em um quadro interpretativo mais amplo do que o estritamente métrico. Por outro lado, ela permite incorporar à análise as observações qualitativas decorrentes dos próprios percalços documentados ao longo do Capítulo 3 (a inadequação do conjunto de calibração na quantização *INT8* do *YOLOv5*, a contenção entre o *OpenMP* e o compositor gráfico, a subtensão do *powerbank USB-C*, entre outros), tratando-os não como ruído experimental, mas como informação metodológica relevante para o entendimento do comportamento do sistema sob restrições reais. Essa estratégia de triangulação consolida a posição assumida desde a Seção 3.8, segundo a qual a arquitetura dos três aplicativos (produção, comparação, coleta iterativa) já constitui, no plano da implementação, um arranjo deliberadamente triangulado de avaliação do sistema.

A análise estruturada nessas três dimensões compõe o eixo argumentativo do Capítulo 4, no qual os resultados experimentais são apresentados, comparados aos

limitares estabelecidos *a priori* e interpretados à luz das condições reais de implementação. As três dimensões dialogam entre si: o desempenho quantitativo (dimensão 1) é avaliado também sob a perspectiva da viabilidade computacional (dimensão 2), e a leitura conjunta das duas é refinada pela triangulação com os percalços e ajustes documentados ao longo do percurso experimental (dimensão 3). Essa abordagem evita o risco, recorrente em pesquisas de aprendizado de máquina aplicado, de reduzir a discussão de resultados a uma listagem de métricas isoladas, sem correspondência com o contexto operacional e com os limites práticos da solução desenvolvida.

3.11 LIMITAÇÕES E PREMISSAS

A explicitação das limitações e das premissas adotadas constitui parte integrante do rigor metodológico de uma pesquisa científica, ao permitir a interpretação adequada dos resultados e ao orientar pesquisadores futuros que venham a desenvolver estudos correlatos. Esta seção sintetiza, de modo articulado, as cinco limitações principais identificadas ao longo do desenvolvimento desta dissertação. A explicitação aqui adotada não busca minimizar nem ocultar essas limitações, mas, ao contrário, integrá-las à compreensão do que o trabalho efetivamente entrega e do que permanece como agenda de pesquisa para iterações futuras.

A primeira limitação refere-se à escala do *corpus* utilizado para o treinamento dos modelos. As 2.000 imagens consolidadas conforme descrito na Seção 3.3, embora diversificadas em termos de fontes (quatro repositórios públicos complementados por busca direcionada na ferramenta *Bing Image Search*) e de condições de captura (variações de iluminação, ângulo e contexto), constituem uma base de escala moderada quando comparadas aos conjuntos de referência da literatura de visão computacional, que tipicamente excedem 10.000 imagens por classe. Essa limitação é parcialmente mitigada por três estratégias adotadas no trabalho: a priorização da variabilidade sobre o volume bruto, fundamentada empiricamente em Adorno *et al.* (2023); o uso de *transfer learning* a partir de pesos pré-treinados no conjunto COCO, que confere à rede um conhecimento prévio robusto sobre detecção de objetos genéricos; e a operacionalização do ciclo de coleta iterativa por meio do Aplicativo 3, descrito na Seção 3.8.4, que constitui o caminho

deliberadamente projetado para a expansão progressiva do *corpus* em iterações futuras. Apesar dessas estratégias, a escala atual pode impactar a capacidade de generalização do modelo para as tipologias sub-representadas no banco de dados, conforme já antecipado nas Seções 3.5.2 e 3.6.2.

A segunda limitação refere-se ao fato de que a anotação manual das 2.000 imagens, descrita em detalhe na Seção 3.5.1, foi realizada integralmente pelo pesquisador, sem participação de um segundo anotador independente. Essa restrição impossibilita o cálculo de métricas formais de concordância interanotadores, como o coeficiente *Kappa de Cohen*, que constituem padrão de robustez metodológica em estudos de visão computacional aplicada. Para mitigar essa limitação, foram adotadas duas medidas operacionais. A primeira consistiu na revisão sistemática integral das marcações ao término da fase de anotação, com inspeção visual auxiliada pela função de visualização do próprio *Labellmg*, conforme descrito no item (e) do protocolo da Seção 3.5.1. A segunda foi a implementação do ciclo de pseudo-anotação revisada do Aplicativo 3, que introduz, na prática, uma forma de validação cruzada distribuída ao longo do tempo: à medida que o sistema opera em campo e produz pseudo-anotações que são revisadas pelo pesquisador, gera-se um histórico crescente de marcações revistas que pode ser comparado retrospectivamente com as anotações originais do *corpus*. Recomenda-se, contudo, em pesquisas subsequentes, a incorporação de um segundo anotador independente para a validação cruzada de ao menos uma amostra representativa do banco de dados, possibilitando o cálculo formal das métricas de concordância usuais.

A terceira limitação decorre da incompatibilidade entre a plataforma *AMD ROCm* e o sistema operacional *Windows* utilizado na estação de trabalho do pesquisador, conforme detalhado na Seção 3.6.1. Essa incompatibilidade impôs a condução do treinamento exclusivamente em *CPU*, em vez da aceleração por *GPU AMD Radeon RX 7600* originalmente prevista, com impacto significativo sobre o tempo de treinamento (aproximadamente 41 horas distribuídas em cinco dias para os quatro modelos, contra estimativas substancialmente menores em *GPU* dedicada) e, conseqüentemente, sobre a viabilidade de varreduras exaustivas do espaço de hiperparâmetros e de múltiplas execuções para análise de variância dos resultados. Cabe ressaltar, contudo, que essa limitação afeta apenas o tempo de treinamento e a possibilidade de exploração intensiva do espaço de hiperparâmetros, não comprometendo a qualidade dos pesos finais dos modelos resultantes, que são

funcionalmente independentes do dispositivo de processamento utilizado durante o treinamento.

A quarta limitação está intrinsecamente associada à própria estratégia ética que orienta esta pesquisa. O princípio de volatilidade total dos dados em tempo de inferência, estabelecido no Capítulo 1 e operacionalizado nas Seções 3.5.1 e 3.8.2, determina que as imagens capturadas em campo sejam descartadas imediatamente após a inferência, preservando-se apenas os metadados estruturados de detecção. Essa decisão, embora coerente com os princípios de privacidade e de proteção de dados pessoais, implica uma limitação metodológica relevante: a impossibilidade de auditoria retrospectiva de eventos individuais a partir das imagens originais. Em termos práticos, isso significa que, se uma detecção registrada nos logs do sistema for posteriormente questionada quanto à sua correção (por exemplo, na investigação de uma tipologia eventualmente atribuída de modo equivocado), não há possibilidade de recuperação visual do quadro que gerou aquela detecção. A mitigação dessa limitação ocorre indiretamente pela arquitetura do Aplicativo 3, no qual os quadros são deliberadamente persistidos em uma sessão controlada de coleta para retreinamento, com consentimento informado e protocolos específicos de proteção do material gerado, desacoplada da operação rotineira do Aplicativo 1. Tal segregação preserva o princípio ético central da pesquisa ao mesmo tempo em que mantém disponível um caminho técnico para a auditoria visual quando necessário.

A quinta e última limitação refere-se ao escopo dos cenários nos quais o sistema foi efetivamente testado. Os experimentos descritos nas Seções 3.7 e 3.9 foram conduzidos em ambientes controlados, com *webcam USB* de resolução 720p, sob condições de iluminação interior e com presença previsível dos objetos de interesse na cena. A validade externa do sistema para ambientes reais de operação (aeroportos, hospitais, terminais de transporte público, edifícios institucionais e centros de reabilitação) ainda demanda etapas adicionais de validação em campo, que extrapolam o escopo temporal desta dissertação. Variáveis como condições variáveis de iluminação (luz solar direta, contraluz, ambientes mal iluminados ou parcialmente sombreados), distorções decorrentes de ângulos não convencionais de captura, oclusões por outras pessoas em ambientes densamente ocupados, e reflexos em superfícies metálicas ou envidraçadas características de espaços públicos podem introduzir desafios adicionais não capturados pelo *corpus* atual. A operacionalização do ciclo iterativo do Aplicativo 3 em ambientes reais constitui a estratégia direta

projetada para a progressiva incorporação dessas variabilidades ao *corpus* em iterações de aprimoramento, conforme tratado no Capítulo 5 como linha de continuidade da pesquisa.

O conjunto de limitações aqui identificadas não invalida os resultados experimentais que serão apresentados no Capítulo 4, mas qualifica sua interpretação e orienta o desenho de pesquisas subsequentes. As cinco limitações enumeradas, longe de constituírem fragilidades estruturais do trabalho, representam fronteiras conscientemente assumidas em decorrência de decisões metodológicas explícitas (como o princípio de volatilidade dos dados, que prioriza a privacidade do usuário em detrimento da auditoria retrospectiva) ou de restrições operacionais delimitadas pelo escopo temporal e pelos recursos disponíveis para a pesquisa. A arquitetura do sistema desenvolvido prevê, em sua própria concepção, mecanismos de superação progressiva dessas limitações, em particular por meio do ciclo iterativo de coleta e retreinamento operacionalizado pelo Aplicativo 3 da Seção 3.8.4.

O conjunto das decisões metodológicas aqui sistematizadas habilita a apresentação dos resultados experimentais, conduzida no Capítulo 4.

3.12 SÍNTESE E CONSIDERAÇÕES PARCIAIS

Este capítulo descreveu, de forma reproduzível, o percurso metodológico que conduz da concepção do sistema à sua validação experimental. Definiu-se a pesquisa como aplicada e exploratória, ancorada no método hipotético-dedutivo de Marconi e Lakatos (2017), com a hipótese operacional de $mAP@0.5$ igual ou superior a 0,85 e taxa de inferência igual ou superior a 15 quadros por segundo no *Raspberry Pi*, sob a premissa transversal de baixo custo e preservação da privacidade. Apresentou-se a constituição do *corpus* (das 19.452 imagens brutas, oriundas de quatro repositórios públicos e de duas rodadas de busca direcionada, às 2.000 imagens efetivamente anotadas) e a elaboração do quadro de onze tipologias que orientou a anotação no *LabelImg* e a divisão estratificada em treinamento, validação e teste (70/20/10), já registrando o desbalanceamento estrutural entre as classes. Descreveram-se o treinamento comparativo dos quatro modelos (*YOLOv5n*, *YOLOv5s*, *YOLOv8n* e *YOLOv8s*) em *CPU*, a conversão para a arquitetura *ARM* nos formatos *TFLite* e *NCNN*, os protocolos de *benchmark* no *Raspberry Pi 5*, o sistema operacional embarcado composto por três aplicativos articulados como triangulação metodológica

e a viabilidade operacional em campo viabilizada pelo módulo *UPS Geekworm X1202*. Por fim, explicitaram-se a estratégia de análise dos resultados e as cinco limitações e premissas assumidas. O conjunto dessas decisões fornece a base sobre a qual o Capítulo 4 apresenta e discute os resultados experimentais.

4 RESULTADOS

Este capítulo apresenta os resultados experimentais decorrentes das decisões metodológicas descritas no Capítulo 3, conduzindo a análise segundo as três dimensões complementares anunciadas na Seção 3.10: a avaliação quantitativa do desempenho dos modelos treinados, a análise comparativa de eficiência computacional na plataforma embarcada, e a triangulação metodológica entre os dados quantitativos e as observações qualitativas registradas ao longo do percurso experimental, conforme preconizado por Marconi e Lakatos (2017).

O capítulo organiza-se em sete seções: a Seção 4.1 caracteriza o *corpus* efetivamente utilizado; as Seções 4.2 e 4.3 apresentam os resultados de treinamento e o desempenho por classe; as Seções 4.4 e 4.5 examinam o sistema embarcado em laboratório e em campo; e as Seções 4.6 e 4.7 conduzem a discussão crítica e a transição para o Capítulo 5.

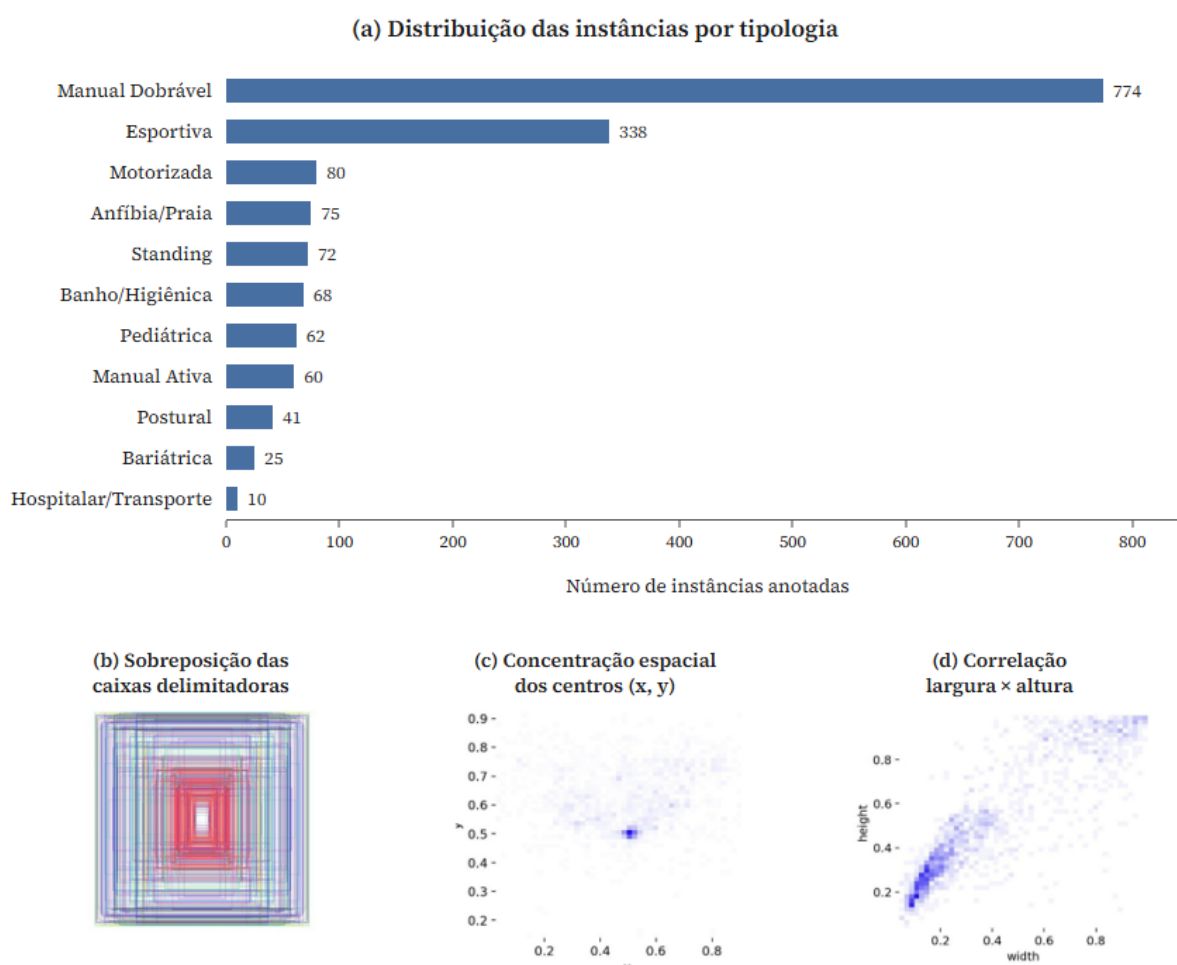
4.1 CARACTERIZAÇÃO DO CORPUS DE TREINAMENTO

O *corpus* utilizado no treinamento e na validação dos modelos resulta da consolidação de quatro repositórios públicos complementados por busca direcionada via Bing Image Search, conforme descrito na Seção 3.3, totalizando 2.000 imagens anotadas e particionadas estratificadamente em treinamento (1.394 imagens, 69,7%), validação (394 imagens, 19,7%) e teste (212 imagens, 10,6%), em conformidade com o protocolo apresentado na Seção 3.5.2. A composição final por tipologia mantém a assimetria observada nas fontes públicas consultadas, refletindo a frequência relativa real das classes em circulação e configurando um desbalanceamento estrutural cujo impacto sobre o desempenho por classe é discutido nas seções subsequentes deste capítulo.

A Figura 17 reúne quatro representações complementares do conjunto de treinamento, identificadas pelos rótulos (a) a (d): em (a), a distribuição quantitativa das

instâncias anotadas por tipologia; em (b), a sobreposição de todas as caixas delimitadoras anotadas; em (c), a concentração espacial dos centros das caixas em coordenadas normalizadas; e, em (d), a correlação geométrica entre a largura e a altura dos retângulos de anotação. Cada um desses componentes é discutido a seguir.

Figura 17 - Distribuição das instâncias por tipologia no conjunto de treinamento.



Fonte: Elaborado pelo autor (2026).

O painel (a) da Figura 17, que apresenta a distribuição das instâncias por tipologia, evidencia três achados relevantes. Em primeiro lugar, a tipologia Manual Dobrável reúne 774 instâncias no conjunto de treinamento, o que corresponde a aproximadamente 48% do total de 1.605 objetos anotados nessa partição. A predominância dessa classe é coerente com o cenário de mercado descrito na Seção 3.4, em que a cadeira manual dobrável constitui o modelo de maior penetração comercial e, portanto, o mais frequentemente representado nos repositórios públicos consultados. Em segundo lugar, a tipologia Esportiva ocupa a segunda posição em

volume, com 338 instâncias (21% do total), refletindo a abundância de imagens de práticas paradesportivas, especialmente basquete e tênis em cadeira de rodas, disponíveis em bancos de imagens orientados a esporte e mídia. As nove demais tipologias somam, em conjunto, cerca de 31% das anotações.

Em terceiro lugar, e de maior relevância para a análise dos resultados subsequentes, o desbalanceamento entre as classes minoritárias é severo. A tipologia Hospitalar/Transporte dispõe de apenas 10 instâncias no conjunto de treinamento, em proporção aproximada de 1:77 em relação à classe Manual Dobrável. Outras tipologias apresentam volumes também reduzidos: Bariátrica com 25 instâncias, Postural com 41, Manual Ativa com 60 e Pediátrica com 62, todas abaixo de 65 instâncias cada. Essa distribuição assimétrica caracteriza um cenário típico de conjuntos de dados com cauda longa (*long-tail distribution*), em que poucas classes concentram a maior parte das amostras e diversas classes raras dispõem de volume amostral insuficiente para que modelos baseados em aprendizado supervisionado generalizem com robustez. A análise por classe apresentada na Seção 4.3 confronta essa hipótese com os resultados experimentais obtidos.

O painel (b) sobrepõe, em um mesmo referencial normalizado, todas as caixas delimitadoras anotadas no conjunto de treinamento. O padrão concêntrico e aproximadamente quadrado resultante indica que as cadeiras de rodas, uma vez enquadradas, tendem a ocupar a região central da imagem e a apresentar proporções semelhantes entre si, com predomínio de caixas de dimensões médias a grandes em relação ao quadro. Essa regularidade morfológica das anotações é favorável ao treinamento, por reduzir a variabilidade de escala e de posição que o modelo precisa acomodar.

Os painéis (c) e (d) revelam, por fim, dois padrões geométricos das anotações. A projeção espacial dos centros das caixas em coordenadas normalizadas, no painel (c), evidencia alta concentração em torno do centro horizontal ($x \approx 0,5$) e alturas variadas, resultado da prática de enquadramento centralizado em catálogos comerciais e da disposição típica de cadeiras de rodas em cenários institucionais com a câmera posicionada à altura média do tronco. A correlação entre largura e altura das caixas, no painel (d), é fortemente linear e tende à proporção 1:1, confirmando que cadeiras de rodas, quando vistas em ângulo frontal ou semilateral, ocupam regiões aproximadamente quadradas no quadro de imagem. Esse padrão geométrico é favorável às arquiteturas YOLO, que utilizam caixas-âncora pré-definidas: quanto

mais regular a forma dos objetos, melhor o ajuste das âncoras às proporções reais e mais eficiente a convergência do treinamento.

Cabe ressaltar que o conjunto documentado nesta seção corresponde à versão inicial do *corpus*, gerada conforme o protocolo de coleta da Seção 3.3 e o protocolo de anotação da Seção 3.5.1. O ciclo de coleta iterativa operacionalizado pelo Aplicativo 3 (Seção 3.8.4) prevê a expansão progressiva desse *corpus*, com foco preferencial nas classes sub-representadas identificadas nesta seção e detalhadas na análise por classe da Seção 4.3, configurando uma estratégia de aprimoramento contínuo cujos desdobramentos são discutidos no Capítulo 5.

4.2 RESULTADOS DO TREINAMENTO COMPARATIVO DOS QUATRO MODELOS

Esta seção apresenta os resultados quantitativos do treinamento dos quatro modelos selecionados na Seção 3.6 (*YOLOv5n*, *YOLOv5s*, *YOLOv8n* e *YOLOv8s*), todos treinados em *CPU* sob as mesmas condições experimentais, com 150 épocas planejadas, paciência de 30 épocas para o mecanismo de *early stopping* e os hiperparâmetros consolidados no Quadro 7. A análise comparativa que segue compreende três dimensões: o desempenho agregado mensurado pelas métricas *mAP@0.5* e *mAP@0.5:0.95*, a curva de convergência de cada modelo ao longo das épocas e o tempo computacional consumido em cada treinamento.

O Quadro 17 sintetiza os resultados quantitativos finais dos quatro modelos, apresentando o desempenho mensurado no melhor *checkpoint* (*best.pt*) preservado automaticamente pelo processo de *treinamento* em cada execução. O F1-Score, média harmônica entre Precisão e Revocação derivada diretamente das duas últimas colunas, é de 0,584 (*YOLOv5n*), 0,622 (*YOLOv5s*), 0,676 (*YOLOv8n*) e 0,653 (*YOLOv8s*), confirmando a melhor relação entre precisão e revocação do *YOLOv8n* entre os quatro modelos.

Quadro 17 - Síntese comparativa dos resultados de treinamento dos quatro modelos avaliados

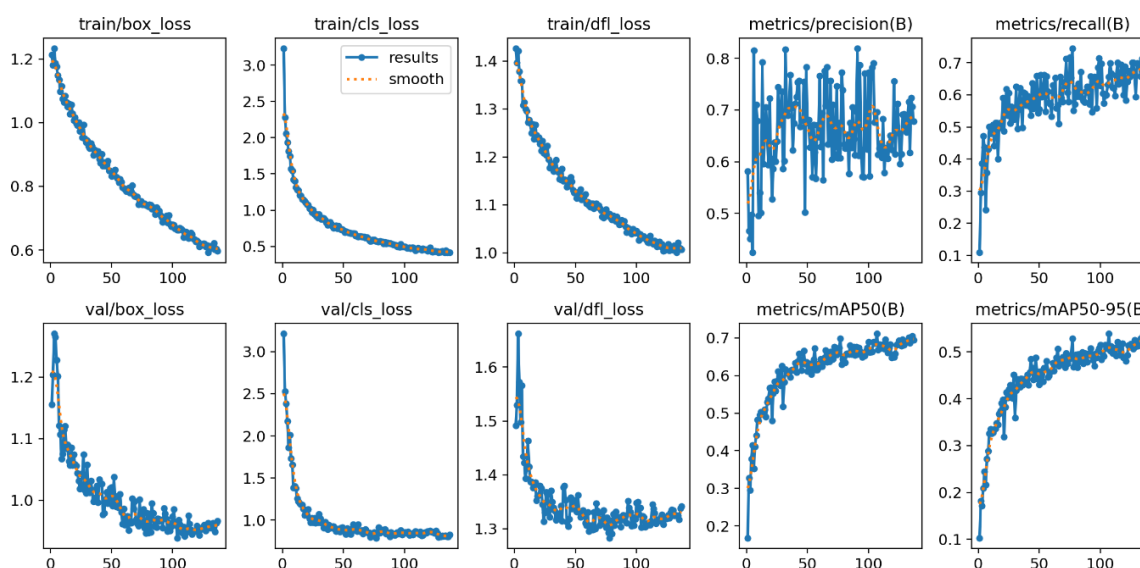
Modelo	Parâmetro	Épocas até best	Tempo (CPU)	$mAP@0.5$	$mAP@0.5:0.95$	Precision	Recall
YOLOv5n	1,78 M	77	≈ 11 h	0,609	0,427	0,606	0,564
YOLOv5s	7,04 M	109	≈ 11 h	0,641	0,481	0,604	0,642
YOLOv8n	3,01 M	107	≈ 7 h	0,711	0,539	0,697	0,657
YOLOv8s	11,14 M	99	≈ 12 h	0,684	0,519	0,678	0,629

Fonte: Elaborado pelo autor (2026).

Os resultados sintetizados no Quadro 17 evidenciam dois achados centrais que orientam a discussão subsequente. O primeiro é a superioridade agregada do YOLOv8n sobre os demais três modelos avaliados: com $mAP@0.5$ de 0,711, o YOLOv8n supera o YOLOv5s (segundo melhor desempenho agregado) em 0,070 pontos absolutos, ou cerca de 11% em ganho relativo, e o YOLOv5n em 0,102 pontos absolutos, ou aproximadamente 17%. O segundo achado, mais relevante do ponto de vista metodológico, é o desempenho inferior do modelo YOLOv8s frente ao YOLOv8n, apesar de o primeiro possuir 3,7 vezes mais parâmetros (11,14 milhões contra 3,01 milhões da variante nano). Esse comportamento contraintuitivo, aprofundado adiante nesta seção, constitui o ponto-chave da discussão metodológica. Esclarece-se, para evitar leituras equivocadas adiante, que esse $mAP@0.5$ de 0,711 corresponde à avaliação do melhor *checkpoint* em *PyTorch*; os valores entre 0,670 e 0,708 reportados na Seção 4.4 referem-se às variantes do modelo exportadas para a plataforma embarcada e constituem, portanto, medições distintas, e não divergências entre fontes.

A análise visual da convergência dos modelos durante o treinamento permite caracterizar o padrão de aprendizado de cada um. A Figura 18 apresenta as curvas de evolução das funções de perda nos conjuntos de treinamento e validação e das métricas agregadas (*precision*, *recall*, $mAP@0.5$ e $mAP@0.5:0.95$) ao longo das épocas para o modelo YOLOv8n, selecionado como modelo de produção do sistema assistivo. As curvas equivalentes dos três demais modelos avaliados apresentam padrão semelhante de convergência e seguem disponíveis como material complementar.

Figura 18 - Evolução das funções de perda no conjunto de treinamento.



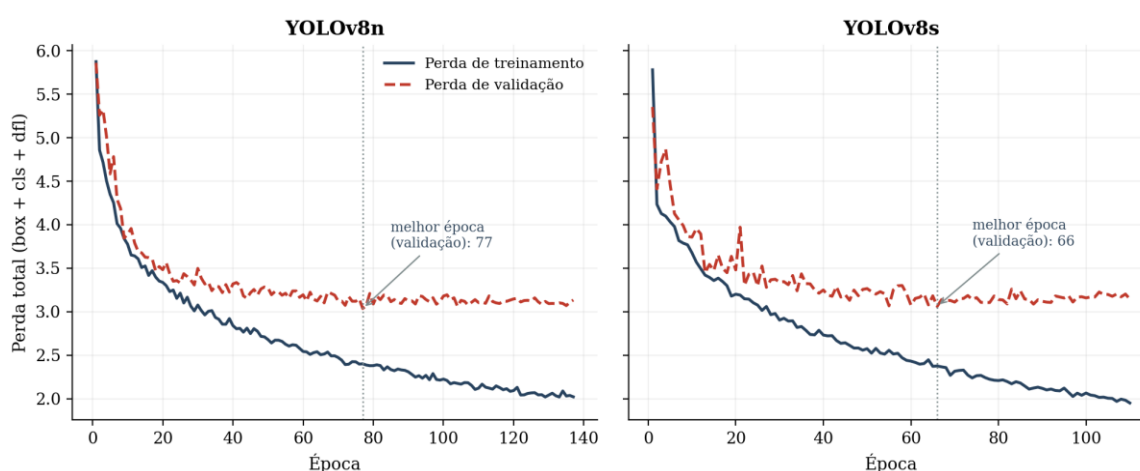
Fonte: Elaborado pelo autor (2026).

A Figura 18 demonstra três comportamentos esperados em treinamentos bem conduzidos. Em primeiro lugar, as três funções de perda no conjunto de treinamento decaem de forma monotônica ao longo das épocas, com inflexão suave após cerca de 30 épocas e estabilização em valores próximos ao mínimo a partir da época 80. Em segundo lugar, as funções de perda no conjunto de validação acompanham o mesmo padrão até aproximadamente a época 90, ponto em que se observa leve estabilização sem aumento sustentado, o que indica ausência de sobreajuste catastrófico ao longo do treinamento. Em terceiro lugar, as quatro métricas de detecção crescem de forma rápida nas primeiras 30 épocas e seguem em ascensão moderada, com o pico de $mAP@0.5$ ocorrendo na época 107, conforme registrado no Quadro 17. O mecanismo de *early stopping* interrompeu o treinamento na época 137, preservando os pesos da época 107 como melhor *checkpoint* do modelo, em conformidade com a paciência de 30 épocas configurada no Quadro 7.

O paradoxo do desempenho dos modelos *YOLOv8* merece análise específica. Em condições típicas de aprendizado de máquina supervisionado, espera-se que modelos com maior número de parâmetros, e, portanto, maior capacidade representacional, apresentem desempenho superior em conjuntos de treinamento adequadamente dimensionados. No experimento aqui reportado, contudo, o *YOLOv8s*, com 11,14 milhões de parâmetros, foi superado pelo *YOLOv8n*, que dispõe de apenas 3,01 milhões. A explicação mais plausível para esse comportamento é o sobreajuste do modelo maior frente ao *corpus* relativamente reduzido de 1.394

imagens de treinamento. Modelos com elevada capacidade representacional, quando expostos a conjuntos de dados de escala moderada, tendem a memorizar peculiaridades das amostras de treinamento em detrimento do aprendizado de padrões generalizáveis, fenômeno consistentemente reportado na literatura de aprendizado profundo aplicado a domínios específicos (Adorno *et al.*, 2023). A regularização implícita oferecida pelo *YOLOv8n*, decorrente de sua arquitetura mais enxuta, mostrou-se mais adequada ao tamanho efetivo do *corpus* desta pesquisa, e esse achado constitui evidência empírica direta da recomendação de Adorno *et al.* (2023) quanto à priorização da variabilidade sobre o volume bruto de dados, premissa adotada na Seção 3.3 e agora confirmada no plano dos resultados. A Figura 19 evidencia empiricamente esse comportamento ao confrontar, lado a lado, a evolução das perdas de treinamento e de validação das duas variantes: em ambas a perda de treinamento prossegue em queda, ao passo que a de validação se estabiliza, mas no *YOLOv8s* a perda de validação atinge seu valor mínimo mais cedo (por volta da época 66, contra a época 77 no *YOLOv8n*) e mantém, ao final do treinamento, uma distância maior em relação à perda de treinamento, indício de sobreajuste mais acentuado na variante de maior capacidade representacional.

Figura 19 - Curvas de perda de treinamento e validação dos modelos *YOLOv8n* e *YOLOv8s*.



Fonte: Elaborado pelo autor (2026).

Quanto ao tempo computacional, o *YOLOv8n* consumiu aproximadamente 7 horas para concluir o treinamento, valor inferior ao registrado nos demais modelos. Os modelos *YOLOv5n* e *YOLOv5s* demandaram cerca de 11 horas cada, apesar das diferenças significativas de capacidade entre as variantes, enquanto o *YOLOv8s* consumiu aproximadamente 12 horas. O tempo total acumulado dos quatro

treinamentos aproxima-se de 41 horas de processamento contínuo em *CPU*, distribuídas em sessões noturnas ao longo de cinco dias consecutivos, conforme documentado na Seção 3.6.1. Essa magnitude reforça a justificativa metodológica para a opção pelas variantes nano e small das duas arquiteturas, em detrimento das versões medium, large e xlarge: além das considerações de viabilidade no embarcado expostas na Seção 3.6, a varredura completa das variantes maiores em *CPU* teria sido inviável dentro do escopo temporal da pesquisa.

A combinação dos critérios métricos e operacionais conduz à seleção do *YOLOv8n* como modelo de produção do sistema assistivo. Esse modelo apresenta simultaneamente o melhor $mAP@0.5$ agregado (0,711), o melhor $mAP@0.5:0.95$ (0,539) e a melhor precisão (0,697) entre os quatro modelos avaliados, associando esse desempenho a tempo de treinamento substancialmente inferior e arquitetura mais compacta, atributos cuja relevância é explorada nas Seções 4.4 e 4.5 quando da análise do desempenho na plataforma embarcada. Cabe ressaltar, contudo, que o atendimento agregado ao critério de $mAP@0.5 \geq 0,85$ estabelecido na hipótese operacional (Seção 3.1) não foi obtido por nenhum dos quatro modelos avaliados. O melhor desempenho registrado, do *YOLOv8n*, situa-se a 0,139 pontos absolutos abaixo do limiar formal. Essa constatação não invalida a viabilidade do sistema, mas exige a análise por classe apresentada na Seção 4.3, que distingue tipologias com desempenho satisfatório de classes sub-representadas que demandam expansão do *corpus*, e a discussão crítica retomada na Seção 4.6, em que o atingimento parcial do critério é confrontado com as condições reais de implementação e com os caminhos de superação previstos na arquitetura do sistema.

4.3 ANÁLISE DO DESEMPENHO POR CLASSE

A análise agregada apresentada na Seção 4.2 estabeleceu o *YOLOv8n* como modelo de melhor desempenho global e identificou que nenhum dos quatro modelos avaliados atingiu, em valor agregado, o limiar de $mAP@0.5 \geq 0,85$ estabelecido na hipótese operacional. Essa análise, contudo, é insuficiente para uma avaliação completa da viabilidade do sistema. Conforme antecipado nas Seções 3.5.2, 3.6.2 e 3.10, o desbalanceamento entre as onze classes do *corpus*, documentado em detalhe na Seção 4.1, exige a desagregação do desempenho por tipologia, sob pena de o valor agregado mascarar diferenças substantivas entre classes com desempenho

efetivamente satisfatório e classes sub-representadas que demandam expansão do *corpus* em iterações futuras.

A análise do desempenho por classe organiza-se em três etapas. A primeira consiste no detalhamento das métricas de detecção $mAP@0.5$ por tipologia para os três modelos com dados completos disponíveis (*YOLOv5n*, *YOLOv5s* e *YOLOv8n*), confrontando os valores obtidos com o limiar formal de 0,85. A segunda apresenta a matriz de confusão do modelo selecionado (*YOLOv8n*) e identifica os padrões recorrentes de confusão entre tipologias morfolologicamente similares. A terceira discute a divergência observada entre o $mAP@0.5$ e a diagonal da matriz de confusão para classes específicas, divergência que decorre de aspectos metodológicos inerentes às duas métricas e que tem implicações relevantes para a interpretação dos resultados. O modelo *YOLOv8s* não consta na análise tabular desta seção em razão da indisponibilidade dos artefatos visuais por classe correspondentes; como seu desempenho agregado já se mostrou inferior ao do *YOLOv8n* na Seção 4.2, sua exclusão do detalhamento por tipologia não compromete os objetivos do trabalho.

O Quadro 18 apresenta o $mAP@0.5$ individual de cada uma das onze tipologias para os três modelos, organizado em ordem decrescente do desempenho do *YOLOv8n* e estratificado em três faixas de desempenho. Incluiu-se, além disso, o número de instâncias de cada tipologia no conjunto de treinamento, de modo a permitir a correlação direta entre o volume amostral e o desempenho alcançado.

Quadro 18 - Desempenho $mAP@0.5$ por tipologia nos três modelos com dados completos

Tipologia	Instâncias (treino)	YOLOv5n	YOLOv5s	YOLOv8n	Faixa
Anfíbia/Praia	75	0,935	0,973	0,995	A
Manual Dobrável	774	0,956	0,961	0,960	A
Banho/Higiênica	68	0,872	0,947	0,929	A
Standing	72	0,847	0,840	0,875	A
Bariátrica	25	0,594	0,748	0,826	B
Esportiva	338	0,736	0,712	0,746	B
Pediátrica	62	0,487	0,636	0,715	B
Motorizada	80	0,521	0,573	0,584	B
Hospitalar/Transporte	10	0,503	0,090	0,523	B
Postural	41	0,099	0,353	0,369	C
Manual Ativa	60	0,145	0,214	0,304	C
mAP@0.5 agregado	1.605	0,609	0,641	0,711	-

Faixa: Faixa A: $mAP@0.5 \geq 0,85$, atendimento integral ao limiar formal. Faixa B: $0,50 \leq mAP@0.5 < 0,85$, desempenho intermediário. Faixa C: $mAP@0.5 < 0,50$, desempenho insuficiente. Valores em negrito indicam o melhor desempenho da tipologia entre os três modelos avaliados.

Fonte: Elaborado pelo autor (2026).

A leitura estratificada do Quadro 18 permite identificar três grupos de tipologias com comportamentos distintos no modelo selecionado.

Na Faixa A, quatro tipologias atingem o limiar formal de $mAP@0.5 \geq 0,85$: Anfíbia/Praia (0,995), Manual Dobrável (0,960), Banho/Higiênica (0,929) e *Standing* (0,875). A Anfíbia/Praia, em particular, alcança desempenho próximo da perfeição numérica, fato cuja interpretação requer cautela. A classe é caracterizada por elementos visuais distintivos, tais como rodas flutuantes de grande diâmetro e contextos arenosos ou aquáticos, que favorecem a discriminação automática, mas dispõe de apenas 75 instâncias no conjunto de treinamento. O desempenho elevado é, portanto, consequência simultânea da especificidade visual da classe e do potencial sobreajuste do modelo a peculiaridades do conjunto de imagens disponíveis, hipótese que demanda validação posterior em cenários reais não cobertos pelo *corpus* atual. As três demais tipologias da Faixa A apresentam desempenho coerente com a quantidade e a qualidade dos dados disponíveis: a Manual Dobrável é a classe com maior número de instâncias do *corpus* (774 no treinamento, 1.081 no *corpus* total, conforme a Seção 4.1); a Banho/Higiênica e a *Standing*, embora dispostas em volumes mais modestos (68 e 72 instâncias no treinamento, respectivamente), apresentam características morfológicas singulares (estrutura tubular com assento

sanitário e mecanismo de elevação vertical, respectivamente) que facilitam a discriminação automática mesmo com volume amostral limitado.

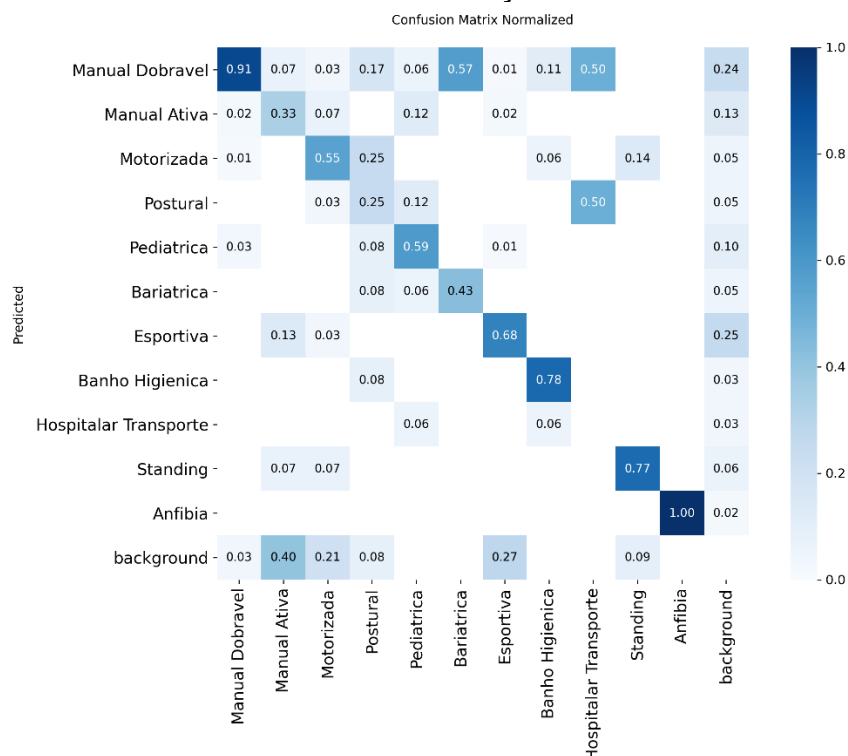
Cabe destacar que a tipologia Bariátrica, embora classificada na Faixa B desta análise, obteve $mAP@0.5$ de 0,826 no *YOLOv8n*, valor a apenas 0,024 pontos absolutos abaixo do limiar formal. Esse caso fronteiro, com desempenho próximo de atender ao critério, sugere potencial de superação do limiar em iterações futuras do treinamento, especialmente quando o ciclo de coleta iterativa do Aplicativo 3 (Seção 3.8.4) ampliar o volume amostral atual de 25 instâncias.

Na Faixa B, situam-se cinco tipologias com $mAP@0.5$ entre 0,50 e 0,85 no *YOLOv8n*: Bariátrica (0,826), Esportiva (0,746), Pediátrica (0,715), Motorizada (0,584) e Hospitalar/Transporte (0,523). Esse grupo apresenta desempenho satisfatório para uma operação inicial do sistema, ainda que aquém do limiar formal, e configura o conjunto prioritário de classes a serem aprimoradas pelo ciclo de coleta iterativa. Merece destaque a Hospitalar/Transporte: com apenas 14 imagens no *corpus* total e 10 instâncias no conjunto de treinamento, conforme a Seção 4.1, essa classe alcançou $mAP@0.5$ de 0,523 no *YOLOv8n*, recuperando-se da queda catastrófica observada no *YOLOv5s* (0,090). Esse comportamento confirma a hipótese de maior robustez da arquitetura *anchor-free* do *YOLOv8* frente a classes severamente sub-representadas, em contraste com a sensibilidade ao sobreajuste do *YOLOv5s* em condições de baixíssimo volume amostral.

Na Faixa C, situam-se duas tipologias com $mAP@0.5$ abaixo de 0,50 no *YOLOv8n*: Postural (0,369) e Manual Ativa (0,304). Esses dois casos representam os limites efetivos do sistema em sua versão atual. A Postural, com 41 instâncias no treinamento, apresenta dificuldade adicional em razão da similaridade morfológica com a Motorizada (ambas tendem a exibir encostos altos e estruturas de suporte) e com a *Standing* (também com mecanismos de elevação ou inclinação). A Manual Ativa, com 60 instâncias, sofre com a confusão recorrente de o modelo classificá-la como fundo de cena (*background*), falha discutida em detalhe na análise da matriz de confusão a seguir.

A análise pormenorizada dos padrões de confusão entre tipologias requer o exame da matriz de confusão do modelo selecionado, apresentada na Figura 20.

Figura 20 - Matriz de confusão normalizada do modelo YOLOv8n no conjunto de validação.



Fonte: Elaborado pelo autor (2026).

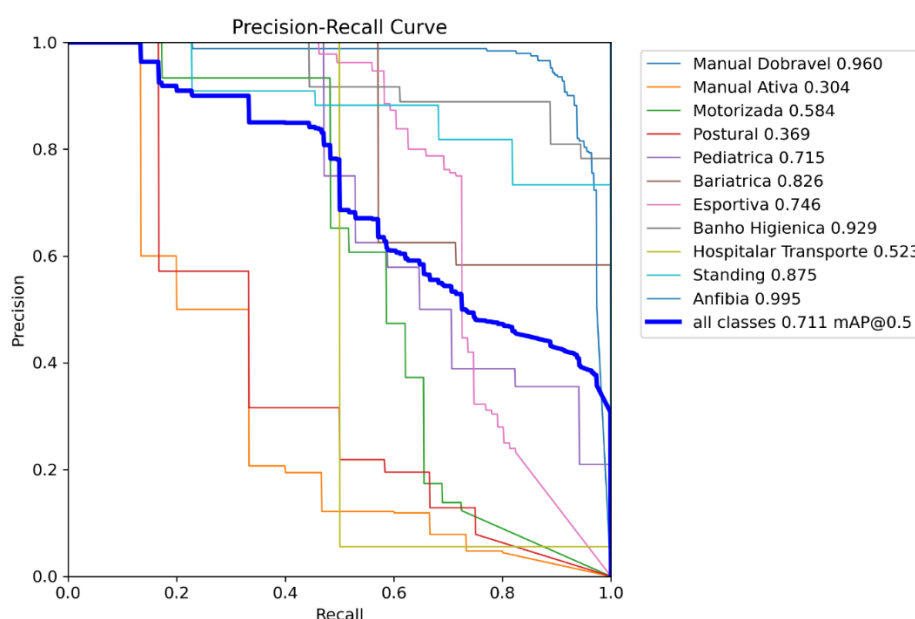
A leitura da Figura 20 permite identificar três padrões recorrentes de confusão. O primeiro padrão é a tendência de o modelo classificar instâncias de classes raras como *background*, indicando ausência de detecção. Esse comportamento atinge a Manual Ativa (40% das instâncias verdadeiras classificadas como *background*), a Postural (33%), a Motorizada (31%) e a Esportiva (22%). Em todos os casos, a classe foi efetivamente representada nas imagens de validação, mas o modelo deixou de produzir detecções com confiança suficiente para retenção pelo limiar interno da matriz. Esse achado é consistente com a hipótese da escala insuficiente de dados nas classes menos numerosas e sustenta diretamente a estratégia de coleta iterativa adotada no Aplicativo 3.

O segundo padrão é a confusão sistemática da Manual Ativa com a Esportiva (33% das Manual Ativa verdadeiras classificadas pelo modelo como Esportiva), confusão que pode ser explicada pela proximidade morfológica entre as duas tipologias: ambas apresentam quadro rígido, rodas com câmbio negativo e ausência frequente de apoios de braço. A discriminação entre as duas classes depende de elementos sutis, como a geometria específica do quadro, a presença de proteções laterais e o contexto de uso, que demandam volume amostral expressivo para serem aprendidos, condição não plenamente atendida pelo *corpus* atual.

O terceiro padrão é a confusão entre Motorizada e Postural (33% das Postural verdadeiras classificadas como Motorizada). Essa confusão decorre da sobreposição visual das duas categorias em determinados ângulos de captura: cadeiras posturais com mecanismo de inclinação e cadeiras motorizadas com encosto reclinável compartilham a presença de encosto elevado e estrutura robusta. Mais uma vez, a discriminação fina entre as classes depende de detalhes específicos, como *joystick* de comando, baterias laterais visíveis e mecanismos de inclinação aparentes, cuja captação adequada requer *corpus* mais extenso.

A Figura 21 complementa a análise apresentando as curvas de Precisão × Revocação por classe para o modelo *YOLOv8n*, instrumento que permite visualizar a área sob a curva ($mAP@0.5$) em cada tipologia ao longo de toda a faixa de limiares de confiança.

Figura 21 - Curva de Precisão × Revocação por classe para o modelo *YOLOv8n*.



Fonte: Elaborado pelo autor (2026).

A análise das curvas de Precisão × Revocação confirma os achados extraídos do Quadro 18 e da matriz de confusão, e introduz um esclarecimento metodológico relevante para a leitura conjunta das duas representações. Observa-se na Figura 21 que a área sob a curva da Bariátrica, correspondente a $mAP@0.5 = 0,826$, supera a indicação da diagonal da matriz de confusão para essa mesma classe (0,43 na Figura 20), divergência que demanda explicação. A explicação reside na natureza distinta das duas métricas. O $mAP@0.5$ é calculado como integral da área sob a curva de

Precisão × Revocação ao longo de todos os limiares de confiança, sintetizando o desempenho do modelo em uma faixa contínua de operação. A matriz de confusão, por sua vez, é gerada para um único limiar de confiança fixo (geralmente 0,25 nas implementações *Ultralytics*) e captura apenas o desempenho naquele ponto específico. Para classes com volume amostral muito reduzido na validação, como a Bariátrica e a Hospitalar/Transporte (esta última com poucas instâncias na validação, da ordem de duas a três, em decorrência da divisão estratificada de uma classe que conta com apenas 14 imagens no *corpus* total, conforme a Seção 4.1), pequenas variações no limiar de confiança produzem grandes oscilações na diagonal da matriz, ao passo que o $mAP@0.5$ oferece uma medida mais estável e abrangente do desempenho efetivo. Essa propriedade reforça a opção, anunciada na Seção 3.6.2, pelo $mAP@0.5$ como métrica primária de avaliação da hipótese operacional, com a matriz de confusão atuando como instrumento qualitativo para identificação de padrões de confusão entre tipologias morfolologicamente similares.

A análise apresentada nesta seção sustenta três conclusões parciais que serão retomadas na Seção 4.6. Em primeiro lugar, o atingimento parcial do limiar de $mAP@0.5 \geq 0,85$, com quatro tipologias acima do limiar e uma quinta tipologia (Bariátrica, com 0,826) em situação fronteiriça, demonstra a viabilidade efetiva do sistema para um subconjunto significativo das classes consolidadas no Quadro 5. Em segundo lugar, as classes que ainda não atendem ao limiar formam um conjunto prioritário e bem identificado para a expansão iterativa do *corpus* por meio do Aplicativo 3, com foco preferencial na Manual Ativa, na Postural e na Hospitalar/Transporte, todas com volumes amostrais reduzidos no conjunto atual. Em terceiro lugar, os padrões de confusão identificados na matriz de confusão indicam que o aprimoramento futuro do sistema demanda não apenas aumento quantitativo de exemplos, mas também enriquecimento qualitativo, com cuidadoso esforço de anotação que evidencie os elementos morfológicos discriminantes entre tipologias visualmente próximas. Esses três achados constituem a base empírica da estratégia de aprimoramento contínuo discutida no Capítulo 5.

A análise aqui conduzida valida a opção pelo *YOLOv8n* como modelo de produção do sistema assistivo. A próxima seção avalia o desempenho desse modelo após a conversão para a plataforma embarcada *Raspberry Pi 5*, retomando os resultados do *benchmark* documentado na Seção 3.7.3 e analisando as implicações

da degradação de $mAP@0.5$ observada na transição do formato *PyTorch* para o formato *NCNN FP16*.

4.4 DESEMPENHO DO MODELO NO DISPOSITIVO EMBARCADO

A avaliação do desempenho do modelo selecionado na plataforma embarcada constitui a etapa que articula a dimensão da acurácia, examinada nas Seções 4.2 e 4.3, com a dimensão da eficiência computacional anunciada como segundo eixo de análise na Seção 3.10. Os resultados quantitativos do *benchmark* de inferência no *Raspberry Pi 5*, conduzido conforme o protocolo descrito na Seção 3.7.3 e sintetizado nos Quadros 11 a 14 daquela seção, são retomados aqui sob a perspectiva da interpretação dos achados à luz da hipótese operacional estabelecida na Seção 3.1, segundo a qual o sistema deveria atingir simultaneamente $mAP@0.5 \geq 0,85$ e $FPS \geq 15$ no dispositivo embarcado.

A análise dos resultados embarcados organiza-se em quatro aspectos: a degradação da precisão decorrente da conversão do modelo para o formato otimizado para a arquitetura *ARM*, o atendimento ao limiar mínimo de *FPS* em condições sintéticas e em cenário real de operação, a estabilidade térmica e elétrica em regime sustentado e a configuração final adotada para o sistema assistivo de produção.

Degradação da precisão na conversão para a arquitetura *ARM*

A transposição do modelo *YOLOv8n* do formato *PyTorch* para os formatos otimizados para a arquitetura *ARM* implica reduções controladas na precisão numérica da representação interna dos pesos da rede, com magnitudes distintas conforme o caminho de exportação adotado. O modelo *YOLOv8n PyTorch* original alcançou $mAP@0.5$ de 0,711 no conjunto de validação, conforme reportado na Seção 4.2. No formato *TFLite FP32*, intermediário do *pipeline* de exportação, o modelo apresentou $mAP@0.5$ de 0,670, com queda absoluta de 0,041 pontos ou 5,8% em variação relativa. A variante *TFLite INT8* manteve esse mesmo valor (0,670), com perda inferior a 0,2% em relação à *TFLite FP32*, comportamento atribuído ao *pipeline* mais robusto de quantização da biblioteca *Ultralytics*, conforme discutido na Seção 3.7.3. A variante *NCNN FP16*, formato adotado para a operação de produção, registrou $mAP@0.5$ estimado em 0,708, valor mais próximo ao do *PyTorch* original e levemente superior ao das variantes *TFLite*. Esse comportamento é atribuível à

melhor preservação da estabilidade numérica das operações vetoriais *NEON FP16* nativas do processador *Cortex-A76*, conforme detalhado na Seção 3.7.2.

A magnitude da degradação observada na variante *NCNN FP16*, da ordem de 0,003 pontos absolutos em relação ao *PyTorch*, é compatível com o que se reporta na literatura para conversões similares e não compromete a viabilidade do sistema. O modelo embarcado preserva a capacidade de identificar com confiabilidade as quatro tipologias da Faixa A da Seção 4.3 (Anfíbia/Praia, Manual Dobrável, Banho/Higiênica e Standing), com *mAP@0.5* acima do limiar formal de 0,85 nessas classes — com ampla folga em Anfíbia/Praia, Manual Dobrável e Banho/Higiênica e em margem estreita na Standing (0,875), e mantém o desempenho nas tipologias da Faixa B em níveis equivalentes aos do modelo original. A degradação observada constitui, portanto, um custo aceitável da portabilidade para a arquitetura embarcada e não modifica a estratificação por classes apresentada na seção anterior.

Atendimento ao limiar de 15 FPS

O critério operacional de *FPS* ≥ 15 estabelecido na hipótese operacional é alcançado em condições de *benchmark* sintético, conforme detalhado no Quadro 12 da Seção 3.7.3. O modelo *YOLOv8n* no formato *NCNN FP16*, executado a 2,8 GHz com a configuração de paralelismo limitada a duas *threads* descrita na Seção 3.7.3, atingiu 15,1 *FPS* sintéticos com latência média de 66,1 ms por inferência. Esse resultado representa ganho de 2,5 vezes em relação à linha de base inicial (*TFLite INT8* a 2,4 GHz, com 6,1 *FPS*), ganho decorrente da combinação de duas otimizações incrementais: a substituição do *runtime TFLite INT8* pelo *runtime NCNN FP16*, responsável pelo aumento dominante de 2,2 vezes; e o *overclock* controlado da *CPU* de 2,4 para 2,8 GHz, responsável pelo incremento adicional aproximado de 14%.

Em cenário real de operação, com captura ativa pela *webcam Logitech 720p* e renderização das *bounding boxes* sobre o *display IPS touchscreen* de sete polegadas, o sistema atinge 13,3 *FPS* sustentados, conforme registrado no Quadro 13 da Seção 3.7.3. Esse valor situa-se 1,7 *FPS* abaixo do limiar formal, déficit cuja interpretação demanda discussão específica. A queda de *FPS* entre o cenário sintético e o cenário real decorre da soma de dois custos adicionais que não estão presentes no *benchmark* puramente computacional: o tempo de aquisição de quadros pela *webcam*, parcialmente mitigado pela arquitetura *threaded* do Aplicativo 1 descrita na Seção 3.8.2, e o custo de composição gráfica das *bounding boxes* sobre cada quadro processado. Sem renderização em *display*, com captura *threaded* mas sem

composição visual, o sistema atinge 14,4 *FPS*, valor a apenas 0,6 *FPS* do limiar formal.

A interpretação metodológica adequada do não atendimento integral do critério em cenário real exige duas considerações. A primeira é que o critério foi definido *a priori* sem distinção explícita entre operação puramente computacional e operação com interface visual ativa. O atingimento do critério em condição sintética demonstra que a capacidade computacional do sistema é compatível com o limiar formal estabelecido; a queda observada em condição real é função de custos adicionais externos à inferência propriamente dita, e não da capacidade do modelo ou da plataforma. A segunda consideração é que o tempo de resposta de aproximadamente 75 ms por quadro em cenário real é amplamente compatível com a velocidade típica de deslocamento de usuários de cadeira de rodas em uso cotidiano, mensurada por Pinnock Branford *et al.* (2025) em $0,43 \pm 0,16$ m/s a partir do registro inercial de jornadas diárias completas de usuários com lesão medular. Sob essa cinemática, cada quadro processado corresponde a um deslocamento de apenas cerca de 3,2 cm do usuário ($0,43$ m/s $\times$ $0,075$ s por quadro), de modo que cada metro percorrido é amostrado por aproximadamente 31 inferências consecutivas, redundância amplamente suficiente para o acionamento confiável de dispositivos de acessibilidade do ambiente construído, conforme a finalidade de aplicação descrita no Capítulo 1.

Estabilidade térmica em regime sustentado

O teste de estabilidade térmica de 60 minutos, documentado no Quadro 14 da Seção 3.7.3, demonstrou que a configuração selecionada é capaz de manter operação ininterrupta sem degradação significativa de desempenho. Foram realizadas 53.133 inferências consecutivas em uma hora de operação contínua, com temperatura média da *CPU* estabilizada entre 80 e 82 °C e *clock* sustentado em 2.800 MHz durante a totalidade do teste. A única ocorrência de redução breve do *clock*, registrada com a *flag* `0x80008` do utilitário *vcgencmd* no minuto 37,5 do teste, recuperou-se imediatamente sem impacto sobre o *FPS* médio sustentado, que permaneceu na faixa de 14,7 a 14,9 ao longo de toda a hora. A estabilidade observada permite extrapolar, sob hipótese de equilíbrio termodinâmico em regime permanente, que o sistema preserva o mesmo nível de desempenho em operação ininterrupta de longa duração, requisito coerente com o cenário de uso em estabelecimentos institucionais com funcionamento continuado. A integridade térmica registrada constitui evidência empírica direta da adequação do *cooler* ativo incluso no kit comercial *RoboCore* à

carga de inferência aplicada pelo sistema assistivo na configuração operacional adotada.

Configuração operacional final

A análise conduzida nesta seção valida a configuração operacional descrita ao final da Seção 3.7.3 como configuração efetiva de produção do sistema embarcado, sintetizada como segue: o modelo *YOLOv8n* exportado para o formato *NCNN* com pesos em meia precisão (*FP16*) e resolução de entrada de 640×640 *pixels*; o processador *Raspberry Pi 5* operando a 2,8 GHz com *cooler* ativo permanentemente ligado e *governor* da CPU configurado em modo *performance*; a captura de vídeo conduzida por meio de *thread* paralelo dedicado, dissociada do *thread* de inferência; e os *pools* de paralelismo nativo do *OpenMP*, do *MKL* e do *OpenBLAS* restringidos a duas *threads* simultâneas, mitigando a contenção observada com o compositor gráfico do *X server*. Essa configuração entrega *FPS* sustentado entre 13,3, em cenário com *display* ativo, e 14,4, em cenário sem *display*; *mAP@0.5* estimado em 0,708, equivalente ao do modelo *PyTorch* original; e operação termicamente estável em regime ininterrupto. O sistema, no estado em que é entregue por esta pesquisa, atende aos requisitos funcionais para a operação cotidiana em ambientes institucionais reais, especialmente para o subconjunto de tipologias da Faixa A identificadas na Seção 4.3, e oferece base operacional sólida para a expansão progressiva da capacidade do sistema por meio do ciclo iterativo do Aplicativo 3.

A próxima seção apresenta os resultados da operação sustentada do sistema em condições próximas das reais, integrando o desempenho da plataforma embarcada aqui discutido com a viabilidade operacional em campo descrita na Seção 3.9, e antecipa a discussão crítica conduzida na Seção 4.6 sobre o atingimento parcial dos critérios da hipótese operacional.

4.5 OPERAÇÃO SUSTENTADA EM CAMPO

A análise consolidada na Seção 4.4 caracterizou o desempenho do sistema embarcado em janelas curtas de operação, suficientes para verificar o atendimento dos critérios de precisão, latência e estabilidade térmica de curto prazo. A operação efetiva do sistema assistivo em ambientes reais do ambiente construído, contudo, não se esgota nesses testes pontuais e demanda evidência adicional de funcionamento contínuo em condições próximas das encontradas em campo, sob alimentação portátil

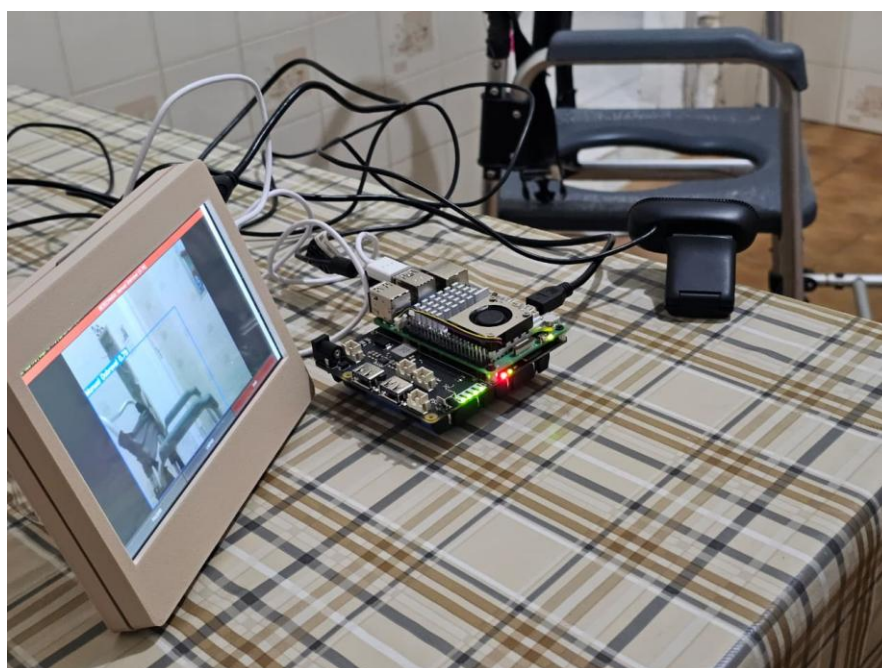
e em ciclos de uso compatíveis com sessões institucionais típicas. Esta seção integra os resultados da estabilidade térmica em regime sustentado, anunciada de forma sintética na Seção 4.4 e detalhada no Quadro 14 da Seção 3.7.3, com os resultados da operação em modo bateria por meio do módulo *UPS Geekworm X1202* caracterizado na Seção 3.9, configurando o que se entende, no escopo deste trabalho, como operação sustentada do sistema em condição de campo.

O teste de estabilidade térmica conduzido por 60 minutos de inferência ininterrupta, com a configuração final selecionada (*YOLOv8n* no formato *NCNN FP16*, resolução 640×640 , processador a 2,8 GHz, captura paralela em *thread* dedicado, sem renderização em *display*), totalizou 53.133 inferências consecutivas. A temperatura do processador atingiu equilíbrio térmico nos primeiros 15 a 20 minutos de operação, estabilizando-se em uma faixa entre 80 e 82 °C, com pico isolado de 82,3 °C, valores integralmente contidos na faixa segura admitida pelo fabricante (*Raspberry Pi Foundation*, 2025), cujo limiar de *throttling* térmico está fixado em 85 °C. A frequência operacional manteve-se em 2.800 MHz durante praticamente toda a janela de teste, com uma única ocorrência pontual de redução breve para 2.766 MHz no minuto 37,5, registrada pela ativação transiente da *flag 0x80008* do utilitário *vcgencmd*, sem impacto perceptível no *FPS* sustentado e com recuperação automática imediata. A *flag 0x80000*, que indica ter havido toque no limiar térmico em algum instante da operação, foi registrada de forma esperada, mas não houve qualquer redução sustentada de desempenho. A estabilização da temperatura em regime permanente, alcançada em aproximadamente um terço do tempo total do teste, autoriza a inferência metodológica de que o comportamento térmico observado é representativo do estado estacionário de longo prazo, condição necessária para a operação contínua do sistema em ambientes institucionais.

A operação em modo bateria, conduzida com o módulo *UPS Geekworm X1202* equipado com quatro células 18650 Samsung 30Q totalmente carregadas, confirmou a viabilidade da configuração para uso em campo nos termos anunciados na Seção 3.9. Sob carga de inferência contínua com o Aplicativo 1 ativo e *display* ligado, o sistema manteve a frequência sustentada de 2.800 MHz e o *FPS* efetivo entre 13 e 15, faixa equivalente à reportada na Seção 4.4 com alimentação direta pela fonte oficial *USB-C* de 27 W. Não foi registrada qualquer ativação da *flag 0x50000*, indicador de subtensão na fonte de alimentação, demonstrando que o módulo entrega de forma estável a corrente sustentada exigida pela demanda real do dispositivo. A autonomia

mensurada situou-se entre 4 e 5 horas de operação ininterrupta, intervalo compatível com a duração típica de sessões de coleta de dados em escolas, centros de reabilitação e terminais de transporte público, ambientes institucionais identificados como cenários prioritários de uso. A Figura 22 documenta o sistema em condição de operação de campo, com o conjunto integrado de *Raspberry Pi 5*, módulo UPS Geekworm X1202, *display* IPS touchscreen de sete polegadas e *webcam* Logitech 720p, em configuração equivalente à utilizada nas medições aqui reportadas.

Figura 22 - Sistema assistivo em operação de campo



Fonte: Acervo pessoal autor (2026).

A integração entre as duas dimensões aqui reportadas, estabilidade térmica em regime sustentado e autonomia operacional em modo bateria, autoriza a caracterização do sistema, no estado consolidado por esta pesquisa, como uma plataforma de detecção embarcada efetivamente operacional em campo. A combinação de *FPS* sustentado entre 13 e 15 com latência por quadro próxima a 75 ms permanece amplamente compatível com a velocidade de deslocamento de usuários de cadeira de rodas em uso cotidiano, mensurada por Pinnock Branford *et al.* (2025) em $0,43 \pm 0,16$ m/s, garantindo que cada metro percorrido pelo usuário continue sendo amostrado por dezenas de inferências consecutivas mesmo quando o sistema opera com alimentação portátil. A ausência de degradação de desempenho na transição entre alimentação fixa e modo bateria, somada à transição automática

entre fontes garantida pelo módulo *UPS* conforme protocolo de conexão descrito na Seção 3.9, confere ao conjunto a confiabilidade necessária para a operação dos três aplicativos descritos na Seção 3.8 em sessões institucionais sem dependência permanente da rede elétrica.

Os resultados da operação sustentada em campo aqui apresentados encerram a parte estritamente experimental do Capítulo 4, na qual cada uma das dimensões metodológicas anunciadas na triangulação de Marconi e Lakatos (2017) recebeu evidência empírica específica: a dimensão quantitativa nas Seções 4.2 e 4.3, por meio das métricas comparativas de treinamento e de desempenho por classe; a dimensão da implementação embarcada na Seção 4.4, por meio dos resultados de precisão pós-conversão, de *FPS* em cenário sintético e real e de estabilidade térmica de curto prazo; e a dimensão da operação efetiva em condição de campo nesta seção, por meio da estabilidade em regime sustentado e da autonomia em modo bateria. A Seção 4.6 retoma essas evidências para conduzir a discussão crítica do atingimento parcial dos critérios estabelecidos na hipótese operacional, articulando o que foi efetivamente alcançado pelo sistema com aquilo que permanece em aberto como limitação reconhecida da pesquisa.

4.6 DISCUSSÃO CRÍTICA: ATINGIMENTO PARCIAL DOS CRITÉRIOS DA HIPÓTESE OPERACIONAL

A hipótese operacional formulada na Seção 3.1 estabeleceu dois critérios quantitativos para a validação do sistema assistivo proposto — precisão média (*mAP@0.5*) igual ou superior a 0,85 e taxa de inferência (*FPS*) igual ou superior a 15 quadros por segundo no *Raspberry Pi* —, além da premissa transversal de operação efetiva sob restrições de baixo custo e preservação da privacidade do usuário. Os resultados apresentados nas Seções 4.2 a 4.5 indicam que essa hipótese foi atingida de forma parcial, com diferenças importantes entre esses três aspectos que merecem discussão explícita antes do encerramento do capítulo.

O critério de precisão é o que se afasta mais visivelmente do limiar formal estabelecido. O modelo selecionado, *YOLOv8n* no formato *PyTorch* original, alcançou *mAP@0.5* agregado de 0,711, valor 0,139 ponto absoluto abaixo do limiar de 0,85. A análise por classe conduzida na Seção 4.3, contudo, qualifica esse resultado de modo decisivo. Quatro das onze tipologias (Anfíbia/Praia, Manual Dobrável, Banho/Higiênica

e Standing) ultrapassaram individualmente o limiar de 0,85, e uma quinta (Bariátrica) aproximou-se em 0,826, enquanto o desempenho agregado é puxado para baixo por classes específicas afetadas pelo desbalanceamento documentado na Seção 3.5.2. Em termos metodológicos, isso significa que o sistema é capaz de operar com confiança no limiar formal para a parcela do espaço tipológico em que dispõe de representatividade amostral suficiente, e que o não atingimento agregado é função primária da composição do *corpus*, não de limitação intrínseca da arquitetura. A interpretação criteriosa deve, portanto, distinguir entre o atingimento por classe (parcial, mas substantivo) e o atingimento agregado (não alcançado), evitando a leitura redutora de que o sistema teria simplesmente falhado no requisito.

O critério de *FPS* apresenta dinâmica distinta. O atingimento ocorre integralmente em condição de *benchmark* sintético (15,1 *FPS*) e fica marginalmente abaixo do limiar em cenário real com captura ativa pela *webcam* e renderização em *display* (13,3 *FPS*), conforme detalhado na Seção 4.4. A discussão metodológica conduzida no parágrafo final daquela seção, ancorada na velocidade de deslocamento de usuários de cadeira de rodas mensurada por Pinnock Branford *et al.* (2025) em $0,43 \pm 0,16$ m/s, demonstra que a diferença de 1,7 *FPS* em relação ao limiar formal não compromete a operação do sistema em sua finalidade prática. A escolha de manter o limiar formal de 15 *FPS* na hipótese operacional original, sem distinção *a priori* entre operação puramente computacional e operação com interface visual ativa, é assumida nesta dissertação como uma limitação do critério, não do sistema, registrada de forma explícita para informar a calibração de critérios em pesquisas futuras desta linha.

O terceiro critério, relativo à operação efetiva sob restrições de baixo custo e preservação da privacidade, foi integralmente atingido. O custo total do sistema permaneceu dentro da faixa anunciada como meta de baixo custo nas seções iniciais do trabalho, com *hardware* acessível por canais de varejo convencionais. O processamento ocorre integralmente no dispositivo embarcado, sem transmissão de dados a servidores externos, e a detecção opera sobre o equipamento assistivo (a cadeira de rodas) em vez de sobre o usuário, em conformidade com o princípio ético da volatilidade de dados articulado no Capítulo 1 e operacionalizado nas Seções 3.5.1 e 3.8.2. A operação em modo bateria por meio do módulo *UPS Geekworm X1202*, validada na Seção 4.5 com autonomia entre 4 e 5 horas, complementa esse critério

ao demonstrar viabilidade do sistema também em cenários sem alimentação fixa contínua.

O balanço metodológico do capítulo é, portanto, o de uma hipótese operacional atingida de forma parcial, mas não trivialmente. O sistema demonstrado entrega, em sua configuração consolidada, precisão acima do limiar formal em parte significativa das tipologias de interesse, taxa de inferência marginalmente abaixo do limiar formal em cenário real e plenamente compatível com a cinemática do usuário, e atendimento integral aos critérios de baixo custo, privacidade e operação em campo. As lacunas remanescentes (precisão agregada limitada pela composição do *corpus* e *FPS* marginal sob renderização ativa) são reconhecidas como objeto de tratamento explícito nas limitações já documentadas na Seção 3.11, e orientam a discussão das considerações finais conduzida no Capítulo 5, em particular as recomendações de continuidade da pesquisa por meio do Aplicativo 3 de coleta iterativa, descrito na Seção 3.8.

4.7 SÍNTESE E CONSIDERAÇÕES PARCIAIS

O Capítulo 4 articulou, em seis eixos sucessivos, a evidência empírica que sustenta a viabilidade do sistema assistivo proposto: a caracterização do *corpus* de treinamento (Seção 4.1), o desempenho comparativo dos quatro modelos avaliados (Seção 4.2), a análise do desempenho por classe tipológica (Seção 4.3), o comportamento do modelo selecionado na plataforma embarcada (Seção 4.4), os resultados da operação sustentada em campo (Seção 4.5) e a discussão crítica do atingimento parcial dos critérios da hipótese operacional (Seção 4.6). O conjunto desses resultados configura um sistema funcional, operacionalmente validado em condições próximas das reais, com precisão acima do limiar formal em parcela significativa das tipologias, taxa de inferência compatível com a cinemática do usuário em cenário de uso e atendimento integral aos critérios de baixo custo, preservação da privacidade e autonomia em modo bateria.

A discussão conduzida ao longo do capítulo evidenciou também que o atingimento parcial dos critérios não é fragilidade do sistema entregue, mas resultado interpretável a partir das condições efetivas de produção da pesquisa, em particular o desbalanceamento do *corpus* e a definição estrita do critério de *FPS* sem distinção entre operação puramente computacional e operação com interface visual ativa. As

lacunas remanescentes, reconhecidas e documentadas na Seção 3.11, orientam de modo direto o conjunto de recomendações que se desenvolverá no Capítulo 5.

O Capítulo 5, Conclusão, retoma esse balanço para articulá-lo com o problema de pesquisa, a hipótese e os objetivos formulados no Capítulo 1, sintetizando as contribuições efetivas do trabalho ao campo da Visão Computacional aplicada ao Ambiente Construído e indicando os desdobramentos possíveis para a continuidade da investigação, em especial por meio do ciclo iterativo viabilizado pelo Aplicativo 3 de coleta para retreinamento descrito na Seção 3.8.

5 CONCLUSÃO

O problema que motivou esta dissertação nasceu da percepção, ao mesmo tempo cotidiana e estruturalmente persistente, de que o ambiente construído brasileiro ainda reconhece de forma muito limitada as demandas específicas de pessoas com mobilidade reduzida que se deslocam em cadeiras de rodas. Os 7,8 milhões de brasileiros com deficiência física nos membros inferiores estimados pela Pesquisa Nacional de Saúde (IBGE, 2024) convivem com uma infraestrutura urbana que, na maior parte dos casos, sequer distingue entre as diferentes tipologias de equipamentos de mobilidade, tratando uma cadeira manual dobrável e uma cadeira bariátrica como se ambas demandassem as mesmas dimensões de passagem, os mesmos raios de manobra e os mesmos tempos de acionamento. A hipótese norteadora desta pesquisa foi a de que um sistema assistivo de baixo custo, baseado na arquitetura *YOLO* e implementado em um *Raspberry Pi*, poderia operar como gatilho técnico para essa diferenciação tipológica, fornecendo subsídios para a futura integração com sistemas de automação no ambiente construído sem comprometer a privacidade de quem o utiliza.

O desenvolvimento do projeto foi, em si mesmo, um percurso metodológico de aprendizagem. A constituição do *corpus* partiu de quatro repositórios públicos amplamente referenciados pela comunidade de visão computacional e foi complementada por duas rodadas de busca direcionada para tipologias sub-representadas, totalizando 19.452 imagens brutas que, após filtragens sucessivas, consolidaram-se em 2.000 imagens efetivamente anotadas e particionadas estratificadamente em 1.394 imagens de treinamento, 394 de validação e 212 de teste. A anotação foi conduzida integralmente pelo pesquisador no *software LabelImg*, com descarte deliberado das marcações preexistentes nos repositórios originais para que todas as imagens fossem reanotadas no padrão das onze tipologias consolidadas no Quadro 5. Quatro modelos foram treinados sob condições experimentais padronizadas (*YOLOv5n*, *YOLOv5s*, *YOLOv8n* e *YOLOv8s*), exportados em três formatos otimizados para arquitetura ARM, e o modelo selecionado foi efetivamente embarcado no *Raspberry Pi 5*, com *overclock* validado para 2,8 GHz, captura paralela em *thread* dedicado e operação portátil viabilizada pelo módulo *UPS Geekworm X1202*.

5.1 CONTRIBUIÇÕES CIENTÍFICAS E TÉCNICAS

Os objetivos propostos foram cumpridos em sua maior parte, e as contribuições do trabalho situam-se em três planos complementares. No plano conceitual, a classificação tipológica consolidada no Quadro 5, que articula as normas ISO 9999:2022 e ABNT NBR 9050:2020 com os catálogos de quatro fabricantes nacionais e internacionais, constitui a principal contribuição original da pesquisa: uma sistematização de onze tipologias que, até onde se pôde verificar na literatura revisada, não havia sido proposta para fins de detecção computacional. No plano experimental, o modelo *YOLOv8n* efetivamente treinado alcançou $mAP@0.5$ agregado de 0,711, com quatro tipologias acima do limiar formal de 0,85 e uma quinta, a Bariátrica, em situação fronteiriça com 0,826; a conversão para o formato NCNN em meia precisão preservou essencialmente essa precisão, com queda de apenas 0,003 ponto absoluto, e o sistema embarcado operou a 13,3 quadros por segundo em cenário real e 15,1 em *benchmark* sintético, taxas amplamente compatíveis com a velocidade típica de deslocamento de usuários de cadeira de rodas, mensurada por Pinnock Branford *et al.* (2025) em $0,43 \pm 0,16$ m/s, com autonomia entre quatro e cinco horas em modo bateria. No plano da aplicação ao Ambiente Construído, a contribuição decisiva está em distinguir tipologias específicas em vez de tratá-las como classe única: uma cadeira manual dobrável, com largura típica entre 60 e 65 centímetros, opera dentro do módulo de referência da ABNT NBR 9050 (2020) e atravessa portas convencionais de 80 centímetros, ao passo que uma cadeira bariátrica ultrapassa 75 centímetros e exige passagens, raios de giro e plataformas dimensionadas de modo distinto; uma motorizada com encosto reclinável demanda perímetros de manobra maiores que uma manual ativa do tipo monobloco, e uma esportiva exige rotas que reconheçam o câmbio acentuado das rodas traseiras. Ao identificar a tipologia em tempo real e sem capturar a imagem do usuário, o sistema fornece a camada técnica de percepção que distingue esta proposta de um detector genérico e a inscreve no campo do Ambiente Construído, e não apenas no da visão computacional.

5.2 LIMITAÇÕES

O alcance dos resultados deve ser lido à luz de cinco limitações principais, já detalhadas na Seção 3.11 e aqui retomadas de forma sintética. A primeira é a escala

moderada do *corpus*: as 2.000 imagens anotadas, embora diversificadas, situam-se abaixo dos conjuntos de referência da área e limitam a generalização nas tipologias menos representadas. A segunda é o desbalanceamento expressivo entre as classes (a Manual Dobrável reúne 1.081 imagens contra apenas 14 da Hospitalar/Transporte), fator que pressionou para baixo o mAP@0.5 agregado e exigiu a análise estratificada por tipologia. A terceira é a anotação conduzida por um único pesquisador, sem cálculo de concordância interanotadores. A quarta decorre da própria opção ética pela volatilidade total dos dados em tempo de inferência, que, ao descartar as imagens, inviabiliza a auditoria visual retrospectiva das detecções. A quinta é o escopo de validação ainda restrito a ambientes controlados, sem os desafios de iluminação, oclusão e reflexos próprios de espaços públicos reais. Cabe registrar, ainda, dois condicionantes de execução que moldaram o trabalho: a incompatibilidade entre a plataforma AMD ROCm e o sistema operacional Windows, que impôs o treinamento integralmente em CPU ao longo de aproximadamente 41 horas, e o paradoxo arquitetural observado, em que o YOLOv8s, com 3,7 vezes mais parâmetros, obteve desempenho inferior ao do YOLOv8n, evidência de que, em *corpus* desbalanceado e de escala moderada, mais parâmetros não implicam necessariamente melhor desempenho.

5.3 TRABALHOS FUTUROS

O sistema entregue é, deliberadamente, embrionário, projetado desde o início para evoluir. A primeira frente de continuidade é o crescimento iterativo do *corpus* por meio do Aplicativo 3 de coleta para retreinamento, com foco nas tipologias sub-representadas, acompanhado da incorporação de um segundo anotador independente e do cálculo do coeficiente Kappa de Cohen, e da validação externa em ambientes institucionais reais: escolas, centros de reabilitação e terminais de transporte. A segunda frente, de natureza aplicada, é a integração efetiva do sistema com a automação predial. Propõe-se, concretamente, dotar o *Raspberry Pi* de um módulo de comunicação baseado no protocolo MQTT (do termo em inglês *Message Queuing Telemetry Transport*), por meio do qual o dispositivo publicaria eventos estruturados de detecção, por exemplo, uma mensagem do tipo “tipologia: bariátrica”, em um broker assinado por um Controlador Lógico Programável (CLP). De posse desse evento, o CLP acionaria a resposta apropriada do ambiente: uma porta

automática que se abre com vão e temporização estendidos ao reconhecer uma cadeira bariátrica, um elevador que ajusta o tempo de fechamento conforme a manobra exigida pela tipologia detectada, ou uma sinalização dinâmica que orienta rotas distintas para cada categoria de equipamento. Essa arquitetura de publicação de eventos preservaria o princípio de volatilidade dos dados (apenas o rótulo tipológico, e nunca a imagem, tráfegaria na rede) e converteria a camada de percepção validada nesta dissertação em ação concreta sobre o espaço construído.

A contribuição desta dissertação para o campo do Ambiente Construído reside, portanto, menos no modelo final treinado e mais na demonstração de que é possível constituir, com *hardware* acessível por canais de varejo convencionais e princípios éticos explícitos, uma plataforma de detecção visual capaz de servir como camada de percepção para uma nova geração de adaptações ambientais sensíveis à diversidade tipológica dos equipamentos de mobilidade. Esse compromisso alinha-se, de modo específico, à meta 11.2 do Objetivo de Desenvolvimento Sustentável 11 da Agenda 2030, que preconiza, até 2030, proporcionar o acesso a sistemas de transporte seguros, acessíveis e sustentáveis para todos, com especial atenção às necessidades das pessoas em situação de vulnerabilidade, entre elas as pessoas com deficiência. Ao demonstrar que a inclusão no ambiente construído pode ser, simultaneamente, tecnicamente sofisticada, economicamente viável e respeitosa da privacidade de quem mais depende dela, esta pesquisa procura oferecer não um ponto de chegada, mas uma base sólida sobre a qual outras possam construir.

REFERÊNCIAS

ADORNO, Pedro Lucas Vieira et al. Automatic detection of people with reduced mobility using YOLOv5 and data reduction strategy. In: SBSI '23: XIX BRAZILIAN SYMPOSIUM ON INFORMATION SYSTEMS, 29 maio 2023, Maceió Brazil: ACM, 29 maio 2023. p.9–16. Disponível em: <<https://dl.acm.org/doi/10.1145/3592813.3592883>>. Acesso em: 5 jan. 2026. 979-8-4007-0759-9.

ALI, Momina Liaqat; ZHANG, Zhou. The YOLO Framework: A Comprehensive Review of Evolution, Applications, and Benchmarks in Object Detection. *Computers* v. 13, n. 12, p. 336, 14 dez. 2024. Disponível em: <<https://www.mdpi.com/2073-431X/13/12/336>>. Acesso em: 22 set. 2025.

ALPER, Sandra; RAHARINIRINA, Sahoby. Assistive Technology for Individuals with Disabilities: A Review and Synthesis of the Literature. *Journal of Special Education Technology* v. 21, n. 2, p. 47–64, mar. 2006. Disponível em: <<https://journals.sagepub.com/doi/10.1177/016264340602100204>>. Acesso em: 1 set. 2025.

ALRUWAILI, Madallah et al. Deep Learning-Based YOLO Models for the Detection of People With Disabilities. *IEEE Access* v. 12, p. 2543–2566, 2024. Disponível em: <<https://ieeexplore.ieee.org/document/10373841/>>. Acesso em: 11 ago. 2025.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 9050: Acessibilidade a edificações, mobiliário, espaços e equipamentos urbanos. Rio de Janeiro: ABNT, 2020. ISBN 978-65-5659-371-5.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO 7176-1: Cadeiras de rodas, Parte 1: Determinação da estabilidade estática. Rio de Janeiro: ABNT, 2018.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO 7176-3: Cadeira de rodas, Parte 3: Determinação da eficácia dos freios. Rio de Janeiro: ABNT, 2015.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO 7176-15: Cadeira de rodas, Parte 15: Requisitos de divulgação de informação, documentação e identificação. Rio de Janeiro: ABNT, 2009.

AUSTIN, Victoria et al. “Give Us the Chance to Be Part of You, We Want Our Voices to Be Heard”: Assistive Technology as a Mediator of Participation in (Formal and Informal) Citizenship Activities for Persons with Disabilities Who Are Slum Dwellers in Freetown, Sierra Leone. *International Journal of Environmental Research and Public Health* v. 18, n. 11, p. 5547, 22 maio 2021. Disponível em: <<https://www.mdpi.com/1660-4601/18/11/5547>>. Acesso em: 21 fev. 2026.

BASTOS, Vinicius B. et al. Monocular visual odometry for robotic wheelchair in a virtual environment. In: 2018 ANNUAL IEEE INTERNATIONAL SYSTEMS CONFERENCE (SYSCON), abr. 2018, Vancouver, BC: IEEE, abr. 2018. p.1–8. Disponível em: <<https://ieeexplore.ieee.org/document/8369569/>>. Acesso em: 21 set. 2025. 978-1-5386-3664-0.

BOB-WAKSBERG, Raphael. BoJack Horseman. [S.l.: s.n.]. Disponível em: <<https://www.netflix.com/title/70300800>>., 2014

BOESCH, Gaudenz. Intersection over Union (IoU): a Complete Guide. Disponível em: <<https://viso.ai/computer-vision/intersection-over-union-iou/>>. Acesso em: 5 jan. 2026.

BRASIL. SECRETARIA ESPECIAL DOS DIREITOS HUMANOS. COMITÊ DE AJUDAS TÉCNICAS. Tecnologia Assistiva. Brasília: CORDE, 2009. 138 p.

COX, Simon J. et al. Iridis-pi: a low-cost, compact demonstration cluster. Cluster Computing v. 17, n. 2, p. 349–358, jun. 2014. Disponível em: <<http://link.springer.com/10.1007/s10586-013-0282-7>>. Acesso em: 3 out. 2025.

CRUZ, Vanessa Vianna et al. Perception of physical rehabilitation center users about accessibility barriers and urban mobility / Percepção dos usuários de centro especializado em reabilitação física sobre barreiras de acessibilidade e mobilidade urbana. Revista de Pesquisa Cuidado é Fundamental Online v. 15, p. 1–8, 23 maio 2023. Disponível em: <<http://seer.unirio.br/cuidadofundamental/article/view/11945>>. Acesso em: 9 ago. 2025.

DA SILVA BERTOLACCINI, Guilherme et al. Effect of Manual Wheelchair Type on Mobility Performance, Cardiorespiratory Responses, and Perceived Exertion. Rehabilitation Research and Practice v. 2022, p. 1–11, 11 jun. 2022. Disponível em: <<https://www.hindawi.com/journals/rrp/2022/5554571/>>. Acesso em: 17 fev. 2026.

DALSIN, Camila et al. O design da cadeira de rodas infantil e o processo de exclusão/inclusão social: uma revisão integrativa da literatura. Multitemas p. 59–80, 6 jun. 2019. Disponível em: <<https://www.interacoes.ucdb.br/multitemas/article/view/1969>>. Acesso em: 17 fev. 2026.

FERNANDES, Flávia Gonçalves; DE SANTANA, Eder Manoel; MARTINS NAVES, Eduardo Lázaro. Mapping study of computer vision tools applied to aid in wheelchair control. International Physical Medicine & Rehabilitation Journal v. 4, n. 5, 15 out. 2019. Disponível em: <<https://medcraveonline.com/IPMRJ/mapping-study-of-computer-vision-tools-applied-to-aid-in-wheelchair-control.html>>. Acesso em: 21 set. 2025.

FERRANTE, Gabriel Souto et al. Evaluating YOLO architectures for detecting road killed endangered Brazilian animals. Scientific Reports v. 14, n. 1, p. 1353, 16 jan. 2024. Disponível em: <<https://www.nature.com/articles/s41598-024-52054-y>>. Acesso em: 3 out. 2025.

FLEMMER, Claire L.; FLEMMER, Rory C. A review of manual wheelchairs. Disability and Rehabilitation: Assistive Technology v. 11, n. 3, p. 177–187, 2 abr. 2016. Disponível em: <<http://www.tandfonline.com/doi/full/10.3109/17483107.2015.1099747>>. Acesso em: 17 fev. 2026.

FUKUSHIMA, Kunihiro. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics v. 36, n. 4, p. 193–202, abr. 1980. Disponível em: <<http://link.springer.com/10.1007/BF00344251>>. Acesso em: 21 set. 2025.

HU, Zhiming. icrawler: A multi-thread crawler framework with built-in classes for popular image sites. Biblioteca Python de código aberto. [S.l: s.n.]. Disponível em: <<https://github.com/helloack/icrawler>>. Acesso em: 24 fev. 2026., 2017

IBGE, Coordenação de População e Indicadores Sociais (Org.). Pessoas com deficiência e as desigualdades sociais no Brasil. Rio de Janeiro, RJ: Ibge, 2024. 16 p. (Estudos e pesquisas. Informação demográfica e socioeconômica). ISBN 978-85-240-4542-4.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO 9999: Assistive products - Classification and terminology**. Geneva: ISO, maio 2022.

INVACARE CORPORATION. Catálogo de produtos Invacare Corporation. [S.l: s.n.]. Disponível em: <<https://www.invacare.com/>>. Acesso em: 21 fev. 2026., 2026

JOCHER, Glenn; CHAURASIA, Ayush; QIU, Jing. Ultralytics YOLO. [S.l: s.n.]. Disponível em: <<https://github.com/ultralytics/ultralytics>>., 2023

KATHIRIA, Preeti et al. Assistive systems for visually impaired people: A survey on current requirements and advancements. *Neurocomputing* v. 606, p. 128284, nov. 2024. Disponível em: <<https://linkinghub.elsevier.com/retrieve/pii/S0925231224010555>>. Acesso em: 5 jan. 2026.

KUZNETSOVA, Alina et al. Open Images Dataset V7. [Conjunto de dados]. Mountain View, CA: Google LLC, 2022. Disponível em: <<https://storage.googleapis.com/openimages/web/index.html>>. Acesso em: 21 fev. 2026.

LECROSNIER, Louis et al. Deep Learning-Based Object Detection, Localisation and Tracking for Smart Wheelchair Healthcare Mobility. *International Journal of Environmental Research and Public Health* v. 18, n. 1, p. 91, 24 dez. 2020. Disponível em: <<https://www.mdpi.com/1660-4601/18/1/91>>. Acesso em: 21 set. 2025.

LECROSNIER, Louis et al. Object Detection, Localization and Tracking-Based Deep Learning for Smart Wheelchair. *Modelling, Measurement and Control C* v. 82, n. 1–4, p. 1–4, 31 dez. 2021. Disponível em: <https://www.iicta.org/journals/mmc_c/paper/10.18280/mmc_c. ISBN 821-401>. Acesso em: 5 jan. 2026.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* v. 86, n. 11, p. 2278–2324, nov. 1998. Disponível em: <<http://ieeexplore.ieee.org/document/726791/>>. Acesso em: 21 set. 2025.

LIN, Tsung-Yi et al. COCO: Common Objects in Context (2017 release). Conjunto de dados. [S.l: s.n.]. Disponível em: <<https://cocodataset.org/>>. Acesso em: 21 fev. 2026., 2017

LIN, Tsung-Yi et al. Microsoft COCO: Common Objects in Context. *Lecture Notes in Computer Science*, 2014, Cham: Springer International Publishing, 2014. p.740–755. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-319-10602-1_48>. 978-3-319-10602-1.

LUTTE, Isabelle et al. Impact of the biopsychosocial model of disability on the medicolegal assessment of personal injury. *Archives of Public Health* v. 82, n. 1, p. 248, 31 dez. 2024. Disponível em: <<https://archpublichealth.biomedcentral.com/articles/10.1186/s13690-024-01471-6>>. Acesso em: 10 ago. 2025.

MACUÁCUA, Jaime Carlos et al. Automatic detection of urban infrastructure elements from terrestrial images using deep learning. *Boletim de Ciências Geodésicas* v. 30, p. e2024011, 2024. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1982-21702024000100204&tlng=en>. Acesso em: 5 jan. 2026.

MARCONI, Marina de Andrade; LAKATOS, Eva Maria. *Fundamentos de metodologia científica*. 8. ed. São Paulo: Atlas, 2017.

MARTÍNEZ-CHAO, Tomás E. et al. Urban Pedestrian Routes' Accessibility Assessment Using Geographic Information System Processing and Deep Learning-Based Object Detection. *Sensors* v. 24, n. 11, p. 3667, 5 jun. 2024. Disponível em: <<https://www.mdpi.com/1424-8220/24/11/3667>>. Acesso em: 5 jan. 2026.

MICROSOFT CORPORATION. Bing Image Search. Serviço web. [S.l.: s.n.]. Disponível em: <<https://www.bing.com/images>>. Acesso em: 24 fev. 2026., 2026

NAVAS-BONILLA, Carmen Del Rosario et al. Inclusive education through technology: a systematic review of types, tools and characteristics. *Frontiers in Education* v. 10, p. 1527851, 12 fev. 2025. Disponível em: <<https://www.frontiersin.org/articles/10.3389/feduc.2025.1527851/full>>. Acesso em: 10 ago. 2025.

ORTOBRAS. Cadeira de Banho H1. Barão, RS: Ortobras. Disponível em: <<https://loja.ortobras.com.br/cadeira-de-banho-h1>>., 2026f

ORTOBRAS. Cadeira de Rodas Anfíbia Ipanema. Barão, RS: Ortobras. Disponível em: <<https://loja.ortobras.com.br/ipanema>>., 2026g

ORTOBRAS. Cadeira de Rodas Conform Tilt Reclinável. Barão, RS: Ortobras. Disponível em: <<https://loja.ortobras.com.br/cadeira-de-rodas-conforma-tilt-reclinavel>>., 2026d

ORTOBRAS. Cadeira de Rodas Infantil Genesys. Barão, RS: Ortobras. Disponível em: <<https://loja.ortobras.com.br/cadeira-de-rodas-infantil-genesys>>., 2026e

ORTOBRAS. Cadeira de Rodas KE. Barão, RS: Ortobras. Disponível em: <<https://loja.ortobras.com.br/cadeira-de-rodas-ke>>., 2026a

ORTOBRAS. Cadeira de Rodas Motorizada E4. Barão, RS: Ortobras. Disponível em: <<https://loja.ortobras.com.br/cadeira-de-rodas-motorizada-e4>>., 2026c

ORTOBRAS. Cadeira de Rodas Star Lite. Barão, RS: Ortobras. Disponível em: <<https://loja.ortobras.com.br/cadeira-de-rodas-star-lite>>., 2026b

ORTOBRAS INDÚSTRIA E COMÉRCIO DE ORTOPEDIA. Catálogo de produtos Ortobras. [S.l.: s.n.]. Disponível em: <<https://loja.ortobras.com.br/cadeiras-de-rodas>>. Acesso em: 21 fev. 2026., 2026

ORTOPEDIA JAGUARIBE. Cadeira de Rodas Agile Hospitalar. Mauá, SP: Ortopedia Jaguaribe. Disponível em: <<https://ortopediajaguaribe.com.br/vendas/cadeira-de-rodas-agile-hospitalar/>>., 2026b

ORTOPEDIA JAGUARIBE. Cadeira de Rodas Super Big Monobloco 250kg. Mauá, SP: Ortopedia Jaguaribe. Disponível em: <<https://ortopediajaguaribe.com.br/vendas/cadeira-de-rodas-super-big-monobloco-250kg/>>., 2026a

ORTOPEDIA JAGUARIBE INDÚSTRIA E COMÉRCIO. Catálogo de produtos Ortopedia Jaguaribe. [S.l.: s.n.]. Disponível em: <<https://ortopediajaguaribe.com.br/>>. Acesso em: 21 fev. 2026., 2026

PAIVA, Omir Antunes; MOREIRA, Renata De Oliveira. Raspberry Pi: a 35-dollar device for viewing DICOM images. Radiologia Brasileira v. 47, n. 2, p. 99–100, abr. 2014. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0100-39842014000200012&lng=en&tlng=en>. Acesso em: 3 out. 2025.

PINNOCK BRANFORD, Kathylee et al. An inertial sensor-based comprehensive analysis of manual wheelchair user mobility during daily life in people with SCI. PLOS ONE v. 20, n. 9, p. 1–23, set. 2025. Disponível em: <<https://doi.org/10.1371/journal.pone.0323050>>.

POOJARY, Gayathri; S, Soundarya; N, Harshitha. ENVIRONMENTAL BARRIERS FACED BY WHEELCHAIR USERS IN DISABLED ORGANISATION OF BANGALORE- AN OBSERVATIONAL STUDY. Journal of Public Health and Community Medicine n. 0, p. 1, 2025. Disponível em: <<https://www.ejmanager.com/fulltextpdf.php?mno=302657717>>. Acesso em: 9 ago. 2025.

RASPBERRY PI FOUNDATION. Raspberry Pi 5 product brief. Cambridge: Raspberry Pi Ltd, 2025. Disponível em: <<https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf>>. Acesso em: 9 maio 2026.

REDMON, Joseph et al. You Only Look Once: Unified, Real-Time Object Detection arXiv:1506.02640 [cs]. [S.l.]: arXiv. Disponível em: <<http://arxiv.org/abs/1506.02640>>. Acesso em: 11 ago. 2025., 9 maio 2016

SAKELLARIOU, Dikaio; ROTAROU, Elena S. Access to healthcare for men and women with disabilities in the UK: secondary analysis of cross-sectional data. BMJ Open v. 7, n. 8, p. e016614, ago. 2017. Disponível em: <<https://bmjopen.bmj.com/lookup/doi/10.1136/bmjopen-2017-016614>>. Acesso em: 9 ago. 2025.

SARKER, Md Abdul Baset et al. Autonomous Movement of Wheelchair by Cameras and YOLOv7. In: ASEC 2022, 9 fev. 2022, [S.l.]: MDPI, 9 fev. 2022. p.60. Disponível em: <<https://www.mdpi.com/2673-4591/31/1/60>>. Acesso em: 11 ago. 2025.

SIMPSON, Richard C. Smart wheelchairs: A literature review. *The Journal of Rehabilitation Research and Development* v. 42, n. 4, p. 423, 2005. Disponível em: <<https://www.research.va.gov/isrm/rrdt/jrrd.cfm>>. Acesso em: 8 set. 2025.

SOFTWARE TESTING HELP. Data Mining Vs Machine Learning Vs Artificial Intelligence .Software Testing Help. [S.l.: s.n.]. Disponível em: <<https://www.softwaretestinghelp.com/data-mining-vs-machine-learning-vs-ai/>>. Acesso em: 4 jan. 2026., 2024

SUM, Yin Chuang. Wheel Chair Dataset. Conjunto de dados. [S.l.]: Kaggle. Disponível em: <<https://www.kaggle.com/datasets/yinchuangsum/person-wheel-chair-not-wheel-chair>>. Acesso em: 21 fev. 2026., 2021

SUNRISE MEDICAL. Catálogo de produtos Sunrise Medical. [S.l.: s.n.]. Disponível em: <<https://www.sunrisemedical.com/>>. Acesso em: 21 fev. 2026., 2026

SUNRISE MEDICAL. QUICKIE Q700-UP M: Standing Power Wheelchairs. Fresno, CA: Sunrise Medical. Disponível em: <<https://www.sunrisemedical.com/power-wheelchairs/quickie/standing-power-wheelchairs/q700-up-m>>., 2026b

SUNRISE MEDICAL. Sports Wheelchairs: RGK. Fresno, CA: Sunrise Medical. Disponível em: <<https://www.sunrisemedical.com/manual-wheelchairs/rgk/sports-wheelchairs>>., 2026a

TZUTALIN. Labellmg: graphical image annotation tool. GitHub repository. [S.l.: s.n.]. Disponível em: <<https://github.com/tzutalin/labellmg>>., 2015

VAKALOPOULOU, Maria et al. Deep Learning: Basics and Convolutional Neural Networks (CNNs). In: COLLIOT, Olivier (Org.). *Machine Learning for Brain Disorders. Neuromethods*. New York, NY: Springer US, 2023. 197 v. p. 77–115. Disponível em: <https://link.springer.com/10.1007/978-1-0716-3195-9_3>. Acesso em: 27 out. 2025. 978-1-0716-3194-2.

VASQUEZ, Andres; KOLLMITZ, Marina; EITEL, Andreas; BURGARD, Wolfram. Deep detection of people and their mobility aids for a hospital robot, MobilityAids Dataset. [Conjunto de dados]. Freiburg: University of Freiburg, 2017. Disponível em: <<http://mobility-aids.informatik.uni-freiburg.de/>>. Acesso em: 21 fev. 2026.

YUMANG, Analyn N.; BANGUILAN, Dave Emilson S.; VENERACION, Clark Kent S. Raspberry PI based Food Recognition for Visually Impaired using YOLO Algorithm. In: 2021 5TH INTERNATIONAL CONFERENCE ON COMMUNICATION AND INFORMATION SYSTEMS (ICCIS), 15 out. 2021, Chongqing, China: IEEE, 15 out. 2021. p.165–169. Disponível em: <<https://ieeexplore.ieee.org/document/9645981/>>. Acesso em: 3 out. 2025. 978-1-6654-2216-1.

ZAKS, Zosia. Changing the medical model of disability to the normalization model of disability: clarifying the past to create a new future direction. *Disability & Society* v. 39, n. 12, p. 3233–3260, 2 dez. 2024. Disponível em: <<https://www.tandfonline.com/doi/full/10.1080/09687599.2023.2255926>>. Acesso em: 4 jan. 2026.

APÊNDICE A - DETALHAMENTO TÉCNICO E REPRODUTIBILIDADE

Este apêndice reúne os detalhes de implementação de natureza estritamente técnica que, em nome da fluidez do texto principal, foram dele transferidos, preservando-se no corpo da dissertação as decisões metodológicas e seus fundamentos. Seu objetivo é assegurar a reprodutibilidade integral dos procedimentos descritos no Capítulo 3.

A.1 SCRIPTS DE CONSTITUIÇÃO E ORGANIZAÇÃO DO *CORPUS*

As etapas de filtragem, deduplicação, renomeação e particionamento do corpus, descritas na Seção 3.3, foram operacionalizadas por um conjunto de scripts em Python, entre os quais: `filtrar_mobility_v2.py` (varredura e amostragem temporal da base MobilityAids); `filtrar_resolucao.py` (verificação da resolução mínima de 640×480 pixels com a biblioteca Pillow); `renomear_bing.py` e `achatar_pastas.py` (resolução do conflito de nomenclatura entre as duas rodadas do Bing, com adição de prefixos identificadores da fonte e da tipologia, no padrão `bing_esportiva_000001.jpg`); e `renomear_sequencial.py` (renomeação final das 2.000 imagens, de 0000 a 1999). Um script de validação complementar verifica a correspondência entre cada imagem e seu arquivo de anotação, o intervalo válido dos índices de classe (0 a 10) e a normalização das coordenadas das caixas delimitadoras no intervalo $[0, 1]$.

A.2 ESTRUTURA DE DIRETÓRIOS DO CONJUNTO DE DADOS

A estrutura de diretórios do conjunto de dados, compatível com os formatos esperados pelos frameworks YOLOv5 e YOLOv8, organiza-se conforme a hierarquia a seguir:

```

datasets/
├── data.yaml
├── images/           ← subdivisão por tipo (formato YOLOv5)
│   ├── train/
│   ├── val/
│   └── test/
└── labels/          ← subdivisão por tipo (formato YOLOv5)
```

```

|   |— train/
|   |— val/
|   └— test/
|— train/      ← subdivisão por subset (formato YOLOv8)
|   |— images/
|   └— labels/
|— val/        ← subdivisão por subset (formato YOLOv8)
|   |— images/
|   └— labels/
|— test/       ← subdivisão por subset (formato YOLOv8)
|   |— images/
|   └— labels/
|— raw/        ← preservação das imagens originais
|   |— todas/
|   |   └— classes.txt
|   |— banco_completo/
|   |— Annotations_RGB/
|   └— todas_retreino/
└— split_report.csv

```

A.3 CORREÇÃO DO LABELIMG PARA COMPATIBILIDADE COM O PYTHON

Durante a fase preparatória da pesquisa, o uso do *LabelImg* na estação de trabalho exigiu uma correção pontual no código-fonte da própria ferramenta. O problema surgiu da combinação de duas circunstâncias: o *LabelImg* utiliza a biblioteca gráfica *PyQt5*, que representa as coordenadas do cursor do mouse por meio de objetos do tipo *QPointF* (uma estrutura que guarda os valores em ponto flutuante, com casas decimais); a partir da versão 3.10 do *Python*, no entanto, alguns métodos passaram a exigir explicitamente argumentos inteiros, e a ausência dessa conversão fazia o programa interromper a execução com erro sempre que o pesquisador tentava redimensionar uma caixa delimitadora pelo mouse.

A solução adotada foi um *script Python* denominado *fix_venv.py*, que insere a conversão para inteiro em quatro pontos específicos do código-fonte do *LabelImg*, nas chamadas aos métodos *.x()*, *.y()*, *.width()* e *.height()*. Apesar de não constituir uma

correção oficial da ferramenta, essa adaptação permitiu utilizar o *Labellmg* sem regredir a versão do *Python*, e é registrada aqui em nome da transparência metodológica e para que outros pesquisadores que enfrentem a mesma incompatibilidade possam reproduzir a solução.

A.4 COMANDOS DE EXPORTAÇÃO DOS MODELOS PARA A ARQUITETURA ARM

Os comandos a seguir reproduzem as exportações descritas na Seção 3.7.2, em complemento ao Quadro 10:

a) Modelos YOLOv8 (biblioteca *Ultralytics*), *TFLite FP32*: `model.export(format='tflite', imgsz=640)`.

b) Modelos YOLOv8, *TFLite INT8*: `model.export(format='tflite', imgsz=640, int8=True)`.

c) Modelos YOLOv5 (repositório oficial), *TFLite*: `python export.py --weights best.pt --include tflite --img 640`, acrescentando-se o argumento `--int8` para a variante *INT8*.

d) Modelo YOLOv8n, *NCNN FP16*: `model.export(format='ncnn', imgsz=640, half=True)`.

A.5 CONFIGURAÇÃO DO AMBIENTE EMBARCADO

Esta seção reúne, na íntegra, os procedimentos de preparação da plataforma embarcada sintetizados na Seção 3.7.1, transferidos para cá em nome da fluidez do texto principal e sem prejuízo da reprodutibilidade.

A configuração inicial do dispositivo seguiu um fluxo padronizado que assegurou a reprodutibilidade integral do procedimento. O cartão *microSD* foi preparado em computador externo (*Windows*) por meio do *Raspberry Pi Imager*, ferramenta oficial da *Raspberry Pi Foundation*, com gravação do *Raspberry Pi OS Bookworm* em sua distribuição de 64 bits. Durante a gravação, foram pré-configurados o nome de usuário, a senha de acesso, a rede *wi-fi* para conexão automática e a habilitação do protocolo *SSH* (*Secure Shell*) que é um protocolo de acesso remoto criptografado, o que eliminou a necessidade de monitor, teclado ou mouse conectados ao *Pi* durante a primeira inicialização. A partir do primeiro *boot*, o sistema foi acessado remotamente da estação de trabalho, identificando-se o endereço IP por meio da

tabela *ARP* da rede local (*arp -a*), e a totalidade dos procedimentos subsequentes foi conduzida via linha de comando. Como primeiro passo, o sistema operacional foi atualizado com o comando *sudo apt update && sudo apt upgrade -y*, garantindo que todas as bibliotecas do sistema estivessem em suas versões mais recentes antes da instalação de qualquer dependência específica do projeto.

A fim de garantir o isolamento das bibliotecas *Python* utilizadas pelo sistema assistivo em relação às bibliotecas globais do *Raspberry Pi OS* (prática que evita conflitos de versão e facilita a manutenção), foi criado um ambiente virtual dedicado denominado *yolo_env*, mediante o comando *python3 -m venv ~/yolo_env*. Todas as instalações subsequentes ocorreram com esse ambiente ativado. As dependências foram instaladas em duas etapas. Primeiro, o *PyTorch* foi obtido a partir do índice oficial dedicado a sistemas sem aceleração por *GPU* (<https://download.pytorch.org/whl/cpu>), evitando a inclusão automática dos pacotes *CUDA*, que se destinam a *GPUs NVIDIA* e são incompatíveis com a arquitetura *ARM* do *Pi*. Em seguida, instalou-se o *framework Ultralytics* com o argumento *--no-deps*, seguido pela instalação manual das bibliotecas auxiliares estritamente necessárias, entre elas *NumPy*, *OpenCV* em sua variante *headless* (otimizada para execução sem interface gráfica, adequada ao acesso via *SSH*), *Pillow* e *PyYAML*. Adicionalmente, foram instalados o *TensorFlow 2.21.0* com o *runtime ai-edge-litert 2.1.3*, para suportar a execução de modelos no formato *TFLite*, e o *NCNN 1.0*, *framework* de inferência da *Tencent* otimizado para processadores *ARM* e capaz de explorar nativamente as instruções *NEON FP16* do *Cortex-A76*.

A.6 REGISTRO DE PERCALÇOS TÉCNICOS

Esta seção documenta, na íntegra, os três percalços técnicos resolvidos durante o desenvolvimento, condensados nas Seções 3.7.2, 3.7.3 e 3.9 e aqui preservados para fins de transparência metodológica e reprodutibilidade.

Percalço técnico documentado: exportação inicial em *FP32*

Durante a fase de validação operacional do sistema embarcado, observou-se uma discrepância significativa entre o desempenho de inferência registrado nos *benchmarks* sintéticos do *NCNN* (aproximadamente 14 *FPS*) e o desempenho real medido com a *webcam* ativa, que entregava apenas 8,5 *FPS*, uma queda de cerca de 40% sem justificativa aparente. A investigação diagnóstica seguiu protocolo

sistemático para descartar hipóteses concorrentes: (i) confirmou-se que o *governor* da *CPU* estava em modo *performance*, sem economia dinâmica de energia; (ii) verificou-se a manutenção do *overclock* a 2,8 GHz por inspeção do arquivo `/sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq`; (iii) inspecionou-se a temperatura do processador, estável em aproximadamente 51 °C, valor confortavelmente abaixo do limiar de ativação do *throttling* térmico; (iv) confirmou-se a saturação dos quatro núcleos por meio do *htop*, indicando que o gargalo era computacional e não decorrente de subutilização do *hardware*; e (v) executou-se um *benchmark* sintético comparativo entre as inferências *PyTorch* e *NCNN* no mesmo dispositivo, isolando a variável do *framework*.

A causa raiz foi identificada na inspeção do arquivo *metadata.yaml* gerado pelo procedimento de exportação *NCNN*: o campo *half* constava como *false*, e o tamanho do arquivo do modelo era de 12 MB, sinais inequívocos de que o modelo havia sido exportado em precisão plena (*FP32*), e não em meia precisão (*FP16*) como pretendido. A omissão do parâmetro explícito *half=True* no comando inicial fez com que a biblioteca adotasse silenciosamente o comportamento padrão de exportação em *FP32*, comportamento que pode variar entre versões do *framework* sem aviso explícito ao usuário. A reexportação com o parâmetro corretamente definido gerou um modelo de 5,9 MB e restaurou a latência média de inferência para aproximadamente 71 ms, equivalentes aos 14 *FPS* reportados nos *benchmarks* sintéticos.

Esse percalço é registrado aqui em nome da transparência metodológica e como alerta técnico a pesquisadores que venham a enfrentar situação análoga: a verificação explícita do parâmetro de precisão após qualquer exportação para formatos otimizados, seja por inspeção do *metadata.yaml*, seja pela verificação simples do tamanho do arquivo gerado em comparação com o esperado, constitui boa prática operacional. O comportamento padrão das bibliotecas pode variar entre versões e plataformas, e a economia de tempo proporcionada pela conferência imediata é substancialmente maior do que o tempo de diagnóstico de discrepâncias de desempenho erroneamente atribuídas a outras causas, como, no caso aqui descrito, ao próprio *hardware* embarcado.

Percalço técnico documentado: contenção entre *OpenMP* e o compositor gráfico

Durante a fase de validação com *display* ativo, observou-se um comportamento inicialmente contraintuitivo: a ativação da janela de visualização do *OpenCV*

(*cv2.imshow*) elevava a latência de inferência de aproximadamente 71 ms para 110 ms, reduzindo o *FPS* para cerca de 10. A investigação diagnóstica conduziu a um experimento fatorial com variação do parâmetro *OMP_NUM_THREADS* (que controla o número de *threads* utilizadas pelo *OpenMP*, biblioteca de paralelização explorada pelo *NCNN* para distribuir operações de convolução entre os núcleos do processador) em três níveis (2, 3 e 4) combinada com duas condições da janela de exibição (ativa e inativa). O experimento revelou um padrão consistente: a configuração com *OMP_NUM_THREADS*=2 manteve a latência em 72,4 ms mesmo com o *display* ativo, enquanto *OMP_NUM_THREADS*=4 elevou a latência a 95,6 ms na mesma condição.

A interpretação adotada para esse comportamento é que, ao restringir o *pool* de *threads* do *OpenMP* a apenas dois núcleos do processador, os dois núcleos remanescentes do *Cortex-A76* permanecem disponíveis para o compositor gráfico *Wayland/Qt* utilizado pelo *Raspberry Pi OS*, eliminando a contenção por preempção que ocorre quando todos os quatro núcleos disputam ciclos de *CPU* entre a inferência e a renderização. A solução consolidada como padrão nos três aplicativos desenvolvidos para o sistema (cuja arquitetura é descrita na Seção 3.8) consistiu na definição programática de três variáveis de ambiente antes de qualquer importação do *NCNN* ou de bibliotecas correlatas, ajustando todos os *pools* de paralelismo nativos para o valor 2: *OMP_NUM_THREADS*, *MKL_NUM_THREADS* e *OPENBLAS_NUM_THREADS*. Essa configuração restaurou a taxa sustentada de aproximadamente 14 *FPS* mesmo com a interface gráfica ativa, completando o ajuste fino do sistema.

Subtensão do powerbank USB-C (Seção 3.9)

A primeira hipótese de alimentação portátil avaliada foi a utilização de um *powerbank USB-C* com tecnologia *Power Delivery (PD)*, padrão amplamente difundido em dispositivos móveis modernos e disponível em diversas faixas de capacidade no mercado. Essa solução pareceria, à primeira vista, suficiente para suprir o *Raspberry Pi 5* sob carga de inferência, uma vez que o protocolo *USB-C PD* admite o fornecimento dos 27 W especificados como potência nominal do dispositivo. Os testes operacionais, contudo, revelaram um comportamento contrário ao esperado. Em duas configurações de cabo distintas, incluindo modelos certificados para 5 A de corrente sustentada, o sistema registrou ativação contínua da *flag 0x50000* do utilitário *vcgencmd*, indicador específico do *Raspberry Pi OS* para a condição de subtensão na fonte de alimentação. Em resposta a esse alerta, o *firmware* do dispositivo aplicou

redução automática da frequência da *CPU* de 2.800 MHz para 1.500 MHz, ou seja, exatamente metade da frequência operacional configurada na Seção 3.7.1, provocando degradação severa do desempenho de inferência: o *FPS* sustentado caiu de 14,4 para valores entre 4,1 e 4,2, incompatíveis com qualquer uso operacional do sistema. A temperatura do processador permaneceu em 48,8 °C durante o teste, confortavelmente abaixo do limiar de *throttling* térmico, confirmando que a limitação observada era exclusivamente elétrica, decorrente da insuficiência de corrente sustentada, e não térmica. Esse resultado descartou a alternativa do *powerbank USB-C* convencional para a aplicação proposta.

ANEXO I - Exemplos Visuais das Tipologias de Cadeiras de Rodas

Figura A 1 - Manual Dobrável



Fonte: Ortobras (2026a)

Figura A 2 - Manual Ativa (Monobloco/Rígida)



Fonte: Ortobras (2026b)

Figura A 3 - Motorizada (Elétrica)



Fonte: Ortobras (2026c)

Figura A 4 - Postural (*Tilt-in-Space*/Reclinável)



Fonte: Ortobras (2026d)

Figura A 5 - Pediátrica/Infantil



Fonte: Ortobras (2026e)

Figura A 6 - Bariátrica (*Heavy Duty*)



Fonte: Ortopedia Jaguaribe (2026a)

Figura A 7 - Esportiva



Fonte: Sunrise Medical (2026a)

Figura A 8 - Banho/Higiênica



Fonte: Ortobras (2026f)

Figura A 9 - Hospitalar/Transporte



Fonte: Ortopedia Jaguaribe (2026b)

Figura A 10 - *Standing* (Verticalização)



Fonte: Sunrise Medical (2026b)

Figura A 11 - Anfíbia/Praia



Fonte: Ortobras (2026g)