ESTUDO DE INTERFACE GESTUAL VIA ALEXA COMO TECNOLOGIA COMPLEMENTAR AO COMANDO DE VOZ PARA APLICAÇÃO EM DOMÓTICA

Natã Silva do Nascimento[1]
Stefano Romeu Zeplin[2]

Resumo

Este trabalho apresenta um estudo de viabilidade técnica para a integração de um sistema de visão computacional com dispositivos de automação residencial, propondo uma tecnologia complementar ou alternativa aos tradicionais comandos de voz em assistentes virtuais como a Alexa, possibilitando seu uso também como tecnologia assistiva. A metodologia envolve a captura de imagens em tempo real via webcam com a biblioteca OpenCV e a extração de características anatômicas da mão pelo framework MediaPipe Hands. Esses dados estruturam um conjunto de dados (dataset) utilizado para o treinamento e otimização de uma rede neural artificial com TensorFlow Lite para execução em tempo real. O sistema interpreta sequências de gestos e converte essas informações em comandos enviados à assistente virtual, permitindo o controle de dispositivos domóticos. Os resultados demonstram-se promissores, evidenciando a viabilidade da solução em ambientes inteligentes e ampliando significativamente a acessibilidade e a autonomia residencial para indivíduos com restrições físicas ou limitações fonoaudiológicas no uso do recurso da voz.

Palavras-chave: Visão computacional. Reconhecimento de gestos. Automação residencial. Assistentes virtuais. Tecnologia assistiva.

STUDY OF GESTURE INTERFACE VIA ALEXA AS A COMPLEMENTARY TECHNOLOGY TO VOICE COMMAND FOR HOME AUTOMATION APPLICATIONS

Abstract

This paper presents a technical feasibility study for the integration of a computer vision system with home automation devices, proposing a complementary or alternative technology to traditional voice commands in virtual assistants such as Alexa, enabling its use also as an assistive technology. The methodology involves real-time image capture via webcam using the OpenCV library and hand anatomical feature extraction through the MediaPipe Hands framework. These data structure a dataset used to train and optimize an artificial neural network with TensorFlow Lite for real-time execution. The system interprets gesture sequences and converts this information into commands sent to the virtual assistant, enabling the control of home automation devices. The results are promising, demonstrating the system's viability in smart environments and

1 Acadêmico do curso Bacharel em Engenharia Elétrica do Instituto Federal de Santa Catarina. nata.nascimento@aluno.ifsc.edu.br
2 Docente do curso Bacharel em Engenharia Elétrica do Instituto Federal de Santa Catarina. stefano@ifsc.edu.br

significantly expanding accessibility and home autonomy for individuals with physical restrictions or speech-language limitations in using voice resources.

Keywords: Computer vision. Gesture recognition. Home automation. Virtual assistants. Assistive technology.

1 INTRODUÇÃO

No cenário comercial contemporâneo a aplicação da visão computacional é vasta, abrangendo setores que vão desde o varejo até a medicina de alta precisão. (DS ACADEMY, 2025). Uma dessas aplicações é explorada no presente artigo no âmbito da domótica, que consiste em um conjunto de serviços proporcionados por sistemas tecnológicos integrados, visando atender às necessidades básicas de conforto, segurança, comunicação e gestão energética nas residências.

Muitos desses sistemas contam predominantemente com comandos de voz, os quais oferecem praticidade, agilidade e uma forma intuitiva de interação, permitindo que o usuário execute tarefas sem a necessidade de contato com dispositivos móveis, interruptores ou interfaces táteis.

Contudo, a interface vocal impõe limitações críticas em cenários específicos, como em ambientes com elevado ruído acústico, que compromete a taxa de acerto do reconhecimento de voz, ou em locais que exigem silêncio. Além disso, a barreira se torna ainda maior para usuários com dificuldades de fala ou limitações fonoaudiológicas.

Dessa forma, a solução proposta consiste em um estudo de viabilidade de uma ferramenta que se caracteriza como tecnologia assistiva, atuando de forma complementar aos métodos tradicionais. O sistema utiliza a captura de imagens em tempo real por meio de uma câmera e algoritmos de visão computacional para traduzir gestos manuais em instruções lógicas, permitindo o controle direto de dispositivos de domótica integrados à assistente virtual Alexa. Com isso, o estudo busca validar uma alternativa viável de acessibilidade, podendo promover a autonomia de usuários com limitações fonoaudiológicas no ecossistema de casas inteligentes.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 VISÃO COMPUTACIONAL

A visão computacional é um subcampo da inteligência artificial que equipa sistemas computacionais com a capacidade de processar, analisar e interpretar entradas visuais do mundo físico, como imagens e vídeos. Utilizando dispositivos de captura (câmeras e sensores), essa ciência emprega algoritmos matemáticos para extrair informações significativas de matrizes bidimensionais de pixels, permitindo que a máquina reconheça e manipule objetos, pessoas e eventos (GOODFELLOW; BENGIO; COURVILLE, 2016).

O fluxo operacional que viabiliza o funcionamento de um sistema de visão computacional é estruturado em etapas sequenciais, descritas a seguir:

2.1.1 AQUISIÇÃO DE IMAGEM

A aquisição de imagem constitui a etapa inicial onde dados visuais do mundo físico são convertidos por sensores ópticos em sinais elétricos e, posteriormente, em uma matriz bidimensional de pixels (VENTURA; HIGA, 2023). Nesse processo, o fluxo contínuo de vídeo é segmentado em quadros individuais (frames) lidos como estruturas de dados multidimensionais (BRADSKI; KAEHLER, 2016). Para viabilizar a interface com o hardware e a manipulação em tempo real desses dados brutos, utiliza-se a biblioteca de código aberto OpenCV (Open Source Computer Vision Library). Embora desenvolvida originalmente em C++, suas funções são amplamente integradas ao ambiente Python (VILLÁN, 2019), atuando de forma fundamental na preparação e decodificação das matrizes matemáticas que serão submetidas às etapas subsequentes de pré-processamento e extração de características.

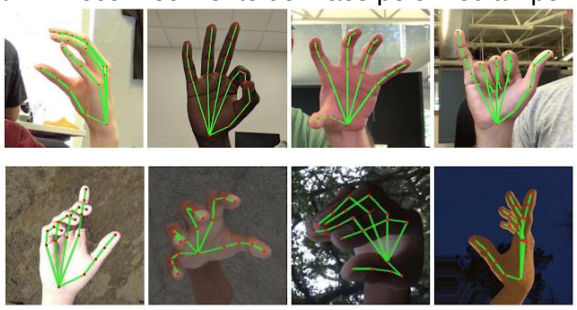
2.1.2 PRÉ-PROCESSAMENTO

O pré-processamento de imagens consiste na aplicação de técnicas que elevam a qualidade visual e padronizam os dados de entrada, otimizando a carga computacional e a precisão dos modelos de aprendizado de máquina (GONZALEZ; WOODS, 2024). Entre as operações essenciais destaca-se o redimensionamento *(resizing),* que estabelece dimensões constantes para assegurar uma estrutura de entrada previsível e otimizada ao processamento em tempo real (ULTRALYTICS, 2025). Adicionalmente, a conversão do espaço de cores RGB tradicional para a escala de cinza atua como uma simplificação clássica ao reduzir a informação de três canais distintos para um único canal. Essa transformação reduz drasticamente a complexidade matemática e o volume de memória utilizado, permitindo que o sistema foque na análise de formas, elementos cruciais para a detecção de estruturas anatômicas humanas (MOTA, 2020).

2.2 MEDIA PIPE

Desenvolvido pela Google, o MediaPipe Hands é um ecossistema de código aberto voltado à construção de pipelines de aprendizado de máquina para o processamento de mídias em tempo real (MEDIAPIPE, 2023), utilizado na etapa de extração de características. Sua arquitetura modular foi projetada para oferecer soluções de rastreamento de alto desempenho e baixa latência em plataformas heterogêneas, incluindo dispositivos móveis e computadores pessoais. No contexto da análise de membros corporais, o framework utiliza modelos de inteligências pré-treinados para segmentar e extrair marcos anatômicos (landmarks) tridimensionais, mapeando estruturas densas em vetores estruturados de coordenadas cartesianas, conforme exemplificado na Figura 1. Essa abordagem reduz drasticamente a dimensão dos dados de entrada, tornando os sistemas subsequentes menos sensíveis a interferências de iluminação, distância ou posicionamento, além de otimizar o processamento computacional.

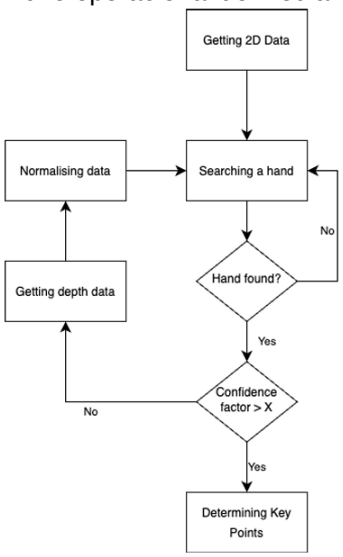
Figura 1 – Reconhecimento de mãos pelo MediaPipe Hands



Fonte: MEDIAPIPE

O fluxo operacional interno para a segmentação segue uma arquitetura lógica sequencial descrita na Figura 2. Inicialmente, o sistema captura os dados bidimensionais da imagem e inicia a busca pela região da mão. Caso uma mão seja localizada, o algoritmo avalia o fator de confiança da detecção contra um limiar crítico preestabelecido, se o índice for satisfatório, determinam-se os pontos-chave anatômicos. Caso contrário, o pipeline executa rotinas de varredura para recalcular as coordenadas e restabelecer o rastreamento da mão dentro do fator de confiança mínimo exigido.

Figura 2 – Fluxo operacional do MediaPipe Hands



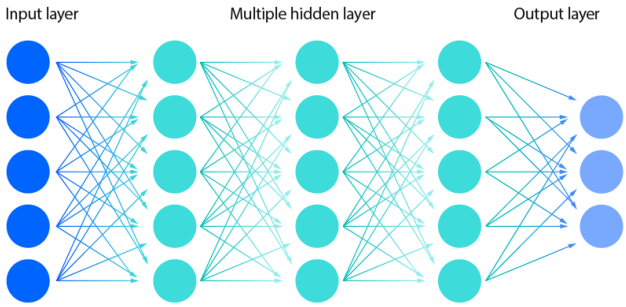
Fonte: Andrushchenko (2024)

2.3 REDES NEURAIS

As Redes Neurais Artificiais (RNAs) simulam estruturas biológicas para aprender padrões complexos por meio de camadas sucessivas de neurônios interconectados que ajustam seus parâmetros iterativamente (IBM, 2026). A arquitetura clássica Feedforward, também conhecida como Multilayer Perceptron (MLP), caracteriza-se pelo fluxo unidirecional da informação, a qual se propaga da camada de entrada para a saída. Nesse modelo, os dados entram na rede como vetores numéricos estruturados e cruzam camadas intermediárias (ocultas) onde cálculos matemáticos de pesos, vieses e funções de ativação extraem os recursos necessários para a classificação (LEE, 2026).

O modelo estrutural clássico de uma rede neural do tipo Multilayer Perceptron (MLP) é ilustrado na Figura 3. Conforme observado no diagrama, essa arquitetura é composta por um arranjo sequencial de camadas de neurônios artificiais interconectados, divididas em três níveis funcionais. A primeira delas é a camada de entrada (input layer), cuja função é receber os dados brutos do ambiente externo e distribuí-los para o sistema. Na sequência, encontram-se as camadas internas ou ocultas (hidden layers), que formam o núcleo de processamento abstrato da rede, nelas os neurônios aplicam transformações matemáticas para extrair características, identificar correlações e aprender os padrões intrínsecos dos dados. Por fim, o fluxo converge para a camada de saída (output layer), responsável por consolidar os resultados processados pelas camadas anteriores e apresentar a resposta final da rede na forma de classes de decisão ou valores preditivos.

Figura 3 – Rede Neural



Fonte: IBM e D. A. Team (2023).

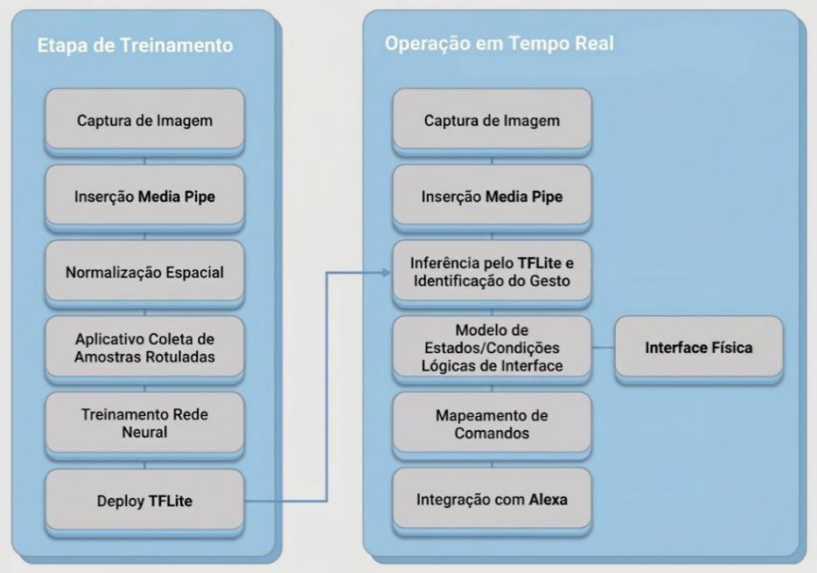
2.3 TENSORFLOW

O TensorFlow é uma plataforma de código aberto voltada à computação numérica e ao aprendizado de máquina (ABADI et al., 2016). A ferramenta viabiliza o ciclo completo de desenvolvimento de inteligência artificial desde a rotulagem de conjuntos de dados até o treinamento e validação de topologias de redes neurais profundas. Para cenários de computação periférica e dispositivos móveis, o ecossistema disponibiliza o TensorFlow Lite (TFLite), um conversor e interpretador especializado que otimiza e compacta os modelos originais. Esse processo reduz a latência de inferência e o consumo de memória, permitindo a execução de redes neurais robustas em tempo real com baixo consumo de CPU.

3 DESENVOLVIMENTO

Para facilitar a compreensão da arquitetura lógica e do processamento de dados do sistema proposto, a Figura 4 apresenta o fluxograma do projeto:

Figura 4 – Fluxograma do projeto



Fonte: O Autor (2026).

3.1 CAPTURA DE IMAGEM

O hardware de entrada utilizado para captura de dados foi o módulo de câmera integrado ao notebook, modelo KS0HD0Q003.

As especificações técnicas do módulo de captura são:

- Modelo: KS0HD0Q003 (OEM)
- Resolução Máxima: HD 720p (1280 x 720pixels)
- Taxa de Quadros: 30 FPS (*Frames per Second*)
- Sensor: CMOS

No script de captura desenvolvido em Python (versão 3.9.13 utilizada), os frames foram lidos utilizando os mecanismos da biblioteca *OpenCV*. Os métodos de tratamento de vídeo utilizado contam com três manipulações essenciais, sendo elas o redimensionamento do tamanho do quadro para um valor arbitrado, o espelhamento horizontal com o objetivo de tornar a execução dos gestos mais intuitiva no processo de desenvolvimento do sistema e a conversão do espaço de cores para o formato RGB. Esta última etapa é fundamental para garantir a compatibilidade dos dados de entrada com os requisitos técnicos do *framework* MediaPipe Hands, que opera nativamente neste formato de cor.

As variações de luminosidade, distância e oclusões propositais, que compõem os diferentes cenários de estresse do sistema, serão detalhadas e analisadas no capítulo referente aos Resultados e Testes.

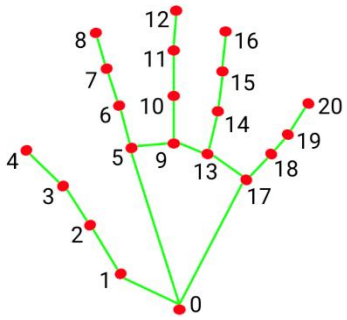
3.2 FRAMEWORK MEDIAPIPE HANDS

O início do processamento da visão computacional se dá pela implementação do framework *MediaPipe Hands* (versão 0.10.11 utilizada). O resultado é o rastreio e a identificação da presença de mãos no vídeo capturado disponibilizando as

coordenadas que correspondem às principais articulações e extremidades dos dedos normalizadas entre 0.0 e 1.0 em relação à largura e à altura do quadro, bem como à aproximação em relação à câmera.

Assim as coordenadas (x, y, z) de cada um dos 21 pontos (Figura 5) são concatenadas em um único vetor de 63 elementos (21x3).

Figura 5 – Marcos da mão reconhecidos pelo MediaPipe Hands



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

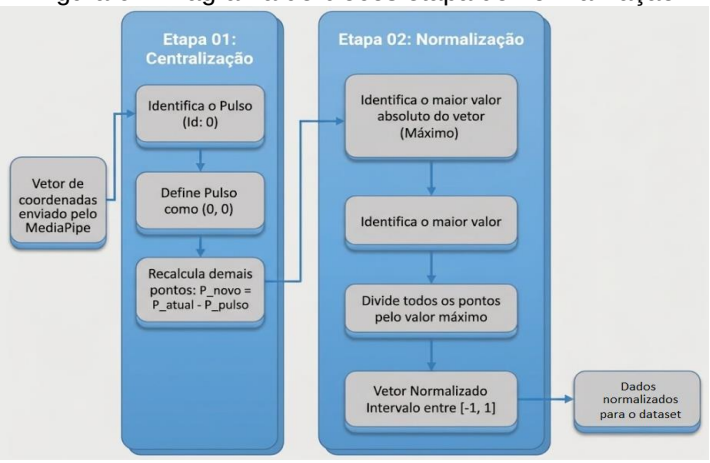
Fonte: GOOGLE (2025).

3.3 INVARIÂNCIA ESPACIAL

Para garantir que o reconhecimento dos gestos seja independente da posição da mão no campo de visão da câmera, aplicou-se um estágio de pré-processamento matemático sobre o vetor recebido do *framework Media Pipe Hands*.

Esta etapa é fundamental para que a rede neural aprenda a morfologia do gesto (a forma da mão) e não a sua localização geográfica na imagem detalhado no diagrama de blocos apresentado na Figura 6.

Figura 6 – Diagrama de blocos etapa de normalização



Fonte: O Autor (2026).

O processo de normalização ocorre em dois níveis principais:

1. **Translação para a Origem (Relatividade):** O sistema identifica o ponto correspondente ao pulso (*landmark* 0) e o define como o ponto de origem 0,0. Todas as outras coordenadas dos dedos são recalculadas subtraindo-se a posição do pulso. Dessa forma, os dados passam a representar a distância

relativa de cada articulação em relação à base da mão, tornando o sistema imune ao deslocamento do usuário diante da câmera.

2. **Normalização de Escala:** Para evitar que o tamanho da mão (ou a distância do usuário em relação à câmera) altere os valores de entrada, o sistema identifica o maior valor absoluto dentro do vetor de coordenadas e divide todos os outros pontos por esse máximo. Esse procedimento achata todos os valores para um intervalo entre -1 e 1 evitando que deslocamento em profundidade, ou seja, distanciamentos em relação a câmera gerem classificações inconsistentes devido à variação do tamanho aparente da mão.

3.4 CONSTRUÇÃO DO DATASET E ROTULAGEM DOS DADOS

Com a lógica de normalização estabelecida, a etapa subsequente consistiu na criação de um banco de dados estruturado em formato CSV, destinado ao treinamento do algoritmo de aprendizado de máquina que consiste em organizar os vetores resultantes em classes rotuladas, onde cada linha do arquivo representa uma amostra única de um gesto, composta por sua identificação numérica seguida pelas 63 coordenadas que definem sua configuração espacial.

Para a criação deste banco de dados foi criado um aplicativo onde a coleta de dados ocorre de maneira síncrona à interação do usuário com o aplicativo de coleta. Cada vez que a tecla arbitrada correspondente ao gesto é acionada, o sistema executa o seguinte fluxo:

- Consolida o rótulo da classe atual com o vetor de coordenadas recém-processado;
- Abre o arquivo CSV em modo de anexação (*append*);
- Grava a nova linha, garantindo que o banco de dados cresça de forma incremental a cada nova amostra coletada.

Esta abordagem permite uma coleta dinâmica, onde foi possível realizar o gesto em diferentes ângulos e posições, registrando centenas de amostras em poucos minutos. Além disso, permitiu o refinamento contínuo do sistema. Durante a fase de testes, gestos que apresentavam “falsos positivos” ou classificações imprecisas puderam ser corrigidos por meio de um incremento manual ou avulso através do aplicativo direto no banco de dados.

3.5 CARACTERÍSTICAS DE TREINAMENTO DO MODELO DE CLASSIFICAÇÃO

A implementação foi realizada em um ambiente de desenvolvimento baseado em Jupyter Notebook, utilizando o TensorFlow.

Para garantir a integridade estatística do modelo, aplicou-se a técnica de Hold-out:

- **Amostras de Treino (75%):** Utilizadas para o ajuste dos pesos e vieses (bias) dos neurônios.

- **Amostras de Teste (25%):** Reservadas exclusivamente para a validação final, garantindo que o modelo seja capaz de generalizar para dados nunca vistos.
- **Semente Aleatória (Seed 42):** Fixada para garantir a reprodutibilidade dos experimentos.

3.5.1 TOPOLOGIA DA REDE NEURAL

A rede neural usada possui um modelo **Sequencial** com a seguinte topologia:

- **Camada de Entrada:** Configurada para 42 neurônios (referentes ao vetor de coordenadas X e Y que incluem as posições dos 21 *landmarks* (a coordenada Z foi desprezada)).
- **Primeiro *Dropout* (20%):** Aplicado logo na entrada para mitigar a dependência excessiva de pontos específicos.
- **Primeira Camada Densa (20 neurônios):** Utiliza a função de ativação ReLU, responsável por extrair padrões não-lineares da posição da mão.
- **Segundo *Dropout* (40%):** Uma camada de regularização mais agressiva para evitar o *overfitting* do modelo.
- **Segunda Camada Densa (10 neurônios):** Camada intermediária de refinamento de características.
- **Camada de Saída (10 neurônios):** Correspondente às 10 classes de gestos definidas.

3.5.2 OTIMIZAÇÃO E DEPLOY COM TENSORFLOW LITE (TFLITE)

Após o estágio de treinamento e validação da rede neural, o modelo resultante originalmente salvo em formato .keras ou .h5 passa por um processo de conversão e otimização através do TFLite Converter. Esta etapa é crucial para a viabilidade do projeto, pois transforma a topologia densa da rede em um arquivo de extensão .tflite, reduzindo o tamanho do modelo e a latência de inferência sem perda significativa de acurácia.

No estágio de implantação (*deploy*), o interpretador do TensorFlow Lite é instanciado diretamente no script principal em Python. Essa integração permite que, a cada quadro capturado pela webcam, o vetor de 42 coordenadas pré-processadas seja injetado no modelo otimizado, que retorna instantaneamente a probabilidade do gesto correspondente, viabilizando uma interface de controle fluida e com baixo consumo de recursos de CPU.

3.6 GESTOS VÁLIDOS

O sistema foi projetado para reconhecer 10 gestos fundamentais, organizados em dois grupos de seleção que permitem o acesso a dispositivos específicos através de combinações:

- **Seletores de Grupo (A a C):** Definem o "setor" ou tipo de dispositivo a ser controlado.
- **Seletores de Unidade (1 a 4):** Identificam o dispositivo específico dentro do grupo selecionado.

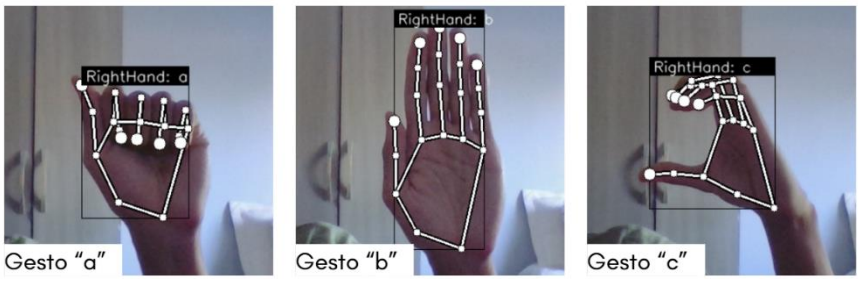
Esta abordagem matricial resulta em 12 combinações possíveis, permitindo que um número reduzido de gestos treinados controle uma vasta gama de aparelhos inteligentes, otimizando a escalabilidade do sistema de automação.

Uma vez selecionado o "endereço" do dispositivo, o sistema entra em modo de execução, onde interpreta três estados distintos:

- **Comandos Binários (Aberto/Fechado):** Utilizados para ações de estado on/off. O gesto de "Mão Aberta" é mapeado para o comando **Ligar**, enquanto o "Punho Fechado" executa o comando **Desligar**.
- **Controle Proporcional:** Foi implementado um gesto específico para ajustes graduais (como volume ou intensidade luminosa). Neste modo, o sistema calcula a variação da distância entre o polegar e o indicador, convertendo o movimento físico em um valor percentual (0% a 100%).

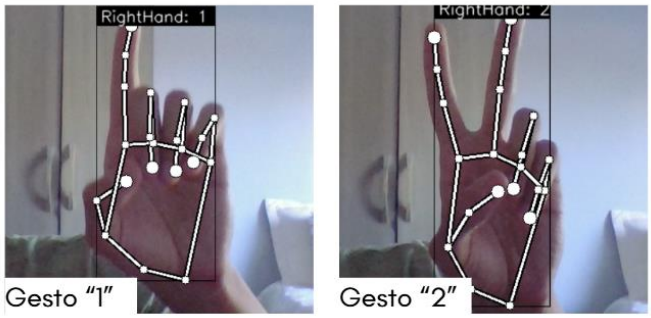
As identificações dos gestos reconhecidos pelo sistema em tempo real são apresentadas nas Figuras 7, 8, 9 e 10.

Figura 7 – Gestos reconhecidos



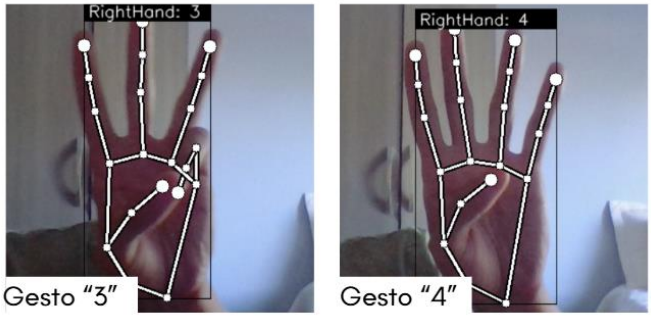
Fonte: O Autor (2026).

Figura 8 – Gestos reconhecidos



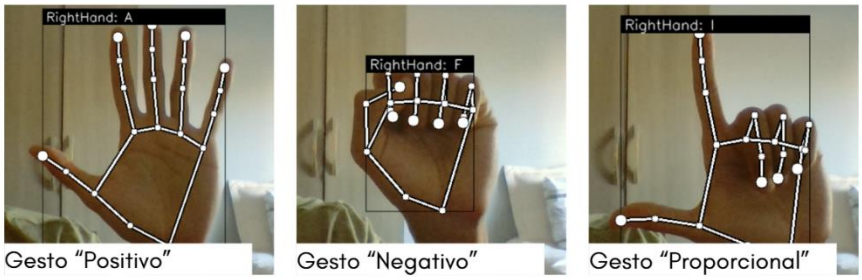
Fonte: O Autor (2026).

Figura 9 – Gestos reconhecidos



Fonte: O Autor (2026).

Figura 10 – Gestos reconhecidos



Fonte: O Autor (2026).

3.7 LÓGICA DE INTERAÇÃO E FLUXO DE DECISÃO (INTERFACE DO USUÁRIO)

Para garantir que o sistema não envie comandos acidentais à Alexa por movimentos involuntários, foi implementada uma lógica de interação baseada em gatilhos temporais e sequenciamento de estados. Este fluxo garante que a interface seja intencional e robusta.

O fluxo de interação segue quatro estágios fundamentais conforme ilustrado na Figura 11:

1. **Estado de Prontidão (Gatilho):** A interação é iniciada apenas quando o usuário mantém a mão na posição "Aberto" por pelo menos 1 segundo. Este

tempo de espera atua como um filtro de ruído, distinguindo um gesto de comando de um movimento casual.

2. **Identificação do Setor (Grupo A-C):** Uma vez ativado, o sistema aguarda a primeira parcela do código. O usuário deve realizar um dos gestos mapeados para as letras (A, B ou C), definindo a categoria do dispositivo a ser controlado.

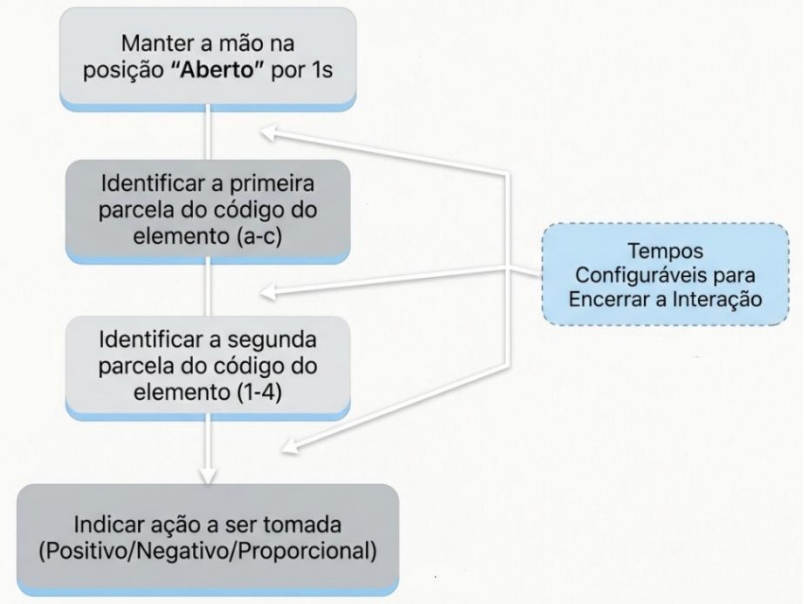
3. **Identificação da Unidade (1-4):** Com o grupo selecionado, o sistema busca a segunda parcela do código através dos gestos numéricos (1, 2, 3 ou 4). A combinação destas duas etapas gera o "endereço" único do dispositivo na rede de automação.

4. **Execução da Ação:** Finalmente, o sistema aguarda o gesto de comando final, que pode ser Positivo (Ligar/Abrir), Negativo (Desligar/Fechar) ou Proporcional (Ajuste de nível).

O sistema também conta com um mecanismo de tempos configuráveis durante a interação, que podem ser ajustados de acordo com a preferência do usuário. Essa mecânica atua como um timeout de segurança: caso o operador inicie uma sequência (como a seleção de um Grupo A-C), mas não realize o gesto subsequente dentro de um intervalo pré-definido, o sistema aborta automaticamente a operação e retorna ao estado de repouso.

Esta lógica de estados temporizados é fundamental para a resiliência do software, pois impede que o sistema permaneça "travado" em um estado de espera infinito por um comando que não será concluído. Além de otimizar o processamento, essa abordagem confere maior liberdade ao usuário, que pode desistir de uma ação em curso simplesmente baixando a mão, sem a necessidade de realizar um gesto específico de cancelamento.

Figura 11 – Fluxograma da lógica de interação

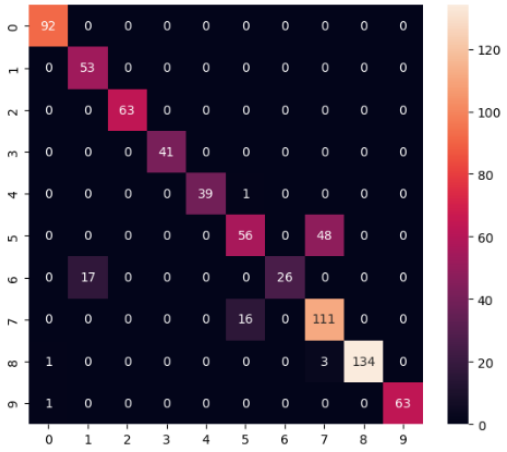


Fonte: O Autor (2026).

3.8 ANÁLISE DE DESEMPENHO DO MODELO DE CLASSIFICAÇÃO

Para analisar o desempenho do modelo de classificação utilizou-se uma matriz de confusão (Figura 12) e o relatório gerados a partir do conjunto de dados de teste:

Figura 12 – Matriz de confusão do conjunto de dados de testes



Fonte: O Autor (2026).

O modelo atingiu uma acurácia global de 89%, com médias dos indicadores *precision, recall e f1-score* atingindo 92%, 89% e 90%, respectivamente. Observa-se que o classificador obteve desempenho próximo da idealidade para as classes 0, 2, 3, 4, 8 e 9 (gestos aberto, 1, 2, 3, C e proporcional, respectivamente), cujos índices de *f1-score* pontuaram entre 0,99 e 1,00. Contudo, a matriz de confusão revela sobreposições no espaço de características de determinadas classes. O comportamento mais crítico ocorreu entre as classes 5 e 7 (gestos 4 e B, respectivamente), assim como entre as classes 6 e 1 (gestos Fechado e A, respectivamente). Esse fenômeno decorre da proximidade geométrica entre os gestos dessas respectivas classes.

3.9 PROTÓTIPO FÍSICO DE FEEDBACK DE OPERAÇÃO

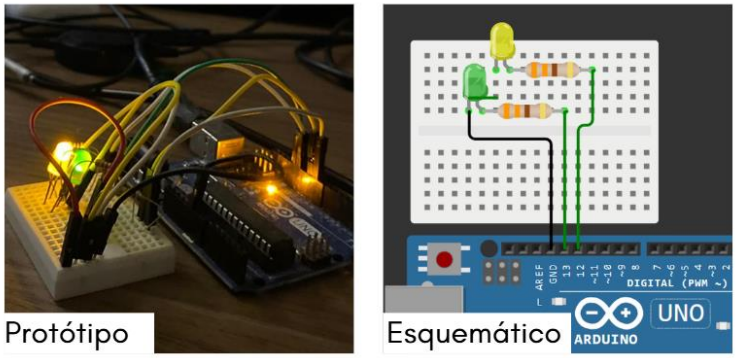
Foi desenvolvido um protótipo físico auxiliar baseado na plataforma Arduino. Este componente funciona como um módulo de feedback visual, fornecendo uma resposta sobre o estado lógico do sistema durante a operação.

- **Luz Verde (Detecção):** Este indicador ilumina-se assim que o sistema reconhece a presença de uma mão na área de captura da câmera.
- **Luz Amarela (Status do Comando):** Este é o indicador central de interação. Ele funciona em sincronia com o tempo de gatilho e as etapas do comando:
    - **Ativação:** Ao manter a mão aberta por 1 segundo, a luz amarela se acende, indicando que o sistema está em "modo de escuta" e aguardando a seleção do dispositivo.
    - **Feedback de Execução:** Após a conclusão bem-sucedida da sequência de gestos, a luz amarela pisca rapidamente por algumas vezes, servindo

como uma confirmação física de que o comando foi enviado para a nuvem e a Alexa o recebeu.

Este painel indicativo (Figura 13) permite que o usuário perceba claramente quando o sistema está ativo e quando uma ação foi concluída com sucesso. Além disso, o comportamento das luzes permite que o usuário visualize se o sistema retornou ao estado de repouso (desligando as luzes), garantindo que nenhum comando involuntário seja enviado posteriormente.

Figura 13 – Protótipo de Feedback de Operação



Fonte: O Autor (2026).

3.10 TABELA DE CONFIGURAÇÃO E MAPEAMENTO DE COMANDOS

A relação entre os gestos reconhecidos e as ações finais executadas pela Alexa não é fixa no código fonte. Em vez disso, o sistema utiliza um arquivo de configuração externo (formato .csv) que atua como um dicionário de mapeamento.

Esta abordagem permite adicionar novos dispositivos ou alterar ações sem a necessidade de modificar o código Python, facilitando a manutenção e a escalabilidade do projeto.

Desta forma, a lógica de controle interpreta a combinação dos gestos (ex: "A1", "B2") como um endereço único, consultando a tabela de configuração (Figura 14) para obter a ação correspondente. A ação é dividida em três categorias principais de ações baseadas no gesto final do usuário:

- **Ação Positiva:** Mapeada para gestos de confirmação (ex: "Ligar", "Ativar").
- **Ação Negativa:** Mapeada para gestos de interrupção (ex: "Desligar").
- **Ação Proporcional:** Mapeada para gestos de ajuste gradual (ex: "Brilho", "Volume").

Figura 14 – Configurador dos dispositivos

Device	Função do Gesto Positivo	Função do Gesto Negativo	Função do Gesto Proporcional
Alexa	Ligar	Desligar	Volume
Luz da Sala	Ligar	Desligar	Brilho
Luz do Quarto	Ligar	Desligar	Brilho
Luz da Cozinha	Ligar	Desligar	Brilho
Ventilador	Ligar	Desligar	Velocidade
Ar-Condicionado	Ligar	Desligar	Temperatura
Televisão	Ligar	Desligar	Volume
Cortina Inteligente	Ligar	Desligar	Abertura
Tomada Inteligente	Ligar	Desligar	Temporizador
Luminária da Mesa	Ligar	Desligar	Brilho
Home Theater	Ligar	Desligar	Volume
Cafeteira Inteligente	Ligar	Desligar	Intensidade

Fonte: O Autor (2026).

O conteúdo indicado na tabela de configuração é convertido em uma string de comando direto, representando a ação exata a ser executada. Por fim, esta string é enviada como uma instrução de texto para a Alexa, interagindo com o dispositivo inteligente mapeado.

3.11 INTEGRAÇÃO COM ALEXA

Para viabilizar a comunicação entre o sistema de visão computacional e os dispositivos inteligentes, foi utilizada uma arquitetura de controle via CLI (*Command Line Interface*). Esta abordagem permite que o software em Python interaja com a nuvem da Amazon como se fosse um usuário autenticado, enviando comandos de texto que são processados pela inteligência artificial da Alexa.

A continuidade da conexão é garantida através da biblioteca alexa-cookie-cli (ADN77, 2023). Este componente é responsável por realizar o login na conta Amazon do usuário e extrair o cookie de autenticação, que é armazenado localmente, evitando a necessidade de reautenticação manual a cada execução do programa.

Com a sessão autenticada, a emissão dos comandos é realizada através do script alexa-remote-control (GEHRIG, 2023). Após a rede neural identificar a combinação de gestos e localizar a ação correspondente no arquivo de configuração, o sistema dispara uma instrução via terminal utilizando o parâmetro de comando de texto (-e).

Este método permite uma flexibilidade superior às APIs convencionais, pois possibilita enviar qualquer frase que seria dita vocalmente à Alexa, como "ligar luz do quarto" ou "aumentar volume da TV", garantindo que a lógica do projeto seja compatível com qualquer dispositivo já configurado no ambiente doméstico do usuário.

3.12 RESULTADOS

Nesta seção, apresentam-se e discutem-se os resultados obtidos a partir dos testes de desempenho do sistema proposto. A avaliação foi estruturada com base na análise da precisão do algoritmo frente às baterias de ensaios (com ensaios divididos pelas distâncias de 1m, 2m e 3m entre o usuário e o sensor da câmera), totalizando cinco repetições para cada padrão gestual. Os dados coletados foram tabulados para permitir uma análise comparativa entre as taxas de acerto individuais e a performance global do sistema, buscando identificar a eficácia do sistema.

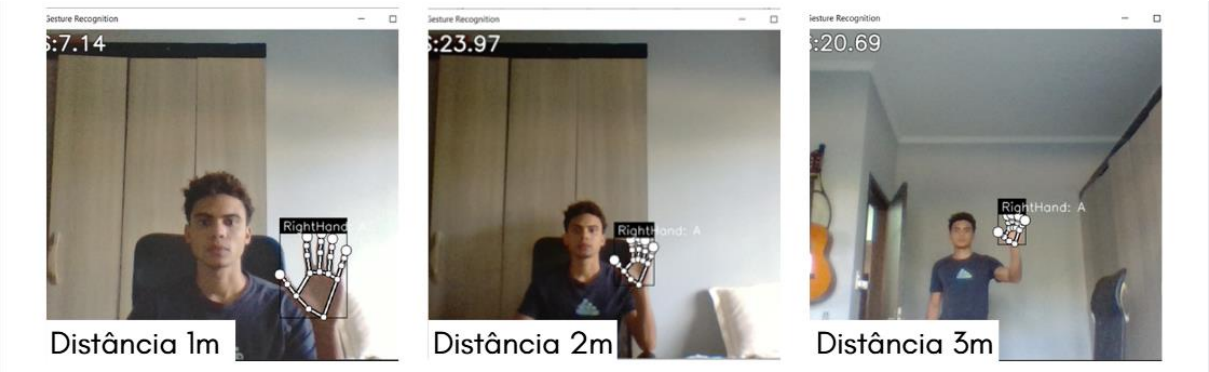
Utilizou-se o índice de precisão *(A)*, calculado pela razão entre o volume de predições corretas *(Vp)* e o total de tentativas realizadas *(Tt)* para cada padrão gestual. O resultado é expresso em base percentual conforme descrito pela equação:

$$A = \left(\frac{Vp}{Tt}\right) x100 \quad (1)$$

Este cálculo foi aplicado individualmente para cada gesto e, posteriormente, de forma agregada para compor a precisão da série e a precisão global de cada bateria de testes.

As imagens da Figura 15 apresentam o campo de visão do sistema e o rastreamento da mão do usuário variando entre as distâncias de teste.

Figura 15 – Captura da câmera para cada distância de teste



Fonte: O Autor (2026).

3.12.1 ANÁLISE E DISCUSSÃO DOS RESULTADOS

A análise dos dados da Tabela 1 revela que, enquanto o framework MediaPipe Hands foi capaz de identificar os marcos da mão (landmarks), a precisão do sistema manteve-se consistente nas distâncias de 1m, 2m e 3m. Esse comportamento é atribuído à etapa de normalização dos dados, que assegurou a invariância de escala, permitindo que o classificador obtivesse resultados similares independentemente da variação de profundidade, desde que a segmentação da mão fosse bem-sucedida.

Tabela 1 – Resultados de precisão para a distância de 1m, 2m e 3m

Gesto	Precisão (1m)	Precisão (2m)	Precisão (3m)
A1	100%	100%	80%
A2	100%	100%	100%
A3	100%	80%	80%
A4	80%	100%	100%
B1	80%	80%	100%
B2	80%	60%	80%
B3	100%	100%	100%
B4	100%	80%	100%
C1	100%	100%	100%
C2	80%	80%	80%
C3	100%	100%	100%
C4	100%	100%	80%
Precisão Global	**93%**	**90%**	**93%**

Fonte: O Autor (2026).

Observou-se, contudo, que os gestos que compõem a classe numérica “2” apresentaram maior tendência a erros, com a configuração “B2” sendo a mais problemática. Tecnicamente, isso decorre do fato de que as transições cinemáticas desse gesto geram vetores de características muito próximos aos padrões dos gestos “2” ou “4”, causando uma sobreposição no espaço de decisão do modelo.

É importante ressaltar que os ensaios foram conduzidos em ambiente com iluminação controlada e com o usuário devidamente alinhado ao eixo central do sensor.

A comunicação com a API da Alexa demonstrou-se fluida, apresentando latência perceptível apenas na interação inicial após longos períodos de inatividade, comportamento típico de *cold start* em requisições de API.

Por fim, identificou-se um limite operacional na distância de 3m. A partir deste ponto, conforme demonstrado na Figura 16, a resolução espacial da captura torna-se insuficiente para que o MediaPipe Hands identifique a mão com a confiabilidade necessária, inviabilizando a execução dos testes de precisão em distâncias maiores.

Figura 16 – Captura da câmera do sistema para a distância a partir de 3m

Fonte: O Autor (2026).

4 CONSIDERAÇÕES FINAIS

O presente trabalho demonstrou a viabilidade técnica do desenvolvimento de um sistema de controle para dispositivos de automação residencial baseado no reconhecimento de gestos manuais por visão computacional. A integração das tecnologias OpenCV, MediaPipe Hands, TensorFlow Lite e a assistente virtual Alexa resultou em uma solução funcional, capaz de operar em tempo real com baixo consumo de recursos computacionais.

Os resultados obtidos nas baterias de testes evidenciaram que o sistema apresentou uma precisão global satisfatória, demonstrando consistência diante das variações de profundidade. Esse desempenho foi possibilitado principalmente pela etapa de normalização dos dados, que garantiu a invariância de escala independente do posicionamento do usuário em relação à câmera. A comunicação com a API da Alexa se mostrou fluida, com latência perceptível apenas em situações de *cold start*, sem comprometer a experiência de uso.

Conclui-se que o sistema proposto representa uma alternativa concreta e acessível às interfaces de controle por voz tradicionais, ampliando as possibilidades de acessibilidade para usuários com limitações fonoaudiológicas ou em ambientes acusticamente desfavoráveis, alinhando-se ao objetivo central do trabalho de atuar como complemento aos comandos de voz em assistentes virtuais por meio de computação visual.

Como perspectivas para trabalhos futuros, destaca-se a possibilidade de transformar o sistema em um projeto *standalone*, com a migração da aplicação para plataformas embarcadas como o Raspberry Pi. Essa transição visa eliminar a dependência de um computador dedicado, ampliando a praticidade de implantação em ambientes residenciais.

Ademais, aponta-se como próxima etapa essencial a avaliação do sistema sob diferentes condições ambientais, validando o modelo em locais reais de teste (como salas, quartos e cozinhas com fundos complexos) e sob maiores variações de distâncias entre o usuário e o sensor.

Para superar as limitações de níveis de luminosidade, a incorporação de câmeras infravermelhas representa uma evolução relevante, habilitando a operação do sistema em ambientes noturnos ou de baixa iluminação. Adicionalmente, a expansão da biblioteca de gestos reconhecidos permitiria o mapeamento de um número maior de dispositivos e ações, aumentando a versatilidade do sistema.

Por fim, a otimização do pipeline de processamento e o aumento do banco de dados de treinamento, aliado à experimentação com diferentes topologias e configurações de arquiteturas de redes neurais, poderá elevar ainda mais a acurácia do algoritmo de classificação, tornando o sistema ainda mais robusto frente a variações de usuário e ângulo de captura.

Referências

ABADI, Martín et al. **TensorFlow: A system for large-scale machine learning.** Savannah, GA: USENIX Association, 2016. p. 265-283.

ADN77. **alexa-cookie-cli**. Versão 1.0.0. GitHub, 2023. Disponível em: https://github.com/adn77/alexa-cookie-cli. Acesso em: 10 mar. 2026.

ANDRUSHCHENKO, M. et al. **Hand movement disorders tracking by smartphone based on computer vision methods.** IAPGOŚ, v. 14, n. 2, p. 5-10, 2024. Disponível em: http://doi.org/10.35784/iapgos.6126. Acesso em: 8 jun. 2026.

BRADSKI, Gary; KAEHLER, Adrian. **Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library.** 2. ed. Sebastopol: O'Reilly Media, 2016.

DS ACADEMY. **Inteligência Artificial no Cenário Comercial Contemporâneo.** São Paulo: Data Science Academy, 2025.

GEHRIG, Alexander. **alexa-remote-control.** GitHub, 2023. Disponível em: https://github.com/alextech/alexa-remote-control. Acesso em: 10 mar. 2026.

GONZALEZ, Rafael C.; WOODS, Richard E. **Processamento Digital de Imagens.** 4. ed. Amsterdã: Blucher/Edgard Blücher, 2024.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning.** Cambridge: MIT Press, 2016.

GOOGLE. **MediaPipe Hands Landmark Model.** Google Developers, 2025. Disponível em: https://developers.google.com/mediapipe. Acesso em: 4 jun. 2026.

IBM; D. A. TEAM. **AI vs. machine learning vs. deep learning vs. neural networks: what's the difference?** [S. l.]: IBM, 2023. Disponível em: https://www.ibm.com. Acesso em: 7 jun. 2026.

LEE, James. ***Introduction to Neural Networks.*** Disponível em: https://www.ibm.com/think/topics/neural-networks. Acesso em: 10 mar. 2026.

MEDIAPIPE. **MediaPipe Solutions Guide.** Google Open Source, 2023. Disponível em: https://mediapipe.dev. Acesso em: 4 jun. 2026.

MOTA, Carlos S. **Processamento Digital de Imagens utilizando Python**. Rio de Janeiro: Ciência Moderna, 2020.

ULTRALYTICS. **Preprocessing Annotated CV Data**. Ultralytics YOLO Docs, 2025. Disponível em: https://docs.ultralytics.com/guides/preprocessing-annotated-data. Acesso em: 4 jun. 2026.

VENTURA, Rodrigo; HIGA, Roberto M. **Visão Computacional e Aquisição de Matrizes Digitais**. São Paulo: Érica, 2023.

VILLÁN, Alejandro Rodas. **Mastering OpenCV 4 with Python.** Birmingham: Packt Publishing, 2019.

NATÃ SILVA DO NASCIMENTO

ESTUDO DE INTERFACE GESTUAL VIA ALEXA COMO TECNOLOGIACOMPLEMENTAR AO COMANDO DE VOZ PARA APLICAÇÃO EM DOMÓTICA

Monografia apresentada ao curso de Engenharia Elétrica do Instituto Federal de Educação Ciência e Tecnologia de Santa Catarina, para obtenção do título de Bacharel em Engenharia Elétrica.

Joinville, 29 de junho de 2026.

Msc. Stefano Romeu Zeplin
Orientador
Instituto Federal de Educação Ciência e Tecnologia de Santa Catarina

Msc. Joice Luiz Jerônimo
Instituto Federal de Educação Ciência e Tecnologia de Santa Catarina

Dr. Rodrigo Coral
Instituto Federal de Educação Ciência e Tecnologia de Santa Catarina