PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RECONHECIMENTO FACIAL: ANÁLISE EXPERIMENTAL DE ROBUSTEZ E LIMITES DE CONFIABILIDADE DE SEGURANÇA

HIGOR SILVA SUDARIO

GOIÂNIA
2026

HIGOR SILVA SUDARIO

RECONHECIMENTO FACIAL: ANÁLISE EXPERIMENTAL DE ROBUSTEZ E LIMITES DE CONFIABILIDADE DE SEGURANÇA

Trabalho de Conclusão de Curso apresentado à Escola Politécnica, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.
Orientador(a): Prof: Rafael Leal Martins

GOIÂNIA
2026

HIGOR SILVA SUDARIO

RECONHECIMENTO FACIAL: ANÁLISE EXPERIMENTAL DE ROBUSTEZ E LIMITES DE CONFIABILIDADE DE SEGURANÇA

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola Politécnica e de Artes, Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Ciência da Computação, em ____/____/_____.

_____
Orientador(a): Prof. Me. Rafael Leal Martins

_____
Prof. Me. Fernando Gonçalves Abadia

_____
Prof. Dr. Ernesto Fonseca Veiga

GOIÂNIA
2026

RESUMO

A tecnologia de reconhecimento facial tornou-se onipresente em autenticação, vigilância e organização de mídias. Embora modelos de ponta, como FaceNet e ArcFace, alcancem excelente desempenho em bancos de dados bem controlados, seu uso em contextos reais ainda enfrenta duas fragilidades centrais: (i) sensibilidade às condições ambientais (iluminação, resolução, pose) e (ii) exposição a ataques de apresentação (fotos, vídeos, máscaras) e a mídias sintéticas (deepfakes). Este trabalho apresenta um estudo aplicado da robustez de um protótipo de reconhecimento facial sob três cenários de iluminação (ideal, baixa luz, e contraluz) e uma revisão técnica sobre limites de segurança e técnicas de detecção de ataques de apresentação (Presentation Attack Detection - PAD), também associadas à verificação de vivacidade. A avaliação empírica executada concentrou-se em acurácia rank-1 em regime fechado, taxa de detecção facial, falhas de detecção, erros de identidade e tempo de execução, comparando os pipelines dlib/face_recognition e ArcFace + RetinaFace. Também foi realizada uma avaliação exploratória de PAD ativo por desafio-resposta visual, com tentativas de fraude por foto e vídeo, sem reivindicar validação biométrica formal por APCER/BPCER. O estudo descreve como sistemas modernos podem detectar tentativas de fraude por meio de análise de textura, profundidade, sinais fisiológicos (rPPG), pistas fotoelétricas e inconsistências audiovisuais para deepfakes, indicando o estado da arte e seus limites.

Palavras-chave: reconhecimento facial; PAD; liveness detection; deepfake; biometria; LGPD.

ABSTRACT

Facial Recognition: Robustness Under Adverse Conditions and Security Reliability Limits. Facial recognition is widely used in authentication, surveillance, and media management. While state-of-the-art models (e.g., FaceNet, ArcFace) excel on controlled datasets, real-world deployments face two major challenges: sensitivity to environmental factors (illumination, resolution, pose) and exposure to presentation attacks (photos, videos, masks) and synthetic media (deepfakes). This work reports an applied study of robustness under three lighting scenarios (ideal, low light and backlight) and a technical review of security limits and presentation attack detection (PAD), commonly associated with liveness detection. The empirical evaluation performed in this study focused on closed-set rank-1 accuracy, face detection rate, detection failures, identity errors, and execution time, comparing the dlib/face_recognition and ArcFace + RetinaFace pipelines. An exploratory evaluation of active challenge-response PAD was also conducted with photo and video spoofing attempts, without claiming a formal biometric validation based on APCER/BPCER. The study describes how modern systems may detect fraud through texture analysis, depth sensing, physiological signals (rPPG), photometric cues, and audio-visual inconsistencies for deepfakes, outlining the state of the art and its limitations.

Keywords: facial recognition; PAD; liveness detection; deepfake; biometrics; LGPD.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

AAM-Softmax – Additive Angular Margin Softmax
ANPD – Autoridade Nacional de Proteção de Dados
API – Application Programming Interface
APCER – Attack Presentation Classification Error Rate
A/V – Audiovisual
AWS – Amazon Web Services
BPCER – Bona Fide Presentation Classification Error Rate
CNN – Convolutional Neural Network
CVPR – Conference on Computer Vision and Pattern Recognition
DET – Detection Error Tradeoff
DPIA – Data Protection Impact Assessment
EAR – Eye Aspect Ratio
EER – Equal Error Rate
FAR – False Acceptance Rate
FNMR – False Non-Match Rate
FRR – False Rejection Rate
FRTE – Face Recognition Technology Evaluation
HOG – Histogram of Oriented Gradients
HDR – High Dynamic Range
IA – Inteligência Artificial
IR – Infravermelho
ISO/IEC – International Organization for Standardization / International Electrotechnical Commission
LBPH – Local Binary Patterns Histograms
LGPD – Lei Geral de Proteção de Dados Pessoais
MAR – Mouth Aspect Ratio
MTCNN – Multi-task Cascaded Convolutional Networks
NIR – Near Infrared
NIST – National Institute of Standards and Technology
PAD – Presentation Attack Detection
ROC – Receiver Operating Characteristic
RGB – Red, Green and Blue
rPPG – Remote Photoplethysmography
SNR – Signal-to-Noise Ratio
SVM – Support Vector Machine
TCLE – Termo de Consentimento Livre e Esclarecido
ToF – Time of Flight
2D / 3D – Duas dimensões / três dimensões
1:1 / 1:N – Verificação um-para-um / identificação um-para-muitos

SUMÁRIO

1 INTRODUÇÃO

A identificação automática de pessoas por meio do reconhecimento facial tornou-se parte do cotidiano em smartphones, controle de acesso e videomonitoramento. O salto de desempenho nas últimas décadas decorre do uso de redes neurais profundas e perdas com margens angulares que produzem embeddings mais discriminativos, como FaceNet e ArcFace (SCHROFF et al., 2015; DENG et al., 2019). Em testes controlados, esses sistemas alcançam resultados muito elevados; porém, na prática, fatores como iluminação, pose, oclusões e resolução impõem variações relevantes de qualidade, exigindo que projetos reais descrevam com precisão onde e até que ponto o sistema se mantém confiável. Avaliações independentes reforçam que o desempenho em cenários reais deve ser documentado por métricas compatíveis com o protocolo utilizado, como acurácia rank-1 e taxa de detecção em avaliações de identificação, ou por métricas biométricas clássicas quando há protocolo formal de limiares e comparações genuínas/impostoras (NIST, 2023a; NIST, 2023b).

Além de desempenho, há o desafio da segurança: sistemas puramente visuais, sem mecanismos de detecção de ataques de apresentação (Presentation Attack Detection - PAD), também associados à verificação de vivacidade, podem ser enganados por fotos, vídeos em tela (replay), máscaras e mídias sintéticas (deepfakes). Práticas modernas combinam PAD passivo (pistas de textura, frequência e coerência temporal) com PAD ativo e/ou sensores de profundidade/IR, mitigando ataques comuns antes da etapa de reconhecimento (ISO/IEC 30107-3, 2023; NIST, 2023c; APPLE, 2024). Como a geração de deepfakes evolui rapidamente, a literatura recomenda defesas em camadas e reavaliação contínua de limiares e modelos (TOLOSANA et al., 2020; NGUYEN et al., 2022).

Há, ainda, aspectos legais e éticos. No Brasil, dados biométricos são classificados como dados pessoais sensíveis pela LGPD, o que impõe bases legais adequadas, minimização, segurança e prestação de contas (BRASIL, 2018; ANPD, 2024). Em paralelo, estudos apontam vieses demográficos e efeitos de ambiente que podem impactar a equidade de desempenho entre grupos, exigindo curadoria de dados, auditorias e documentação transparente de métricas por cenário (WU et al., 2022).

Diante desse contexto, este trabalho investiga, de forma aplicada, quão robusto é um sistema de reconhecimento facial sob três condições ambientais (ideal, baixa luz e contraluz) e discute seus limites de confiabilidade em segurança, com foco em PAD contra fotos, vídeos e máscaras, e na detecção de deepfakes. Apresenta-se um protocolo replicável de avaliação

baseado nas métricas efetivamente executadas no experimento: acurácia rank-1 em regime fechado, taxa de detecção facial, falhas de detecção, erros de identidade e tempo de execução. Também são discutidas diretrizes práticas de implementação (hardware 2D/3D, IR, iluminação mínima), bem como considerações legais e de governança. O objetivo é oferecer uma base técnica clara e transparente para adoção responsável da tecnologia em cenários reais (NIST, 2023a; NIST, 2023b; ISO/IEC 30107-3, 2023).

1.1 Justificativa

Aplicações reais de reconhecimento facial — controle de acesso, autenticação em dispositivos e vigilância — impõem requisitos simultâneos de acurácia e segurança sob condições ambientais variáveis. Estudos comparativos e avaliações independentes mostram que o desempenho que se observa em bases controladas pode degradar significativamente em cenários de baixa luz e contraluz, afetando especialmente a taxa de verdadeiros positivos em regimes de FPR muito baixos (NIST, 2023a; NIST, 2023b). Em paralelo, sistemas sem Detecção de Ataques de Apresentação (PAD) ficam suscetíveis a ataques de apresentação (foto impressa, replay em tela, máscara 3D) e a mídias sintéticas (deepfakes); por isso, normas e relatórios técnicos recomendam PAD passivo/ativo e/ou sensores de profundidade/IR como salvaguardas essenciais (ISO/IEC 30107-3, 2023; NIST, 2023c; APPLE, 2024; TOLOSANA et al., 2020; NGUYEN et al., 2022).

No Brasil, o tratamento de dados biométricos é regulado pela LGPD como dado pessoal sensível, exigindo minimização, base legal adequada, segurança e prestação de contas, além de atenção a possíveis vieses demográficos reportados na literatura (BRASIL, 2018; ANPD, 2024; WU et al., 2022). Diante desse cenário, justificar, medir e documentar a robustez em diferentes condições de iluminação/resolução e a eficácia das camadas de PAD é indispensável para implementar soluções confiáveis, seguras e aderentes às normas técnicas e legais.

1.2 Pergunta de Pesquisa

Quão robusto é um sistema de reconhecimento facial padrão sob condições adversas comuns e até onde ele consegue resistir a tentativas de fraude (fotos, vídeos, máscaras) e deepfakes?

1.3 Objetivos

1.3.1 Objetivo Geral

Avaliar a robustez de um sistema de reconhecimento facial sob condições adversas de iluminação e resolução, e discutir seus limites de confiabilidade quanto a ataques de apresentação e deepfakes.

1.3.2 Objetivos Específicos

- Implementar um protótipo funcional de reconhecimento facial com modelo pré-treinado.
- Definir e simular cenários de iluminação: ideal, baixa luz e contraluz.
- Medir, por cenário, a acurácia rank-1 em regime fechado, a taxa de detecção facial, as falhas de detecção, os erros de identidade e o tempo de execução dos pipelines avaliados.
- Implementar e avaliar um mecanismo de segurança ativo baseado em instruções aleatórias (desafio-resposta), em substituição a técnicas passivas de detecção de ataques de apresentação (PAD), para mitigar tentativas de fraude com fotos estáticas e vídeos pré-gravados, alinhando os testes às diretrizes da ISO/IEC 30107-3.
- Consolidar diretrizes de implementação considerando ambiente e segurança (2D vs 3D, IR, requisitos mínimos de luz, posicionamento da câmera).
- Contextualizar riscos de viés demográfico e conformidade com a LGPD.

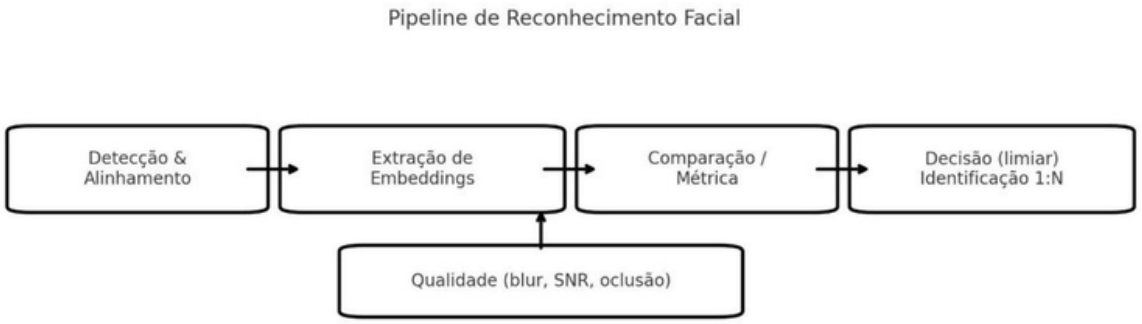
2 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta os conceitos, técnicas e normas que sustentam o trabalho, com foco no pipeline de reconhecimento facial, nos modelos de embeddings, em métricas de avaliação, nos fatores ambientais, na segurança (PAD e deepfakes), em vieses demográficos e nos aspectos legais/éticos.

2.1 Pipeline do Reconhecimento Facial

O pipeline de reconhecimento facial pode ser compreendido como uma sequência de etapas interdependentes, em que a qualidade de uma fase influencia diretamente o desempenho da etapa seguinte. Antes de comparar modelos ou métricas, é necessário entender como a imagem capturada é transformada em uma decisão de identidade. A Figura 1 sintetiza esse fluxo geral, desde a detecção e o alinhamento da face até a extração de embeddings, a comparação vetorial e a decisão final.

Figura 1 - Arquitetura do pipeline de reconhecimento facial.



Fonte: autoria própria (2026).

Na Figura 1, observa-se que a etapa de qualidade da imagem atua como um fator de controle sobre o pipeline, pois problemas como desfoque, baixo sinal-ruído e oclusões podem comprometer a extração dos embeddings e, consequentemente, a comparação final. Por isso, as etapas apresentadas a seguir devem ser lidas como uma cadeia única, e não como módulos totalmente independentes.

- Detecção e alinhamento. Um sistema moderno começa localizando o rosto (e.g., Haar/LBP, HOG, CNNs; hoje prevalecem detectores como MTCNN/RetinaFace) e alinhando a face com base em pontos-chave (olhos, nariz, boca) para reduzir variações de pose/escala (NIST, 2023a; NIST, 2023b).

- Extração de embeddings. Uma CNN profunda (ResNet/IR-SE) transforma a imagem alinhada em um vetor de características L2-normalizado; a separação entre vetores de pessoas distintas é ampliada por perdas com margem angular (DENG et al., 2019).

- Comparação e decisão. A verificação 1:1 usa distância cosseno/L2 e um limiar; a identificação 1:N busca o menor score em uma galeria. Monitorar qualidade (blur, SNR, oclusões) reduz erros a montante (NIST, 2023a; NIST, 2023b).

- Gestão de templates. Cadastro, atualização, revogação e auditoria de uso são práticas de governança requeridas em aplicações reais (BRASIL, 2018; ANPD, 2024).

2.2 Modelos e Embeddings (FaceNet, ArcFace e afins)

Modelos FaceNet introduziram a perda triplet, uma função de perda utilizada em aprendizagem métrica que compara três amostras ao mesmo tempo: uma imagem âncora, uma imagem positiva da mesma identidade e uma imagem negativa de outra identidade. O objetivo dessa função é aproximar, no espaço vetorial, imagens da mesma pessoa e afastar imagens de pessoas diferentes, maximizando a separação entre classes. A partir dessa lógica, o FaceNet mapeia imagens faciais para um espaço métrico discriminativo; já o ArcFace aprimora a separabilidade com margem angular aditiva na softmax, produzindo embeddings mais compactos e com maior margem interclasse (SCHROFF et al., 2015; DENG et al., 2019). Avaliações FRTE do NIST reportam contínua melhora de desempenho em verificação e identificação, inclusive sob variações de qualidade (NIST, 2023a; NIST, 2023b). Bases amplas como LFW (HUANG et al., 2007), IJB-B/IJB-C (NIST, 2025), VGGFace2 (CAO et al., 2018) e MegaFace (KEMELMACHER-SHLIZERMAN et al., 2016), além de conjuntos de vídeo como FIVE (NIST, 2024), servem para medir robustez em condições in the wild.

2.3 Métricas e Avaliação

A avaliação de sistemas de reconhecimento facial depende da tarefa executada e do protocolo experimental adotado. Métricas de verificação 1:1, identificação 1:N e detecção de ataques de apresentação medem aspectos diferentes do desempenho; por isso, os tópicos a seguir apresentam os principais indicadores usados como referencial teórico e esclarecem quais deles foram efetivamente aplicados neste TCC.

- Verificação (1:1) e identificação (1:N): tarefas distintas com métricas específicas (NIST, 2023a; NIST, 2023b).

- FAR/FRR e EER: o EER é o ponto em que FAR=FRR; operação segura costuma fixar FPR muito baixos (10^{-3} ou 10^{-4}) e reportar TPR@FPR (NIST, 2023a; NIST, 2023b).

- Curvas ROC/DET: exibem o compromisso entre sensibilidade e especificidade; recomenda-se relatar intervalos de confiança por bootstrap quando o protocolo e o tamanho da base permitem esse cálculo (NIST, 2023a; NIST, 2023b).

- CMC/Rank-k (1:N): mede a chance do indivíduo correto aparecer no topo do ranking ou entre os k primeiros resultados em tarefas de identificação (NIST, 2023a; NIST, 2023b).

- PAD (ISO/IEC 30107-3): usa APCER/BPCER para quantificar ataques aceitos e bona fides rejeitados; competições como LivDet-Face consolidam benchmarks (ISO/IEC 30107-3, 2023; IGENE et al., 2024).

2.4 Fatores Ambientais: Iluminação, Pose/Oclusões

A robustez de um sistema facial não é determinada apenas pelo modelo de reconhecimento, mas também pelas condições em que a imagem é capturada. Iluminação, pose, oclusões e qualidade geral da amostra interferem diretamente na detecção, no alinhamento e na extração de embeddings, tornando necessária a análise desses fatores antes da comparação entre pipelines.

- Iluminação. Em baixa luz, cresce o ruído e cai o SNR; em contraluz, a face fica subexposta, perdendo textura essencial aos embeddings. Por isso, recomenda-se iluminação frontal difusa e validação de qualidade antes da comparação biométrica (NIST, 2023a; NIST, 2023b; NIST, 2023c).

- Pose/oclusões. Ângulos extremos de yaw/pitch/roll e oclusões (máscaras, óculos escuros) prejudicam detecção/alinhamento; dados in the wild e alinhadores mais robustos mitigam parcialmente (NIST, 2023a; NIST, 2023b).

- Controle de qualidade. Métricas de blur/SNR e rejeição de amostras ruins reduzem FRR e falsos desconhecidos em cenários adversos (NIST, 2023a; NIST, 2023b).

2.4.1 Exemplos ilustrativos das condições de captura

As Figuras 2 a 4 apresentam exemplos ilustrativos, gerados com apoio de IA generativa, utilizados apenas para demonstrar visualmente como a qualidade da captura interfere no reconhecimento facial. Essas imagens não compuseram a base de testes, não foram processadas pelos scripts experimentais e não interferiram nos resultados quantitativos apresentados no Capítulo 4. Elas servem exclusivamente como apoio didático para representar os três tipos de cenário avaliados: ideal, baixa luz e contraluz. A Figura 2 representa uma condição ideal de aquisição, com o rosto frontal, bem iluminado e com textura visível, cenário em que a detecção e o reconhecimento tendem a ocorrer com maior estabilidade. A Figura 3 exemplifica uma situação de contraluz severa, na qual a face fica subexposta e perde detalhes importantes, o que costuma dificultar a detecção. Já a Figura 4 apresenta uma condição de ambiente escuro, em que o sistema pode falhar com frequência, embora em alguns casos ainda consiga detectar um rosto ou até reconhecer a pessoa, dependendo do nível residual de iluminação e da robustez do pipeline empregado.

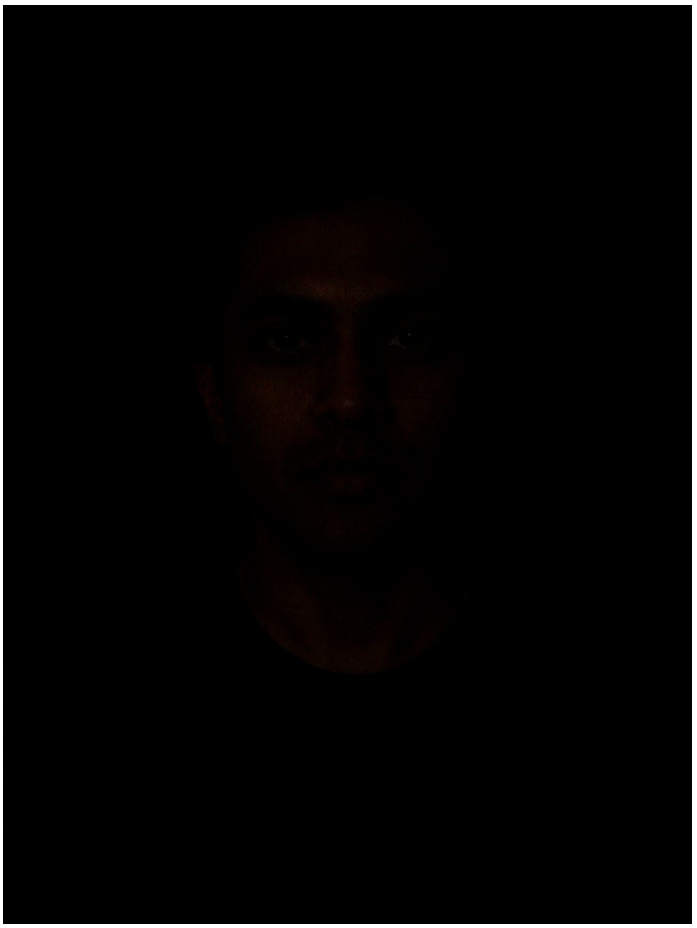
Figura 2 - Exemplo ilustrativo de imagem ideal, em que o rosto permanece frontal, bem iluminado e tipicamente reconhecível.

Fonte: elaboração própria com apoio de IA generativa (2026).

Figura 3 - Exemplo ilustrativo de imagem em contraluz severa, condição em que a subexposição facial tende a impedir a detecção e o reconhecimento.

Fonte: elaboração própria com apoio de IA generativa (2026).

Figura 4 - Exemplo ilustrativo de imagem em ambiente escuro, condição em que o reconhecimento pode falhar com frequência, embora às vezes ainda seja possível detectar o rosto ou identificar a pessoa.

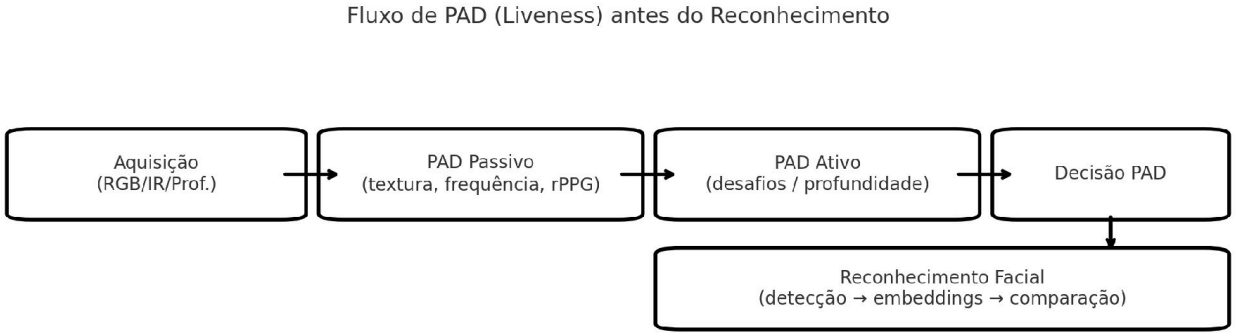


Fonte: elaboração própria com apoio de IA generativa (2026).

2.5 Segurança: PAD (Liveness) e Deepfakes

A segurança em reconhecimento facial envolve mecanismos capazes de distinguir uma pessoa fisicamente presente de uma apresentação fraudulenta, como foto impressa, vídeo em tela, máscara ou mídia sintética. Neste trabalho, a expressão Detecção de Ataques de Apresentação (PAD) é usada para designar essa camada de proteção, também conhecida em aplicações práticas como verificação de vivacidade ou liveness. A Figura 5 apresenta o fluxo conceitual adotado para organizar essa etapa de segurança no contexto do protótipo.

Figura 5 - Fluxo de PAD/liveness por desafio-resposta.



Fonte: autoria própria (2026).

Conforme ilustrado na Figura 5, o processo de segurança ocorre antes da decisão final de reconhecimento: a amostra capturada passa por uma verificação de vivacidade, que pode envolver estratégias passivas, ativas ou baseadas em sensores, e somente depois segue para a etapa de identificação ou comparação facial. Para evitar ambiguidade entre técnicas efetivamente implementadas e técnicas apenas discutidas na literatura, os tópicos seguintes separam as principais abordagens de PAD e o tratamento dado aos deepfakes neste TCC.

• PAD passivo (2D, software): na literatura, essa abordagem analisa textura da pele, pistas frequenciais, coerência temporal e, em alguns trabalhos, sinais fisiológicos estimados por imagem, como rPPG, para distinguir uma pessoa real de ataques com foto ou vídeo. Esses recursos são citados aqui como possibilidades técnicas existentes na área de Detecção de Ataques de Apresentação (PAD) e não correspondem, por si só, ao método implementado neste projeto (NIST, 2023c; SHARMA et al., 2023).

• PAD ativo e sensores: também existem abordagens ativas e multimodais, com desafios de movimento, sensores de profundidade, infravermelho e, em certos sistemas, recursos adicionais de áudio e coerência audiovisual. No entanto, este TCC não utiliza áudio, fala, microfone, sensores biomédicos, profundidade ou infravermelho; a implementação adotada baseia-se em câmera RGB convencional e desafio-resposta visual (ISO/IEC 30107-3, 2023; APPLE, 2024; MICROSOFT, 2026; AMAZON WEB SERVICES, 2026a; AMAZON WEB SERVICES, 2026b).

• Deepfakes: detectores podem buscar inconsistências fisiológicas, geométricas, espaciais e temporais, além de, em alguns cenários, coerência audiovisual. Contudo, essas possibilidades pertencem ao escopo geral da literatura de segurança biométrica e não significam que o presente trabalho utilize análise de voz, áudio ou sincronização labial. Neste

projeto, a ênfase recai sobre a mitigação de fraude com foto e vídeo por meio de PAD ativo visual (NIST, 2023c; IGENE et al., 2024; TOLOSANA et al., 2020; NGUYEN et al., 2022).

2.5.1 PAD Ativo por Desafio-Resposta (Challenge-Response)

Para mitigar ataques de apresentação que utilizam fotos impressas e vídeos reproduzidos em telas (replay attacks), sistemas de reconhecimento facial podem empregar técnicas de PAD ativo baseadas em comandos aleatórios, conhecidas como método de desafio-resposta (challenge-response). Diferente do PAD passivo, que analisa artefatos da imagem sem exigir colaboração explícita do usuário, o PAD ativo exige que o indivíduo comprove sua vivacidade e presença física no momento da captura (ISO/IEC 30107-3, 2023; ADAMIAK; ZUREK; SLOT, 2015; MICROSOFT, 2026; AMAZON WEB SERVICES, 2026b).

A segurança dessa tecnologia depende da imprevisibilidade dos comandos e da verificação temporal da resposta. O sistema solicita que o usuário execute uma sequência curta de ações aleatórias geradas em tempo real, tais como (ISO/IEC 30107-3, 2023; ADAMIAK; ZUREK; SLOT, 2015; MICROSOFT, 2026; AMAZON WEB SERVICES, 2026b):

- Piscar os olhos (eye blink).
- Movimentos faciais simples compatíveis com a câmera RGB, como sorrir ou alterar a expressão facial.
- Rotacionar a cabeça para a esquerda, direita, cima ou baixo (alteração dos ângulos de yaw e pitch).

Um ataque utilizando uma fotografia estática tende a falhar em instruções que exigem movimento facial ou alteração de pose. Por outro lado, um ataque de vídeo pré-gravado pode conter movimentos naturais, mas tende a não sincronizar as ações gravadas com a sequência e o tempo exato dos comandos aleatórios exigidos pelo sistema. Normas e diretrizes técnicas indicam que desafios ativos aleatórios podem atuar como camada adicional de defesa, especialmente em sistemas baseados em câmeras RGB tradicionais (2D) e sem hardware dedicado de profundidade ou infravermelho (ISO/IEC 30107-3, 2023; MICROSOFT, 2026; AMAZON WEB SERVICES, 2026b).

2.5.2 Como o sistema detecta vídeo em tela (replay)

Ataques por replay ocorrem quando uma imagem ou vídeo de uma pessoa autorizada é reproduzido em uma tela e apresentado à câmera como se fosse uma presença real. A

literatura descreve diferentes indícios que podem auxiliar na identificação desse tipo de fraude, embora nem todos tenham sido implementados neste protótipo.

- Na literatura, ataques de replay podem ser identificados por artefatos de tela, como padrões de subpixels, moiré, banding, cintilação e distorções temporais causadas pela diferença entre a taxa de atualização da tela e a captura da câmera. Esses sinais ajudam a distinguir uma apresentação em monitor ou celular de uma presença física real, embora a eficácia dependa do sensor, da iluminação e do algoritmo utilizado (NIST, 2023c; SHARMA et al., 2023).
- Outro indício descrito em estudos é o comportamento dos reflexos e do brilho especular, que pode denunciar a superfície emissiva da tela usada no ataque, especialmente quando combinado com análise temporal e qualidade da imagem (NIST, 2023c; SHARMA et al., 2023).
- Quando sensores de profundidade estão disponíveis, o mapa facial de um replay pode indicar uma superfície plana em vez do relevo natural do rosto humano. Esse recurso, porém, não faz parte da implementação deste trabalho (ISO/IEC 30107-3, 2023; APPLE, 2024; MICROSOFT, 2026).
- Alguns trabalhos também investigam sinais fisiológicos estimados por vídeo, como rPPG, para reforçar a distinção entre rosto real e mídia reexibida. Entretanto, essa técnica não foi implementada neste projeto e é mencionada apenas como referência teórica (NIST, 2023c; SHARMA et al., 2023).
- No contexto deste TCC, a principal defesa contra replay é o desafio-resposta visual: o sistema solicita movimentos imprevisíveis ao usuário, tornando difícil que uma foto ou vídeo pré-gravado acompanhe corretamente a sequência exigida em tempo real (ADAMIAK; ZUREK; SLOT, 2015; ISO/IEC 30107-3, 2023).

2.5.3 Como o sistema detecta máscara 3D

Máscaras tridimensionais representam um ataque mais elaborado do que fotos ou vídeos, pois podem apresentar volume e contornos faciais semelhantes aos de um rosto humano. Ainda assim, a literatura aponta diferenças de material, textura, comportamento térmico e dinâmica facial que podem ser exploradas por sistemas de PAD, especialmente quando há sensores adicionais.

- Assinatura material: máscaras de silicone/impressas podem apresentar textura homogênea, poros artificiais e resposta espectral diferente da pele humana, especialmente quando são empregados sensores NIR/IR. Esse tipo de análise é descrito na literatura de PAD,

mas não foi implementado neste projeto (ISO/IEC 30107-3, 2023; NIST, 2023c; SHARMA et al., 2023).

- Profundidade coerente porém “suave”: máscaras 3D podem apresentar volume, mas com rigidez e pouca naturalidade em microexpressões e movimentos de músculos e pele. Em sistemas com sensores de profundidade, esse comportamento pode ser explorado como pista complementar; neste TCC, porém, a solução usa apenas câmera RGB (ISO/IEC 30107-3, 2023; SHARMA et al., 2023).
- Em abordagens com termografia ou infravermelho ativo, a distribuição térmica e espectral pode diferir entre pele humana e materiais artificiais. Tais recursos são exemplos da literatura, mas não compõem a solução implementada neste projeto (ISO/IEC 30107-3, 2023; NIST, 2023c).
- Da mesma forma, alguns estudos citam sinais fisiológicos ou variações ópticas sutis como apoio à detecção de máscaras. No entanto, essas estratégias não foram usadas aqui (NIST, 2023c; SHARMA et al., 2023).
- No presente trabalho, a resistência contra apresentações artificiais está concentrada no desafio-resposta visual e na exigência de interação em tempo real diante da câmera RGB, abordagem coerente com o escopo aplicado do protótipo (ADAMIAK; ZUREK; SLOT, 2015; ISO/IEC 30107-3, 2023).

2.5.4 Como o sistema detecta deepfakes (mídia sintética)

Deepfakes e outras mídias sintéticas introduzem um desafio diferente dos ataques físicos tradicionais, pois a falsificação pode ser gerada digitalmente e apresentada como imagem ou vídeo aparentemente real. Os tópicos seguintes resumem pistas exploradas pela literatura para identificar manipulações faciais, sem afirmar que todas foram implementadas neste trabalho.

- Inconsistências faciais finas: alinhamento imperfeito entre dentes e lábios, padrões de piscada pouco fisiológicos, direção do olhar incompatível e falhas em microexpressões aparecem na literatura como pistas possíveis para detecção de mídia sintética (LI; CHANG; LYU, 2018; TOLOSANA et al., 2020; NGUYEN et al., 2022).
- Pistas frequenciais/espaciais: upsampling, blending e compressão podem gerar padrões em altas frequências, bordas artificiais e perda de microtexturas; CNNs e analisadores forenses especializados podem explorar essas pistas, embora a generalização contra novos geradores continue sendo um desafio (TOLOSANA et al., 2020; NGUYEN et al., 2022).

- Coerência temporal: flicker na pele entre quadros, ghosting em contornos e deformações temporais em cabelo ou bordas do rosto são exemplos de pistas exploradas por detectores de manipulação facial em vídeo (TOLOSANA et al., 2020; NGUYEN et al., 2022).
- Alguns sistemas também analisam coerência audiovisual entre fala e movimento labial. Essa possibilidade é citada apenas para contextualização bibliográfica e não integra o escopo deste TCC, que não utiliza áudio nem comandos por fala (TOLOSANA et al., 2020; NGUYEN et al., 2022).
- Assinatura fotométrica: reflexos especulares nos olhos/pele com formas/posições incoerentes com a cena real podem indicar manipulação ou apresentação não bona fide, dependendo do sensor e do algoritmo utilizado (NIST, 2023c; SHARMA et al., 2023).
- Verificações contextuais: metadados, origem do arquivo e inconsistência de iluminação/pose com o local declarado podem complementar mecanismos de detecção, especialmente contra mídia sintética ou conteúdo reprocessado (TOLOSANA et al., 2020; NGUYEN et al., 2022).

Detectores de deepfake precisam de reavaliação e atualização contínua, pois novos métodos de síntese podem reduzir ou eliminar artefatos previamente explorados. Por isso, a literatura recomenda defesa em camadas, combinando PAD, checagens contextuais, procedência de mídia e, quando aplicável, verificação audiovisual (TOLOSANA et al., 2020; NGUYEN et al., 2022; NIST, 2023c).

2.5.5 Tecnologias/sensores usados

A escolha dos sensores influencia diretamente o nível de segurança e robustez de uma solução de reconhecimento facial. Enquanto câmeras RGB são acessíveis e suficientes para protótipos e aplicações de menor risco, sensores infravermelhos, de profundidade ou soluções multimodais podem ampliar a capacidade de detecção de ataques de apresentação.

- RGB (câmeras comuns): base para reconhecimento facial e para parte dos métodos de PAD passivo/ativo por software, embora com limitações em relação a profundidade e infravermelho (ISO/IEC 30107-3, 2023; NIST, 2023c).
- NIR/IR ativo: exemplo de tecnologia empregada em algumas soluções comerciais e laboratoriais para melhorar a robustez do liveness. Não foi utilizado neste projeto (ISO/IEC 30107-3, 2023; APPLE, 2024).

- Profundidade (ToF/Luz Estruturada/Estéreo): recurso presente em determinados dispositivos para obter mapa 3D do rosto e elevar a segurança contra ataques 2D. Não foi utilizado neste projeto (APPLE, 2024; MICROSOFT, 2026).
- Microfone e áudio: podem ser empregados em sistemas multimodais para desafios falados e análise de coerência audiovisual. Não foram utilizados neste projeto (TOLOSANA et al., 2020; NGUYEN et al., 2022).
- Software (CNNs de PAD): classificadores treinados para distinguir bona fide versus ataque; analisam textura, frequência e temporalidade (NIST, 2023c; SHARMA et al., 2023).
- rPPG e outros sinais fisiológicos estimados por vídeo aparecem em parte da literatura de liveness como técnicas complementares. Também não foram utilizados neste projeto (NIST, 2023c; SHARMA et al., 2023).

2.5.6 Métricas específicas de PAD (interpretação simples)

Além das métricas tradicionais de reconhecimento facial, sistemas de PAD exigem indicadores próprios para medir separadamente ataques aceitos por engano e usuários legítimos rejeitados indevidamente. As métricas abaixo são apresentadas de forma interpretativa, pois serviram como referencial teórico, mas não foram calculadas formalmente no experimento exploratório deste TCC.

- APCER (Attack Presentation Classification Error Rate): % de ataques aceitos por engano, isto é, apresentações de ataque classificadas indevidamente como bona fide (ISO/IEC 30107-3, 2023).
- BPCER (Bona Fide Presentation Classification Error Rate): % de pessoas reais rejeitadas por engano, isto é, apresentações bona fide classificadas indevidamente como ataque (ISO/IEC 30107-3, 2023).
- DET/ROC de PAD: curva que mostra o compromisso entre segurança (baixo APCER) e usabilidade (baixo BPCER), podendo apoiar a escolha do ponto operacional do sistema (ISO/IEC 30107-3, 2023).

2.5.7 Boas práticas para implantação

Com base nos limites discutidos nas seções anteriores, a implantação de reconhecimento facial deve considerar não apenas o algoritmo, mas também o ambiente físico, o risco da aplicação, a qualidade da captura, a governança dos dados e as camadas de segurança contra fraude. As recomendações a seguir sintetizam cuidados práticos para uso responsável da tecnologia.

- Defina sensores conforme risco: câmera RGB pode ser adequada em cenários de menor criticidade, enquanto aplicações de maior risco podem exigir combinação com IR, profundidade, liveness comercial validado ou segundo fator de autenticação (ISO/IEC 30107-3, 2023; MICROSOFT, 2026; AMAZON WEB SERVICES, 2026a).
- Calibre a iluminação e a posição do usuário antes da captura, evitando contraluz, subexposição, blur e oclusões. Em vez de tratar 300–500 lux como norma universal, este TCC considera essa faixa uma diretriz prática de laboratório; a qualidade final deve ser validada pelo próprio sistema e pelo cenário de uso (NIST, 2023a; NIST, 2023b; NIST, 2023c).
- Quando disponível, utilize PAD passivo e complemente com desafios ativos curtos e aleatórios conforme o risco da aplicação e a experiência do usuário esperada (ISO/IEC 30107-3, 2023; MICROSOFT, 2026; AMAZON WEB SERVICES, 2026b).
- Monitore qualidade de captura (blur, SNR, exposição e oclusão) e aplique fallback, como recaptura ou segundo fator, quando a confiança do reconhecimento ou do PAD estiver incerta (NIST, 2023a; NIST, 2023b; NIST, 2023c).
- Audite periodicamente com testes de foto, replay e (se possível) máscara; registre APCER/BPCER.
- Atualize modelos (PAD/deepfake) e reavalie thresholds por cenário real.

2.5.8 Referencial teórico aplicado ao trabalho

Como fechamento da fundamentação teórica deste capítulo, esta seção retoma os conceitos apresentados e explicita como eles sustentam o método adotado no trabalho. Do ponto de vista do referencial teórico, este TCC articula três eixos centrais da literatura: (i) modelos de reconhecimento facial baseados em embeddings profundos, como FaceNet e ArcFace; (ii) avaliação de desempenho por métricas biométricas clássicas, usadas aqui como referencial teórico, e por métricas efetivamente executadas no protocolo, como acurácia rank-1, taxa de detecção e erros de identidade; e (iii) mecanismos de segurança contra ataques de apresentação e mídias sintéticas (SCHROFF et al., 2015; DENG et al., 2019; NIST, 2023a; NIST, 2023b; ISO/IEC 30107-3, 2023).

No primeiro eixo, os trabalhos de Schroff et al. (2015) e Deng et al. (2019) são especialmente relevantes porque consolidam a noção de embedding facial como representação discriminativa de identidade. Em vez de comparar imagens brutas, esses métodos projetam a face em um espaço vetorial no qual amostras da mesma pessoa tendem a permanecer próximas e amostras de pessoas diferentes tendem a permanecer separadas. Essa base teórica

sustenta a etapa de comparação utilizada neste projeto, mesmo quando a implementação prática recorre a bibliotecas prontas, como dlib/face_recognition ou variantes compatíveis.

No segundo eixo, a literatura de avaliação biométrica e os relatórios do NIST mostram que desempenho de reconhecimento facial depende fortemente do cenário de captura, da qualidade da imagem e do limiar operacional adotado. Por isso, a análise deste trabalho considera condições ambientais adversas, como baixa luz, contraluz e redução de resolução, relacionando tais fatores às métricas de erro e aos limites de confiabilidade do sistema. Em outras palavras, o referencial adotado sustenta que acurácia isolada não é suficiente; é necessário observar também a taxa de falsas aceitações, falsas rejeições e o comportamento do sistema em diferentes condições operacionais (NIST, 2023a; NIST, 2023b; NIST, 2023c).

No terceiro eixo, a literatura de PAD/liveness e a série ISO/IEC 30107 mostram que o reconhecimento facial, quando utilizado para autenticação, precisa ser complementado por mecanismos de defesa contra fotos impressas, vídeos em tela, máscaras e outras formas de apresentação fraudulenta. Nesse contexto, a bibliografia distingue abordagens passivas, que tentam inferir vivacidade a partir da própria imagem ou do vídeo, e abordagens ativas, que exigem alguma ação do usuário. Soluções comerciais e institucionais recentes, como as descritas por Microsoft Azure Face Liveness e Amazon Rekognition Face Liveness, mostram que o uso de vídeo-selfie, checagens de presença física, desafios de movimento e análise de liveness é uma estratégia reconhecida na prática para reduzir spoofing e replay (ISO/IEC 30107-3, 2023; MICROSOFT, 2026; AMAZON WEB SERVICES, 2026a; AMAZON WEB SERVICES, 2026b).

A posição teórica deste trabalho, portanto, é a de que o método implementado — PAD ativo por desafio-resposta visual com câmera RGB — não constitui uma invenção isolada, mas uma adaptação prática de uma classe de técnicas discutida na literatura de PAD e em soluções comerciais de liveness. A contribuição do projeto está em integrar esse mecanismo a um protótipo acadêmico de reconhecimento facial, discutir sua viabilidade no contexto de implementação adotado e compará-lo, de forma justificada, com a alternativa de liveness passivo que chegou a ser considerada no planejamento, mas não pôde ser incorporada por limitações de compatibilidade no ambiente Python utilizado (ISO/IEC 30107-3, 2023; NIST, 2023c; MICROSOFT, 2026; AMAZON WEB SERVICES, 2026a).

2.6 Vieses Demográficos e Equidade

Relatos acadêmicos apontam variação de desempenho entre grupos demográficos e sob diferentes regimes de iluminação, com potenciais vieses se a curadoria de dados não for adequada (WU et al., 2022; TERHÖRST et al., 2022; KOTWAL; MARCEL, 2025). Boas práticas incluem auditorias periódicas, thresholds por contexto, documentação de métricas estratificadas e governança de dados para evitar decisões automatizadas injustas ou opacas (BRASIL, 2018; ANPD, 2024).

2.7 Aspectos Legais e Éticos (Brasil)

A LGPD classifica dados biométricos como dados pessoais sensíveis, exigindo base legal adequada, finalidade específica, minimização, segurança, transparência e, em situações de maior risco, avaliação de impacto/Relatório de Impacto à Proteção de Dados Pessoais (RIPD/DPIA) (BRASIL, 2018; ANPD, 2024). Normas ISO/IEC 30107-3 orientam ensaios e relatórios de PAD; guias de plataformas com sensor 3D/IR, como o Face ID, ilustram arquiteturas seguras que combinam hardware dedicado e processamento local (ISO/IEC 30107-3, 2023; APPLE, 2024). Essas referências fornecem o lastro normativo e técnico para as decisões de projeto discutidas na metodologia.

3 METODOLOGIA

Esta pesquisa, segundo a sua natureza, caracteriza-se como aplicada, pois visa produzir conhecimento com finalidade prática (avaliação e diretrizes de implantação de reconhecimento facial). Pela abordagem, é experimental e quantitativa, com coleta estruturada de imagens em cenários controlados e cálculo das métricas efetivamente executadas no protocolo: acurácia rank-1 em regime fechado, taxa de detecção facial, falhas de detecção, erros de identidade e tempo de execução. Para o PAD ativo, a análise foi exploratória, baseada em tentativas de fraude por foto e vídeo e na taxa descritiva de sucesso dos ataques, sem cálculo formal de APCER/BPCER. O delineamento segue boas práticas de avaliação biométrica (NIST, 2023a; NIST, 2023b) e, para segurança, referência a ISO/IEC 30107-3 e ao relatório NISTIR 8491 (ISO/IEC 30107-3, 2023; NIST, 2023c).

3.1 Ambiente, Participantes e Ética

Esta seção descreve as condições materiais e éticas sob as quais o experimento foi conduzido. Como o estudo envolveu imagens faciais, optou-se por um ambiente controlado, base restrita e tratamento local dos dados, de modo a reduzir riscos de exposição indevida e manter coerência com os princípios da LGPD.

- Local e equipamento. Sala de testes com controle básico de iluminação; câmera 1080p/30fps (ou superior) e tripé; computador com Python e bibliotecas (OpenCV e face_recognition/InsightFace).

- A base de dados utilizada nos experimentos foi composta por uma amostra restrita ao núcleo familiar, englobando o próprio autor e seus pais. Os resultados de reconhecimento foram obtidos a partir dessa base reduzida e controlada, sendo as limitações de generalização decorrentes do tamanho da amostra devidamente discutidas na seção 4.
- Aspectos éticos. Consentimento para a coleta e o uso das imagens obtido diretamente com os participantes.

- Privacidade. Todos os dados ficam localmente, com pseudonimização e criptografia de pastas, quando possível, em alinhamento com princípios de minimização, segurança e controle de acesso da LGPD (BRASIL, 2018; ANPD, 2024).

3.2 Materiais e Software

A implementação do protótipo exigiu bibliotecas para captura, detecção, alinhamento, extração de embeddings e comparação de identidades. Os materiais e softwares listados a

seguir formam a base operacional utilizada nos testes comparativos entre dlib/face_recognition e ArcFace + RetinaFace.

- Bibliotecas/modelos. OpenCV; face_recognition (pipeline dlib/HOG/CNN + embeddings) e ArcFace com detector RetinaFace e extração de embeddings com margem angular (DENG et al., 2019). Nos experimentos finais, o pipeline ArcFace foi executado com RetinaFace, pois a combinação mostrou maior robustez de detecção e alinhamento do que a configuração inicial com detector OpenCV.
- Scripts. Listagem 1 - reconhecedor_facial.py (captura e verificação 1:1 em tempo real), Listagem 2 - experimento.py (avaliação por cenários com dlib/face_recognition) e Listagem 3 - arcface_retinaface_teste.py (avaliação por cenários com ArcFace + RetinaFace) - ver Apêndice C.
- Organização de dados. Pastas: imagens_conhecidas/ (cadastro) e imagens_teste/{Ideal,Baixa_luz,Contra_luz}/ (probes).

3.3 Protocolo de Avaliação e Critérios de Análise

O protocolo experimental foi estruturado para comparar os dois pipelines sob as mesmas condições de captura e com a mesma regra de decisão. A opção por avaliação em regime fechado permitiu medir acurácia rank-1, taxa de detecção e erros de identidade de forma direta, evitando a introdução de limiares diferentes entre os modelos.

- Os mesmos cenários de captura foram executados para dois pipelines distintos: dlib/face_recognition e ArcFace + RetinaFace. Para evitar uma comparação enviesada por thresholds diferentes de “desconhecido”, a avaliação principal foi conduzida em regime fechado (closed-set rank-1): após a detecção da face e a geração do embedding, cada imagem de teste foi comparada com a base cadastrada e classificada pelo cadastro mais próximo. O resultado foi contado como acerto apenas quando o nome previsto coincidiu com o nome esperado.
- As métricas efetivamente reportadas neste trabalho para o reconhecimento facial foram: acurácia rank-1 por cenário, taxa de detecção facial, número de falhas de detecção, número de erros de identidade e tempo de execução. Essas medidas foram escolhidas porque refletem com fidelidade o que o protótipo realmente entregou na prática e permitem observar separadamente se o problema ocorreu na detecção ou no reconhecimento.
- Métricas como FAR, FRR, EER e curvas ROC/DET permanecem relevantes na literatura biométrica, mas sua estimação adequada exigiria variação sistemática de limiares e

um protocolo mais amplo de comparações entre amostras genuínas e impostoras. Como esse protocolo completo não foi executado nesta etapa do trabalho, tais métricas são tratadas aqui como referencial teórico, e não como resultado empírico do experimento de reconhecimento.

- Para o PAD ativo, foi conduzida apenas uma avaliação exploratória de ataques, com 20 tentativas usando fotos, 13 tentativas usando vídeos comuns e 12 tentativas usando vídeos em que o atacante conhecia previamente o conjunto de desafios possíveis, mas não a ordem em que seriam solicitados. Esse procedimento permitiu observar o comportamento prático do mecanismo contra fraudes estáticas e dinâmicas. Contudo, como não foi estruturado um protocolo formal com amostras bona fide suficientes, isto é, amostras legítimas de usuários reais sem tentativa de ataque para estimar BPCER, os resultados de PAD são apresentados como evidência experimental preliminar, e não como validação biométrica completa.

O script de experimento, apresentado no apêndice, percorre automaticamente as pastas de teste por cenário e consolida os resultados de forma reprodutível. Na etapa final do TCC, essa rotina foi aplicada tanto ao pipeline com dlib quanto ao pipeline com ArcFace, o que permitiu comparar os dois modelos sob as mesmas imagens de teste e com a mesma regra de contagem de acertos e erros.

3.4 Bibliotecas usadas e fundamentos matemáticos

3.4.1 OpenCV e NumPy

O protótipo utiliza OpenCV, NumPy, dlib/face_recognition e ArcFace com detector RetinaFace. O fluxo básico do sistema é: captura de imagem (OpenCV) → detecção e alinhamento de faces (dlib ou RetinaFace) → extração de embeddings (dlib: 128-D, ArcFace: 512-D) → comparação de embeddings (distância euclidiana ou similaridade do cosseno) → decisão final. O modelo dlib/face_recognition é amplamente utilizado devido à sua simplicidade e integração, enquanto ArcFace melhora a discriminatividade dos embeddings por meio de margens angulares. Nos experimentos finais deste TCC, a configuração ArcFace + RetinaFace foi adotada para representar de forma mais fiel o potencial do modelo em condições adversas (BRADSKI, 2000; HARRIS et al., 2020; KING, 2009; AGEITGEY, 2025; DENG et al., 2019; DENG et al., 2020; INSIGHTFACE CONTRIBUTORS, 2025).

3.4.2 Detecção e alinhamento (dlib/face_recognition; alternativas MTCNN/RetinaFace)

A detecção de rostos inicial pode ser feita com técnicas clássicas como HOG (Histogram of Oriented Gradients) combinado a SVM (Support Vector Machine), abordagem que calcula gradientes locais e usa essas características para classificação. Em pipelines modernos, redes neurais convolucionais e detectores como MTCNN e RetinaFace tendem a oferecer maior robustez de detecção e alinhamento em variações de pose, escala e iluminação. No pipeline final com ArcFace adotado neste trabalho, o detector RetinaFace foi utilizado para fortalecer essa etapa de detecção e alinhamento facial (DALAL; TRIGGS, 2005; ZHANG et al., 2016; DENG et al., 2020; NIST, 2023a; NIST, 2023b).

3.4.3 Extração de embeddings (dlib 128-D e ArcFace 512-D) e normalização

Após a detecção e o alinhamento das faces, o próximo passo é a extração dos embeddings, vetores numéricos que representam características discriminativas da face. O modelo dlib/face_recognition utiliza uma rede neural que gera embeddings de 128 dimensões (128-D), enquanto o ArcFace utiliza embeddings de 512 dimensões e treinamento com margem angular. A normalização L2 torna os vetores comparáveis em escala e facilita o uso de métricas como distância euclidiana e similaridade do cosseno (SCHROFF et al., 2015; KING, 2009; AGEITGEY, 2025; DENG et al., 2019).

Em modelos de aprendizagem métrica, a separação entre amostras da mesma identidade e de identidades distintas pode ser representada pela perda triplet, conforme a Equação (1) (SCHROFF et al., 2015):

$$L_{triplet} = \sum max\left(0, \| f(a) - f(p) \|_2^2 - \| f(a) - f(n) \|_2^2 + \alpha\right) \quad (1)$$

Onde α é a margem que força a separação entre a face positiva e a negativa, e f(a), f(p) e f(n) são as saídas da rede neural para o anchor, positive e negative, respectivamente.

ArcFace (AAM-Softmax): a função de perda AAM-Softmax foi desenvolvida para melhorar a precisão, ajustando a separação angular entre os embeddings das diferentes identidades. A fórmula é dada por (DENG et al., 2019):

$$L_{arc} = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{scos\left(\theta_{y_i}+m\right)}}{e^{scos\left(\theta_{y_i}+m\right)} + \sum_{j\neq y_i} e^{scos\left(\theta_j\right)}} \quad (2)$$

Onde θ representa o ângulo entre o vetor do embedding e o vetor do centro de classe (classificação), m é a margem angular aplicada e “s” é o fator de escala que ajusta a separação. Esse método promove aumento da separação entre as identidades e melhora a compactação intra-classe (DENG et al., 2019).

3.4.4 Comparação e decisão (fórmulas usadas no código)

Para comparar os embeddings extraídos, utilizamos duas métricas principais: distância euclidiana e similaridade do cosseno.

- Distância euclidiana: calcula a distância entre dois vetores no espaço vetorial. A fórmula é:

$$d(x, y) = \| x - y \|_2 = \sqrt{} \quad (3)$$

Essa medida reflete a diferença bruta entre as coordenadas de cada vetor, considerando x e y como vetores de dimensão d.

- Similaridade do cosseno: mede o ângulo entre dois vetores, refletindo a sua similaridade:

$$s_{cos}(x, y) = \frac{x \cdot y}{\|x\|_2 \|y\|_2} = x \cdot y \in [-1, 1] \quad (4)$$

No código, quando os vetores são normalizados, a distância cosseno é diretamente relacionada ao grau de similaridade.

Relação útil (com $\|x\|_2 = \|y\|_2 = 1$): $d^2(x,y) = 2(1 - s\_cos(x,y))$. Assim, maximizar a similaridade do cosseno equivale a minimizar a distância euclidiana entre vetores normalizados. No código do face_recognition, o padrão é usar distância euclidiana com tolerância τ; a decisão para um match é dada por $d(x,y) \leq \tau$. Em ArcFace, utiliza-se similaridade do cosseno e o match é detectado quando $s\_cos(x,y) \geq \sigma$ (SCHROFF et al., 2015; DENG et al., 2019; AGEITGEY, 2025).

3.4.5 Decisão, limiar e métricas no contexto deste trabalho

Na literatura biométrica, a variação de limiares permite obter curvas ROC/DET e estimar FAR, FRR e EER. No entanto, como o objetivo desta etapa foi comparar dois

pipelines sobre um conjunto pequeno de imagens previamente conhecidas, optou-se por uma regra única e mais transparente: selecionar sempre o cadastro mais próximo e verificar se ele coincide com a identidade esperada. Essa escolha reduz a influência de thresholds arbitrários e torna a comparação entre dlib e ArcFace mais justa para a base utilizada neste TCC.

3.4.6 Observações práticas

A partir da implementação e dos resultados obtidos, algumas observações práticas ajudam a interpretar o comportamento dos pipelines e a orientar possíveis reproduções do experimento. Esses pontos destacam limitações operacionais que podem afetar o desempenho de reconhecimento facial em aplicações reais.

• Qualidade primeiro: uma falha de detecção/alinhamento inviabiliza a comparação; recomenda-se monitorar blur, SNR, exposição e oclusões antes da extração de embeddings (NIST, 2023a; NIST, 2023b).

• Pipeline completo importa: os resultados deste TCC mostraram que a combinação ArcFace + RetinaFace foi superior à configuração inicial com detector OpenCV, evidenciando que a robustez final depende tanto do embedding quanto da detecção e do alinhamento facial (DENG et al., 2019; DENG et al., 2020).

• Domínio e limiar: thresholds devem ser ajustados por cenário de captura, risco da aplicação e custo relativo de falsas aceitações e falsas rejeições (NIST, 2023a; NIST, 2023b).

• Dimensionalidade: vetores 512-D, como os usados pelo ArcFace, tendem a ser mais discriminativos que vetores 128-D em modelos mais simples, embora possam exigir modelos maiores e mais processamento (SCHROFF et al., 2015; DENG et al., 2019).

3.4.7 Decisão Arquitetural e Implementação do PAD Ativo

No planejamento inicial deste TCC, considerou-se a integração de uma camada de liveness passivo por software. Em uma arquitetura desse tipo, o sistema analisaria automaticamente pistas visuais do vídeo — como textura facial, reflexos, microvariações temporais e indícios de reprojeção em tela ou papel — antes de permitir a etapa de reconhecimento. Essa solução é atraente por exigir menos interação do usuário e por se alinhar a abordagens recentes de PAD descritas na literatura e em serviços comerciais (NIST, 2023c; SHARMA et al., 2023; MICROSOFT, 2026; AMAZON WEB SERVICES, 2026a).

Entretanto, a adoção dessa estratégia não foi concluída no ambiente de desenvolvimento utilizado no projeto. Durante a implementação, surgiram incompatibilidades de versão entre bibliotecas Python necessárias para o pipeline experimental de liveness, o que inviabilizou a estabilização da solução no prazo do trabalho. Por essa razão, o liveness

passivo permaneceu como alternativa estudada teoricamente, mas não como componente efetivamente validado neste TCC.

Como solução implementável dentro das restrições reais do projeto, adotou-se um PAD ativo baseado em desafio-resposta visual. Nessa abordagem, o usuário precisa executar comandos simples exibidos pelo sistema — por exemplo, virar o rosto para a direita, para a esquerda ou alinhar-se corretamente à câmera — e o reconhecimento só prossegue quando a sequência solicitada é cumprida dentro dos critérios definidos.

Na prática, a lógica do protótipo acompanha landmarks faciais e verifica se houve movimento compatível com o desafio apresentado. Se o comportamento esperado não for observado, a tentativa é bloqueada e o sistema interpreta a interação como incompatível com uma apresentação facial autêntica naquele instante, o que ajuda a dificultar fraudes com fotografia, vídeo de replay ou outra mídia estática reproduzida diante da câmera (ADAMIAK; ZUREK; SLOT, 2015; ISO/IEC 30107-3, 2023).

É importante registrar, porém, que esta implementação do PAD ativo foi avaliada de modo funcional, e não por um protocolo quantitativo formal com APCER/BPCER ou banco estruturado de ataques. Assim, o trabalho demonstra a integração e o funcionamento lógico do mecanismo de desafio-resposta, mas não reivindica, nesta versão, uma taxa estatística formal de eficácia contra foto, vídeo, máscara ou deepfake.

Mesmo com essa limitação, a opção pelo PAD ativo foi coerente com o escopo aplicado do projeto: ela elevou a segurança do protótipo com recursos acessíveis, preservou a execução local em câmera RGB e permitiu discutir, de forma honesta, a diferença entre uma solução efetivamente implementada e uma solução apenas considerada no planejamento.

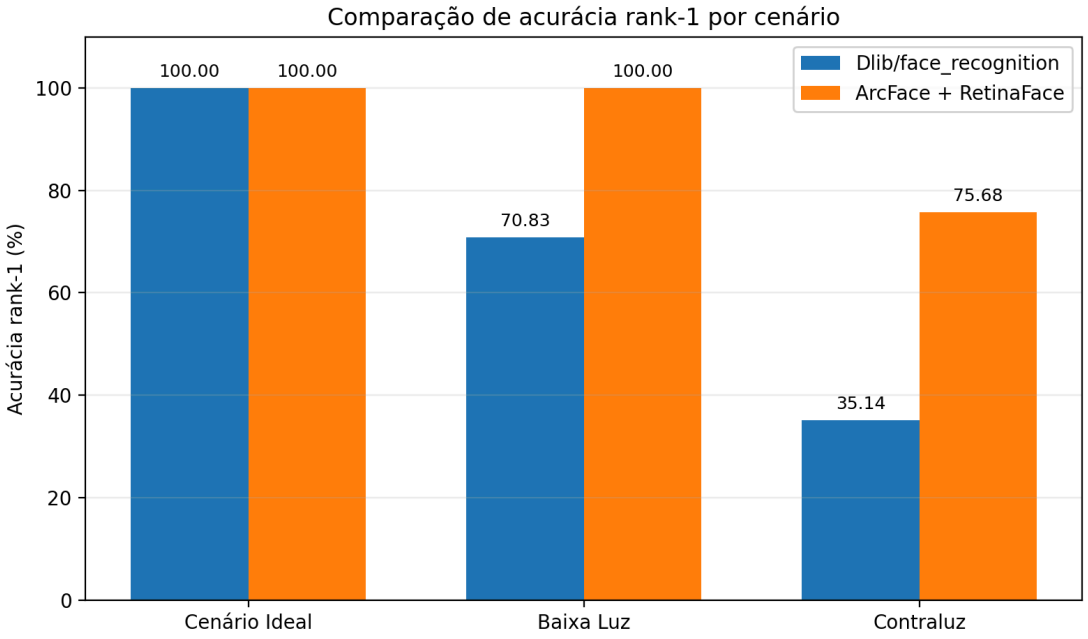
4 RESULTADOS E DISCUSSÃO

4.1 Resultados reais obtidos

Os resultados efetivamente obtidos neste trabalho foram consolidados a partir da execução dos mesmos cenários de teste nos dois pipelines avaliados: dlib/face_recognition e ArcFace. A comparação foi realizada em regime fechado, com regra de decisão idêntica para ambos os modelos, de modo que cada imagem de teste fosse atribuída ao cadastro mais próximo e contabilizada como acerto somente quando a identidade prevista coincidisse com a identidade esperada.

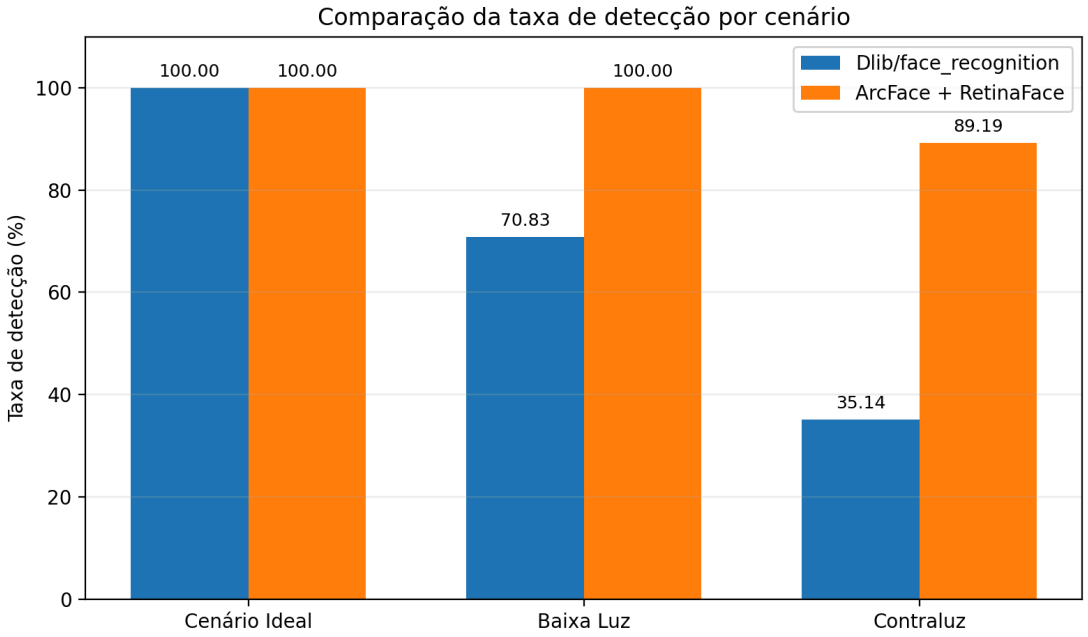
No conjunto final de imagens utilizado, o pipeline ArcFace + RetinaFace apresentou desempenho global superior ao pipeline baseado em dlib/face_recognition. Somando os cenários com imagens válidas, o ArcFace + RetinaFace obteve 65 acertos em 74 imagens (87,84%), enquanto o dlib obteve 43 acertos em 74 imagens (58,11%). Em termos de detecção facial, o ArcFace + RetinaFace detectou 70 das 74 faces testadas (94,59%), ao passo que o dlib detectou 43 das 74 faces (58,11%). Esses números mostram que a reconfiguração do pipeline do ArcFace, com uso do detector RetinaFace, mudou substancialmente o resultado comparativo e passou a favorecer o modelo teoricamente mais forte.

Figura 6 - Comparação da acurácia rank-1 por cenário entre os pipelines dlib e ArcFace. 27



Fonte: autoria própria (2026).

Figura 7 - Comparação da taxa de detecção facial por cenário entre os pipelines dlib e ArcFace. 28



Fonte: autoria própria (2026).

No cenário ideal, ambos os pipelines apresentaram desempenho máximo: tanto o dlib quanto o ArcFace + RetinaFace alcançaram 100,00% de acurácia (13/13), com 100,00% de taxa de detecção e nenhum erro de identidade. Esse resultado indica que, quando a qualidade da amostra é alta e a face está frontalmente iluminada, ambos os sistemas operam de forma estável.

Em baixa luz, a diferença tornou-se bastante expressiva. O dlib atingiu 70,83% de acurácia (17/24), com 7 falhas de detecção, enquanto o ArcFace + RetinaFace atingiu 100,00% de acurácia (24/24), sem falhas de detecção e sem erros de identidade. Nesse cenário, a superioridade do ArcFace + RetinaFace apareceu principalmente na robustez da etapa de detecção e alinhamento, já que o pipeline conseguiu localizar e reconhecer todas as faces do conjunto, mesmo sob iluminação degradada.

Em contraluz, o cenário continuou sendo o mais crítico do experimento, mas o ArcFace + RetinaFace ainda manteve vantagem considerável: 75,68% de acurácia (28/37) contra 35,14% do dlib (13/37). A taxa de detecção também favoreceu o ArcFace + RetinaFace, com 89,19% (33/37), enquanto o dlib permaneceu em 35,14% (13/37). Por outro lado, o ArcFace + RetinaFace registrou 5 erros de identidade nesse cenário, ao passo que o dlib não produziu confusões entre pessoas nas imagens detectadas. Isso indica que o pipeline com RetinaFace foi muito mais sensível para localizar rostos em contraluz, mas ainda

enfrentou alguma degradação discriminativa quando a amostra detectada estava severamente comprometida pela iluminação.

Também foram realizados testes exploratórios do PAD ativo implementado no protótipo. Nas 20 tentativas com foto, o sistema bloqueou todas as apresentações fraudulentas, sem nenhum caso de acesso liberado. Nos 13 testes com vídeos comuns, em que a pessoa realizava movimentos naturais sem sincronização deliberada com os desafios, apenas 2 vídeos conseguiram levar o sistema a aceitar a tentativa. Em um terceiro conjunto, com 12 vídeos nos quais o atacante já conhecia quais tipos de desafio podiam ser solicitados, mas não sabia a ordem exata em que apareceriam, 5 vídeos conseguiram passar pelo mecanismo. Esses resultados mostram que o PAD ativo foi eficaz contra ataques estáticos por fotografia e apresentou resistência parcial contra ataques por replay em vídeo, embora ainda vulnerável a vídeos mais preparados.

Em termos descritivos, isso correspondeu a taxa de sucesso do ataque de 0,00% para fotos, 15,38% para vídeos comuns e 41,67% para vídeos preparados com conhecimento prévio do conjunto de desafios. No total das 45 tentativas de ataque, 7 obtiveram aceitação indevida, o que equivale a 15,56% de sucesso do spoofing no protocolo exploratório adotado. Apesar de úteis para caracterizar o comportamento do protótipo, esses números não substituem uma avaliação formal com APCER/BPCER completos, pois ainda falta um protocolo bona fide padronizado e amostras mais amplas de ataque e de usuários legítimos.

4.2 Discussão dos resultados à luz da literatura

Do ponto de vista teórico, o ArcFace é amplamente considerado um modelo mais moderno e competitivo do que o dlib/face_recognition. Isso se deve, sobretudo, ao uso de embeddings de maior capacidade representacional e à função de perda com margem angular aditiva, proposta por Deng et al. (2019), que aumenta a separação entre classes no espaço vetorial e tende a produzir descritores faciais mais discriminativos em bases amplas e bem controladas. Em teoria, portanto, seria razoável esperar que o ArcFace superasse o dlib em acurácia de identificação facial, desde que fosse utilizado em um pipeline compatível com esse potencial.

Os resultados finais deste TCC mostraram que essa superioridade teórica passou a aparecer quando o ArcFace foi combinado com um detector mais robusto. Na configuração inicial, baseada em detector OpenCV, o ArcFace havia apresentado desempenho inferior ao dlib em cenários adversos. Após a substituição do detector por RetinaFace e a adoção de um pipeline mais adequado de detecção, alinhamento e extração de embeddings, o resultado se

inverteu: o ArcFace + RetinaFace passou a superar o dlib em desempenho global, especialmente em baixa luz e contraluz. A explicação mais consistente, portanto, não está em afirmar que o ArcFace "ficou melhor por si só", mas em reconhecer que a vantagem de um modelo moderno só se manifesta plenamente quando as etapas anteriores ao embedding também são suficientemente robustas.

No conjunto final deste trabalho, o ArcFace + RetinaFace manteve 100,00% de taxa de detecção no cenário ideal e também em baixa luz, enquanto em contraluz alcançou 89,19%. Já o dlib ficou em 100,00% no cenário ideal, 70,83% em baixa luz e 35,14% em contraluz. Esses números indicam que o gargalo principal estava na etapa de detecção e alinhamento facial, e não necessariamente no modelo de embedding isolado. Isso é compatível com a literatura técnica e com avaliações públicas do NIST, segundo as quais a qualidade da amostra de entrada influencia decisivamente o desempenho de sistemas faciais. Assim, um modelo teoricamente superior pode parecer inferior quando operado com detector, alinhamento ou pré-processamento inadequados, mas tende a recuperar sua vantagem quando essas etapas são fortalecidas.

Há ainda fatores metodológicos que ajudam a interpretar esse comportamento. A base usada neste trabalho foi ampliada, mas ainda permanece relativamente pequena, o número de identidades continua reduzido, o cadastro por pessoa é limitado e os cenários de captura apresentam variações severas de iluminação. Mesmo assim, a troca do detector OpenCV por RetinaFace alterou substancialmente a comparação empírica, o que reforça a importância da escolha do detector facial em aplicações reais. A leitura correta dos resultados, portanto, é aplicada e contextual: o ArcFace não deve ser avaliado isoladamente, mas em conjunto com o pipeline que o viabiliza. No recorte experimental final deste TCC, a combinação ArcFace + RetinaFace foi superior ao dlib nos cenários testados, sobretudo em baixa luz e contraluz.

A discussão do PAD ativo também precisa ser interpretada nessa chave analítica. Os testes exploratórios mostraram bloqueio total contra fotos, bom desempenho contra vídeos comuns e vulnerabilidade maior diante de vídeos preparados por um atacante que já conhecia o conjunto de desafios possíveis. Esse comportamento faz sentido à luz da própria arquitetura do protótipo: como o mecanismo depende de uma sequência curta de ações extraída de um vocabulário limitado de desafios, ataques estáticos tendem a fracassar, enquanto replays dinâmicos têm maior chance de sucesso quando conseguem coincidir parcialmente com a sequência exigida. Em outras palavras, o PAD implementado demonstrou utilidade prática e elevou a dificuldade de fraude, mas ainda não oferece evidência suficiente para sustentar uma resistência robusta contra ataques mais elaborados. Por isso, a principal contribuição neste

trabalho é mostrar a viabilidade funcional do desafio-resposta, ao mesmo tempo em que explicita a necessidade de avaliações futuras com APCER/BPCER, protocolos bona fide mais amplos e estratégias adicionais de defesa contra replay.

5 DIRETRIZES DE IMPLEMENTAÇÃO

A implementação de sistemas de reconhecimento facial requer uma atuação cuidadosa no ambiente de captura, com atenção especial à iluminação, distância da câmera, ângulo de visão, e qualidade da imagem. A iluminação é um fator crucial, pois pode afetar diretamente o desempenho do sistema. Em condições ideais, uma iluminação frontal difusa, com intensidade entre 300 a 500 lux, garante que a face esteja bem iluminada sem criar sombras ou brilho excessivo, o que ajuda na detecção e no alinhamento da face. No entanto, em cenários de baixa luz ou contraluz, o sistema pode ter dificuldades, principalmente devido à perda de textura facial. Nestes casos, uma luz auxiliar ou HDR pode ser necessária para melhorar a visibilidade. Além disso, é importante garantir que a câmera esteja estabilizada e que a distância entre o rosto e o sensor seja de aproximadamente 0,6 a 1,2 metros, o que ajuda a minimizar o desfoque e melhora a qualidade da imagem capturada (NIST, 2023a; NIST, 2023b).

Para obter resultados confiáveis, a escolha do hardware também desempenha um papel importante. Em cenários de baixo risco, onde a conveniência é priorizada, sensores RGB com PAD passivo básico podem ser suficientes. No entanto, em ambientes de maior risco, como controle de acesso a áreas restritas, recomenda-se a implementação de sensores IR/ToF ou luz estruturada, que oferecem PAD ativo e garantem a verificação de liveness. Esses sensores medem a profundidade da face, o que é essencial para impedir fraudes com fotos ou vídeos em tela, além de garantir que a face seja real e não uma máscara 3D. O uso de sensor 3D permite uma maior resiliência contra ataques de apresentação, uma vez que detecta a profundidade da face e distingue uma imagem plana de uma tridimensional (ISO/IEC 30107-3, 2023; APPLE, 2024).

Quanto ao software, a escolha da biblioteca e do modelo é outro fator importante. Os resultados finais deste TCC mostraram que o desempenho não depende apenas do poder teórico do embedding, mas do pipeline completo de detecção, alinhamento, extração e decisão. A comparação inicial com ArcFace associado a detector OpenCV não evidenciou a superioridade esperada do modelo. Entretanto, quando o pipeline foi reconfigurado para utilizar RetinaFace na etapa de detecção e alinhamento, o ArcFace passou a superar o dlib/face_recognition, especialmente em baixa luz e contraluz. Assim, a seleção tecnológica deve ser guiada por testes locais e pela compatibilidade entre detector e modelo, e não apenas por desempenho reportado em benchmarks isolados.

No que diz respeito à segurança, a detecção de ataques de apresentação (PAD) é um componente essencial para impedir fraudes. Neste trabalho, a solução implementada foi o

PAD ativo por desafio-resposta, adequado ao uso de câmera RGB e a um protótipo executado localmente. Como diretriz de melhoria, recomenda-se que versões futuras passem a registrar quantitativamente tentativas de foto e vídeo, distinguindo tipos de ataque e calculando métricas próprias de PAD. Também é recomendável explorar, quando o ambiente permitir, liveness passivo por software ou estratégias híbridas que combinem sinais passivos com desafios ativos.

Por fim, a governança e a conformidade legal também são fundamentais. De acordo com a LGPD (Lei Geral de Proteção de Dados), os dados biométricos devem ser tratados com base legal apropriada, além de garantir a minimização de dados e a segurança nas transações. A coleta e armazenamento de dados biométricos devem ser transparentes para os usuários, e DPIAs (avaliações de impacto de dados pessoais) devem ser realizadas quando necessário. Além disso, as empresas devem implementar auditorias regulares para garantir que os dados pessoais sejam mantidos de forma segura e que a tecnologia esteja em conformidade com as regulamentações em vigor. A implementação dessas diretrizes assegura que o sistema de reconhecimento facial seja não apenas eficiente, mas também ético e seguro para os usuários (BRASIL, 2018; ANPD, 2024).

6 CONCLUSÃO

Este trabalho conclui que o desempenho de um sistema de reconhecimento facial depende do pipeline completo de operação e das condições de captura, e não apenas do modelo de embedding adotado. Nos cenários avaliados, a combinação ArcFace + RetinaFace apresentou desempenho superior ao dlib/face_recognition, especialmente em baixa luz e contraluz, ao mesmo tempo em que confirmou a importância crítica da etapa de detecção e alinhamento facial. O PAD ativo por desafio-resposta também se mostrou funcional e eficaz contra fotografias, além de reduzir o sucesso de vídeos comuns, embora ainda tenha apresentado vulnerabilidade parcial diante de vídeos preparados.

Como contribuição, este trabalho apresenta uma avaliação aplicada e comparativa entre dois pipelines de reconhecimento facial em condições ambientais distintas, evidenciando que a robustez do sistema depende não apenas do modelo de embedding, mas também da qualidade da detecção e do alinhamento facial. Além disso, o estudo integra um mecanismo de PAD ativo por desafio-resposta ao protótipo, permitindo discutir, de forma prática, os ganhos e limites dessa abordagem contra tentativas de fraude por foto e vídeo.

As principais limitações do estudo foram a base reduzida de participantes, o protocolo experimental restrito para reconhecimento facial, a ausência de um procedimento completo para FAR, FRR e EER, e a não realização de uma avaliação formal de PAD com APCER/BPCER completos e conjunto bona fide padronizado, isto é, amostras legítimas de usuários reais sem tentativa de ataque de apresentação. Desse modo, os resultados do PAD devem ser entendidos como evidência exploratória, e não como certificação de segurança biométrica.

Como próximos passos, recomenda-se ampliar a base experimental com mais participantes e mais imagens por pessoa, repetir os cenários com maior diversidade de iluminação e pose, refinar a avaliação do ArcFace + RetinaFace em bases mais amplas e estruturar um protocolo formal de PAD com registro sistemático de ataques por foto e vídeo e cálculo de métricas específicas. Essas extensões permitiriam transformar o protótipo em uma avaliação biométrica mais abrangente, comparável e estatisticamente mais sólida.

REFERÊNCIAS

ADAMIAK, Krzysztof; ZUREK, Dominik; SLOT, Krzysztof. Liveness detection in remote biometrics based on gaze direction estimation. Annals of Computer Science and Information Systems, 2015. Disponível em: https://annals-csis.org/Volume_5/pliks/307.pdf. Acesso em: 25 mar. 2026.

AGEITGEY, Adam. face_recognition: simple facial recognition API for Python. GitHub, 2025. Disponível em: https://github.com/ageitgey/face_recognition. Acesso em: 10 nov. 2025.

AMAZON WEB SERVICES (AWS). Detecting face liveness - Amazon Rekognition Developer Guide. 2026a. Disponível em: https://docs.aws.amazon.com/rekognition/latest/dg/face-liveness.html. Acesso em: 25 mar. 2026.

AMAZON WEB SERVICES (AWS). FaceMovementAndLightClientChallenge - Amazon Rekognition API Reference. 2026b. Disponível em: https://docs.aws.amazon.com/rekognition/latest/APIReference/API_rekognitionstreaming_FaceMovementAndLightClientChallenge.html. Acesso em: 25 mar. 2026.

ANPD (Brasil). Radar Tecnológico - Biometria (reconhecimento facial). Brasília: Autoridade Nacional de Proteção de Dados, 2024. Disponível em: https://www.gov.br/anpd/pt-br/centrais-de-conteudo/documentos-tecnicos-orientativos/radar-tecnologico-biometria-anpd-1.pdf. Acesso em: 10 nov. 2025.

APPLE. About Face ID advanced technology. Apple Support, 2024. Disponível em: https://support.apple.com/pt-br/102381. Acesso em: 10 nov. 2025.

BRADSKI, Gary. The OpenCV Library. Dr. Dobb's Journal of Software Tools, 2000. Disponível em: https://opencv.org/. Acesso em: 10 nov. 2025.

BRASIL. Lei nº 13.709, de 14 de agosto de 2018 (Lei Geral de Proteção de Dados Pessoais - LGPD). Brasília: Presidência da República, 2018. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm. Acesso em: 10 nov. 2025.

CAO, Qiong; SHEN, Li; XIE, Weidi; PARKHI, Omkar M.; ZISSERMAN, Andrew. VGGFace2: A dataset for recognising faces across pose and age. In: IEEE International Conference on Automatic Face and Gesture Recognition, 2018. Disponível em: https://www.robots.ox.ac.uk/~vgg/data/vgg_face2/. Acesso em: 10 nov. 2025.

DALAL, Navneet; TRIGGS, Bill. Histograms of Oriented Gradients for Human Detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2005. Disponível em: https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf. Acesso em: 7 maio 2026.

DENG, Jiankang; GUO, Jia; NIINUMA, Koichi; et al. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. Disponível em: https://openaccess.thecvf.com/content_CVPR_2019/papers/Deng_ArcFace_Additive_Angular_Margin_Loss_for_Deep_Face_Recognition_CVPR_2019_paper.pdf. Acesso em: 10 nov. 2025.

DENG, Jiankang; GUO, Jia; ZHOU, Yuxiang; YU, Jinke; KOTSIA, Irene; ZAFEIRIOU, Stefanos. RetinaFace: Single-stage Dense Face Localisation in the Wild. arXiv:1905.00641, 2020. Disponível em: https://arxiv.org/abs/1905.00641. Acesso em: 7 maio 2026.

HARRIS, Charles R.; MILLMAN, K. Jarrod; VAN DER WALT, Stéfan J.; et al. Array programming with NumPy. Nature, v. 585, p. 357-362, 2020. Disponível em: https://www.nature.com/articles/s41586-020-2649-2. Acesso em: 10 nov. 2025.

HUANG, Gary B.; RAMESH, Manu; BERG, Tamara; LEARNED-MILLER, Erik. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. University of Massachusetts Amherst, 2007. Disponível em: https://people.cs.umass.edu/~elm/papers/lfw.pdf. Acesso em: 10 nov. 2025.

IGENE, Lucky; MARCEL, Sébastien; et al. Face Liveness Detection in the Wild. In: IEEE International Joint Conference on Biometrics (IJCB), 2024. Disponível em: https://publications.idiap.ch/attachments/papers/2024/Igene_IJCB_2024.pdf. Acesso em: 10 nov. 2025.

INSIGHTFACE CONTRIBUTORS. InsightFace: 2D and 3D face analysis project. GitHub, 2025. Disponível em: https://github.com/deepinsight/insightface. Acesso em: 10 nov. 2025.

ISO/IEC. ISO/IEC 30107-3:2023 - Information technology - Biometric presentation attack detection - Part 3: Testing and reporting. 2023. Disponível em: https://www.iso.org/standard/79520.html. Acesso em: 10 nov. 2025.

KEMELMACHER-SHLIZERMAN, Ira; SEITZ, Steven M.; MILLER, Daniel; BROSSARD, Evan. The MegaFace Benchmark: 1 Million Faces for Recognition at Scale. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. Disponível em: http://megaface.cs.washington.edu/. Acesso em: 10 nov. 2025.

KING, Davis E. Dlib-ml: a machine learning toolkit. Journal of Machine Learning Research, v. 10, p. 1755-1758, 2009. Disponível em: https://www.jmlr.org/papers/volume10/king09a/king09a.pdf. Acesso em: 10 nov. 2025.

KOTWAL, Karan; MARCEL, Sébastien. Review of Demographic Fairness in Face Recognition. 2025. arXiv:2502.02309. Disponível em: https://arxiv.org/pdf/2502.02309. Acesso em: 10 nov. 2025.

LI, Yuezun; CHANG, Ming-Ching; LYU, Siwei. In Ictu Oculi: Exposing AI Generated Fake Face Videos by Detecting Eye Blinking. IEEE WIFS, 2018. Disponível em: https://arxiv.org/abs/1806.02877. Acesso em: 7 maio 2026.

MICROSOFT. Detecção de vivacidade facial e Face liveness detection - Face. Microsoft Learn, 2026. Disponível em: https://learn.microsoft.com/pt-br/azure/ai-services/face/concept-face-liveness-detection. Acesso em: 25 mar. 2026.

NGUYEN, Thanh Thi et al. Deep Learning for Deepfakes Creation and Detection: A Survey. Computer Vision and Image Understanding, v. 223, 2022. Disponível em: https://arxiv.org/abs/1909.11573. Acesso em: 7 maio 2026.

NIST. Face Challenges - IARPA Janus Benchmark (IJB-B/IJB-C). 2025. Disponível em: https://www.nist.gov/programs-projects/face-challenges. Acesso em: 10 nov. 2025.

NIST. Face in Video Evaluation (FIVE). 2024. Disponível em: https://www.nist.gov/programs-projects/face-video-evaluation-five. Acesso em: 10 nov. 2025.

NIST. Face Recognition Technology Evaluation (FRTE) - 1:1 Verification. 2023a. Disponível em: https://pages.nist.gov/frvt/html/frvt11.html. Acesso em: 10 nov. 2025.

NIST. Face Recognition Technology Evaluation (FRTE) - 1:N Identification. 2023b. Disponível em: https://pages.nist.gov/frvt/html/frvt1N.html. Acesso em: 10 nov. 2025.

NIST. FATE Part 10: Performance of Passive, Software-Based Presentation Attack Detection (PAD) Algorithms (NISTIR 8491). 2023c. Disponível em: https://nvlpubs.nist.gov/nistpubs/ir/2023/NIST.IR.8491.pdf. Acesso em: 10 nov. 2025.

SCHROFF, Florian; KALENICHENKO, Dmitry; PHILBIN, James. FaceNet: A Unified Embedding for Face Recognition and Clustering. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015. p. 815-823. Disponível em: https://openaccess.thecvf.com/content_cvpr_2015/html/Schroff_FaceNet_A_Unified_2015_CVPR_paper.html. Acesso em: 7 maio 2026.

SHARMA, Deepak; SINGH, S. K.; KUMAR, Arun. A survey on face presentation attack detection mechanisms: hitherto and future perspectives. Visual Computing for Industry, Biomedicine, and Art, v. 6, 2023. Disponível em: https://pmc.ncbi.nlm.nih.gov/articles/PMC10025066/. Acesso em: 25 mar. 2026.

TERHÖRST, Philipp; THOMMEN, Simon; SCHLEPS, Martin; et al. A Comprehensive Study on Face Recognition Biases Beyond Demographics. 2022. Disponível em: https://biometrics.eps.uam.es/fierrez/files/2022_TTS_Biases_Terhorst.pdf. Acesso em: 10 nov. 2025.

TOLOSANA, Ruben; VERA-RODRIGUEZ, Ruben; FIERREZ, Julian; MORALES, Aythami; ORTEGA-GARCIA, Javier. DeepFakes and Beyond: A Survey of Face Manipulation and Fake Detection. Information Fusion, v. 64, p. 131-148, 2020. Disponível em: https://arxiv.org/abs/2001.00179. Acesso em: 7 maio 2026.

WU, Hanzi; ALBIERO, Vítor; KRISHNAPRIYA, K.; KING, Michael; BOWYER, Kevin. Face Recognition Accuracy Across Demographics: Shining a Light Into the Problem. 2022. arXiv:2206.01881. Disponível em: https://arxiv.org/abs/2206.01881. Acesso em: 10 nov. 2025.

ZHANG, Kaipeng; ZHANG, Zhanpeng; LI, Zhifeng; QIAO, Yu. Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks. IEEE Signal Processing Letters, v. 23, n. 10, p. 1499-1503, 2016. Disponível em: https://arxiv.org/abs/1604.02878. Acesso em: 7 maio 2026.

APÊNDICE A - ROTEIRO DE COLETA DE DADOS (CHECKLIST)

Este roteiro foi utilizado para padronizar a captura das imagens em todos os testes, garantindo a consistência das variáveis ambientais.

1. Preparação do Ambiente e Hardware

- Câmera RGB (resolução mínima 1080p/30fps) fixada em tripé.
- Distância do participante em relação à câmera ajustada entre 0,6m e 1,2m.
- Foco travado e balanço de branco calibrado para evitar distorções automáticas.

2. Captura por Cenários (Testes de Robustez)

- **Cenário 1 (Ideal):** Iluminação frontal difusa (aprox. 300 a 500 lux). Sem sombras no rosto.
- **Cenário 2 (Baixa Luz):** Redução intencional da iluminação ambiente para forçar queda do sinal-ruído (SNR).
- **Cenário 3 (Contraluz):** Fonte de luz principal posicionada atrás do participante, gerando subexposição facial.

3. Validação do Sistema de Liveness (PAD Ativo)

- Tentativa de fraude com foto impressa em alta qualidade.
- Tentativa de fraude com vídeo comum reproduzido em tela de smartphone/tablet.
- Tentativa de fraude com vídeo preparado (atacante ciente dos comandos possíveis do desafio-resposta).
- Registro em log dos acertos, falhas de detecção e falsos positivos.

APÊNDICE B - GUIA EXPLICATIVO SOBRE PAD PARA LEITORES NÃO TÉCNICOS

O que é o PAD (Presentation Attack Detection)? Em sistemas de segurança, o PAD atua como uma camada de verificação de vivacidade (liveness). Sem ele, um algoritmo de reconhecimento facial atua apenas como um comparador de aparências, podendo validar o acesso de qualquer mídia estática que contenha o rosto da pessoa autorizada. O PAD garante que existe um ser humano vivo e fisicamente presente no momento da captura.

Como o sistema se defende dos principais ataques? Sistemas robustos procuram por características físicas ou comportamentais que mídias artificiais não conseguem replicar perfeitamente:

- **Fotos impressas:** São superfícies planas (sem profundidade em sensores 3D), não apresentam sinais vitais como o micro-ritmo cardíaco (rPPG), possuem textura gráfica de impressão e refletem a luz de forma não natural. No contexto deste projeto, falham em responder aos comandos de movimento.
- **Vídeos em tela (Replay):** Podem apresentar movimento, mas as telas digitais emitem luz própria, gerando artefatos visuais (cintilação e padrões de *moiré*) detectáveis pela câmera. Além disso, têm grande dificuldade de sincronizar a gravação com desafios de movimento gerados aleatoriamente em tempo real (como piscar ou virar o rosto em uma ordem específica).
- **Máscaras 3D:** Embora possuam profundidade, feitas de materiais como silicone ou látex, possuem uma textura e resposta térmica/espectral diferentes da pele humana e não conseguem reproduzir interações visuais e microexpressões dinâmicas de forma natural.
- **Deepfakes (Mídia Sintética):** Costumam apresentar pequenas incoerências temporais ou espaciais, falhas nos contornos, padrões de piscada anormais ou dessincronização entre movimentos faciais e áudio, deixando rastros e artefatos de alta frequência na imagem final.

Neste trabalho, dada a infraestrutura baseada em câmera RGB, a defesa escolhida implementada foi a interativa (desafio-resposta visual), exigindo a colaboração dinâmica do usuário para neutralizar ataques estáticos.

APÊNDICE C - LISTAGENS DE CÓDIGO

A versão organizada do código-fonte, juntamente com as instruções de instalação, replicação experimental, roteiros de coleta e orientações para continuidade da pesquisa, está disponível no repositório GitHub do projeto: https://github.com/RogihShi/Reconhecimento-Facial-Projeto-de-TCC/tree/main/reconhecimento-facial-tcc. Por questões éticas e de privacidade, o repositório não inclui imagens reais dos participantes nem vídeos utilizados nos testes, pois esses arquivos contêm dados biométricos sensíveis.

Listagem 1 – reconhecedor_facial.py (captura em tempo real, comparação com base local)

```
# -*- coding: utf-8 -*-
import face_recognition
import cv2
import numpy as np
import os
import math
import random

# --- Funções Matemáticas Avançadas ---
def calcular_ear(olho):
    a = math.dist(olho[1], olho[5])
    b = math.dist(olho[2], olho[4])
    c = math.dist(olho[0], olho[3])
    if c == 0: return 0.0
    return (a + b) / (2.0 * c)

def calcular_mar(landmarks):
    if 'top_lip' not in landmarks or 'bottom_lip' not in landmarks: return 0.0

    # Usa a parte interna dos lábios para evitar bugs com a espessura do lábio
    altura_interna_boca = math.dist(landmarks['top_lip'][9], landmarks['bottom_lip'][9])
    largura_boca = math.dist(landmarks['top_lip'][0], landmarks['top_lip'][6])

    if largura_boca == 0: return 0.0
    return altura_interna_boca / largura_boca

def calcular_rotacao_3d(landmarks):
    nariz = landmarks['nose_bridge'][-1]
```

```
    maxilar_esq = landmarks['chin'][0]
    maxilar_dir = landmarks['chin'][-1]

    dist_esq = math.dist(nariz, maxilar_esq)
    dist_dir = math.dist(nariz, maxilar_dir)

    if dist_dir == 0: return 1.0
    return dist_esq / dist_dir

def rosto_frontal(razao):
    return 0.85 <= razao <= 1.15

def rosto_virado_direita(razao):
    return razao < 0.65

def rosto_virado_esquerda(razao):
    return razao > 1.5

# --- Configurações Iniciais ---
caminho_imagens = "imagens_conhecidas"
codificacoes_conhecidas = []
nomes_conhecidos = []

if not os.path.exists(caminho_imagens):
    print(f"Erro: A pasta '{caminho_imagens}' não foi encontrada.")
    exit()

print("Carregando imagens conhecidas...")
for arquivo in os.listdir(caminho_imagens):
    try:
        imagem = face_recognition.load_image_file(f"{caminho_imagens}/{arquivo}")
        codificacoes = face_recognition.face_encodings(imagem)
        if codificacoes:
            codificacoes_conhecidas.append(codificacoes[0])
            nomes_conhecidos.append(os.path.splitext(arquivo)[0])
    except Exception as e:
        pass
```

```
        # --- Variáveis da Máquina de Estados Dinâmica ---
        LIMITE_EAR = 0.22
        LIMITE_MAR = 0.25
        LIMITE_TEMPO = 250

        estado_atual = "AGUARDANDO_FRONTAL"
        desafios_pendentes = []
        indice_desafio_atual = 0

        # Travas de movimento
        olhos_fechados = False
        boca_estava_fechada = False

        frames_no_estado = 0
        frames_sem_rosto = 0

        # Cache de Identidade
        nome_reconhecido = None

        # --- Iniciar Captura ---
        cap = cv2.VideoCapture(0)
        print("\nIniciando sistema de Biometria Antifraude (5 Etapas)... (Pressione 'q' para sair)")

        while True:
            success, frame = cap.read()
            if not success: break

            altura_frame, largura_frame = frame.shape[:2]

            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            frame_pequeno = cv2.resize(frame_rgb, (0, 0), None, 0.25, 0.25)

            locais_rostos = face_recognition.face_locations(frame_pequeno)
            landmarks_lista = face_recognition.face_landmarks(frame_pequeno, locais_rostos)

            # Lógica de Reset de Segurança
            if not locais_rostos:
```

```
            frames_sem_rosto += 1
            if frames_sem_rosto > 10:
                estado_atual = "AGUARDANDO_FRONTAL"
                desafios_pendentes = []
                indice_desafio_atual = 0
                frames_no_estado = 0
                olhos_fechados = False
                boca_estava_fechada = False
                nome_reconhecido = None
        else:
            frames_sem_rosto = 0

        for local_rosto, landmarks in zip(locais_rostos, landmarks_lista):
            y1_orig, x2_orig, y2_orig, x1_orig = [coord * 4 for coord in local_rosto]

            padding_lateral = 30
            padding_topo = 50
            padding_base = 30

            x1 = max(0, x1_orig - padding_lateral)
            y1 = max(0, y1_orig - padding_topo)
            x2 = min(largura_frame, x2_orig + padding_lateral)
            y2 = min(altura_frame, y2_orig + padding_base)

            cor = (0, 0, 255)
            texto_exibicao = "ANALISANDO\nROSTO..."

            if nome_reconhecido is None:
                codificacao_rosto = face_recognition.face_encodings(frame_pequeno, [local_rosto])[0]
                matches = face_recognition.compare_faces(codificacoes_conhecidas, codificacao_rosto, tolerance=0.5)
                distancia_facial = face_recognition.face_distance(codificacoes_conhecidas, codificacao_rosto)

                if True in matches:
                    indice = np.argmin(distancia_facial)
```

```
            if matches[indice]:
                nome_reconhecido = nomes_conhecidos[indice].upper()

        if nome_reconhecido is not None:
            nome = nome_reconhecido
            frames_no_estado += 1

            razao_rosto = calcular_rotacao_3d(landmarks)

            ear_medio = 1.0
            if 'left_eye' in landmarks and 'right_eye' in landmarks:
                                    ear_medio = (calcular_ear(landmarks['left_eye']) +
calcular_ear(landmarks['right_eye'])) / 2.0

            mar_atual = calcular_mar(landmarks)

            if estado_atual == "AGUARDANDO_FRONTAL":
                texto_exibicao = f"{nome}\nOLHE RETO PARA\nA CAMERA"
                cor = (255, 255, 0)

                if rosto_frontal(razao_rosto):
                                    todas_instrucoes = ["PISCAR", "VIRAR_DIREITA",
"VIRAR_ESQUERDA", "ABRIR_BOCA"]

                        # Lógica para criar 5 desafios garantindo que nenhum se repete
consecutivamente
                    desafios_pendentes = [random.choice(todas_instrucoes)]
                    while len(desafios_pendentes) < 5:
                        nova_instrucao = random.choice(todas_instrucoes)
                            if nova_instrucao != desafios_pendentes[-1]: # Só adiciona se for
diferente da última
                            desafios_pendentes.append(nova_instrucao)

                    estado_atual = "EXECUTANDO_DESAFIOS"
                    indice_desafio_atual = 0
                    frames_no_estado = 0

            elif estado_atual == "EXECUTANDO_DESAFIOS":
```

```
            if frames_no_estado > LIMITE_TEMPO:
                estado_atual = "FRAUDE"
            else:
                desafio_ativo = desafios_pendentes[indice_desafio_atual]
                passou_desafio = False

                if desafio_ativo == "PISCAR":
                    if not rosto_frontal(razao_rosto):
                        texto_exibicao = f"{nome}\nMANTENHA O ROSTO\nRETO E
PISQUE [{indice_desafio_atual + 1}/5]"
                    else:
                        texto_exibicao = f"{nome}\nPISQUE [{indice_desafio_atual +
1}/5]"
                    if ear_medio < LIMITE_EAR:
                        olhos_fechados = True
                    elif ear_medio >= LIMITE_EAR and olhos_fechados:
                        passou_desafio = True
                        olhos_fechados = False

                elif desafio_ativo == "VIRAR_DIREITA":
                    texto_exibicao = f"{nome}\nVIRE PARA A\nDIREITA
[{indice_desafio_atual + 1}/5]"
                    if rosto_virado_direita(razao_rosto):
                        passou_desafio = True

                elif desafio_ativo == "VIRAR_ESQUERDA":
                    texto_exibicao = f"{nome}\nVIRE PARA A\nESQUERDA
[{indice_desafio_atual + 1}/5]"
                    if rosto_virado_esquerda(razao_rosto):
                        passou_desafio = True

                elif desafio_ativo == "ABRIR_BOCA":
                    if not rosto_frontal(razao_rosto):
                        texto_exibicao = f"{nome}\nMANTENHA O ROSTO\nRETO E
ABRA A BOCA [{indice_desafio_atual + 1}/5]"
                    else:
```

```
                    texto_exibicao = f"{nome}\nABRA A BOCA [{indice_desafio_atual
+ 1}/5]"

                    if mar_atual < 0.10:
                        boca_estava_fechada = True
                    elif mar_atual > LIMITE_MAR and boca_estava_fechada:
                        passou_desafio = True
                        boca_estava_fechada = False

                cor = (0, 165, 255)

                if passou_desafio:
                    indice_desafio_atual += 1
                    frames_no_estado = 0
                    if indice_desafio_atual >= len(desafios_pendentes):
                        estado_atual = "VALIDADO"

            elif estado_atual == "VALIDADO":
                texto_exibicao = f"ACESSO LIBERADO:\n{nome}"
                cor = (0, 255, 0)

            elif estado_atual == "FRAUDE":
                texto_exibicao = "FRAUDE BLOQUEADA\n(FOTO/VIDEO)"
                cor = (0, 0, 255)

        # --- RENDERIZAÇÃO GRÁFICA ---
        cv2.rectangle(frame, (x1, y1), (x2, y2), cor, 2)

        linhas_texto = texto_exibicao.split('\n')
        altura_linha = 25
        altura_total_fundo = (len(linhas_texto) * altura_linha) + 10
        largura_fundo_minima = max(x2, x1 + 250)

        cv2.rectangle(frame, (x1, y2), (largura_fundo_minima, y2 + altura_total_fundo),
cor, cv2.FILLED)
        cor_fonte = (0, 0, 0) if cor == (255, 255, 0) else (255, 255, 255)

        for i, linha in enumerate(linhas_texto):
```

```
            pos_y = y2 + 22 + (i * altura_linha)
            cv2.putText(frame, linha, (x1 + 5, pos_y), cv2.FONT_HERSHEY_SIMPLEX,
0.65, cor_fonte, 2)

    cv2.imshow("Biometria Antifraude de Alta Seguranca", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Fonte: autoria própria (2026).

Listagem 2 – experimento.py (avaliação de acurácia por cenário de iluminação)

```
# -*- coding: utf-8 -*-
"""
Experimento comparativo justo - Motor DLIB / face_recognition

Objetivo:
- Avaliar reconhecimento facial em cenário FECHADO (closed-set), isto é,
  todas as imagens de teste pertencem a pessoas já cadastradas.
- Usar a MESMA lógica de avaliação do script ArcFace ajustado:
  1) detectar o rosto;
  2) gerar embedding;
  3) encontrar o cadastro mais próximo;
  4) contar acerto somente se o nome previsto == nome esperado.

Observação importante:
- Este script NÃO usa tolerance/threshold para decidir "desconhecido".
- Isso foi removido de propósito para deixar a comparação mais justa com o
  ArcFace, usando rank-1 / nearest neighbor para ambos.
"""

import os
import time
from typing import List, Tuple

import numpy as np
```

```
import face_recognition

# ----------------------------
# CONFIGURAÇÕES
# ----------------------------
PASTA_IMAGENS_CONHECIDAS = "imagens_conhecidas"
PASTA_RAIZ_TESTES = "imagens_teste"
MODELO_DETECCAO_DLIB = "hog"  # "hog" ou "cnn" (cnn é mais pesado e exige suporte adequado)

CENARIOS_DE_TESTE = {
    "Cenário Ideal": os.path.join(PASTA_RAIZ_TESTES, "Ideal"),
    "Baixa Luz": os.path.join(PASTA_RAIZ_TESTES, "Baixa_luz"),
    "Baixa Resolução": os.path.join(PASTA_RAIZ_TESTES, "Baixa_resolu"),
    "Contraluz": os.path.join(PASTA_RAIZ_TESTES, "Contra_luz"),
}

EXTENSOES_VALIDAS = {".jpg", ".jpeg", ".png", ".bmp", ".webp"}


def eh_imagem(nome_arquivo: str) -> bool:
    return os.path.splitext(nome_arquivo)[1].lower() in EXTENSOES_VALIDAS



def extrair_nome_pessoa(nome_arquivo: str) -> str:
    """Extrai o nome da pessoa a partir do nome do arquivo.

    Exemplos:
    - "andre.jpg" -> "andre"
    - "andre_01.jpg" -> "andre"
    - "Higor_Ideal5.jpg" -> "higor"
    """
    stem = os.path.splitext(nome_arquivo)[0].strip().lower()
    return stem.split("_")[0]



def carregar_base_dlib(pasta: str) -> Tuple[List[np.ndarray], List[str]]:
```

```
    codificacoes_conhecidas: List[np.ndarray] = []
    nomes_conhecidos: List[str] = []

    if not os.path.exists(pasta):
        raise FileNotFoundError(f"A pasta '{pasta}' não foi encontrada.")

    print("Carregando base de dados com DLIB / face_recognition...")

    for arquivo in sorted(os.listdir(pasta)):
        if not eh_imagem(arquivo):
            continue

        caminho = os.path.join(pasta, arquivo)
        nome_pessoa = extrair_nome_pessoa(arquivo)

        try:
            imagem = face_recognition.load_image_file(caminho)
                                        locais = face_recognition.face_locations(imagem,
model=MODELO_DETECCAO_DLIB)
            encodings = face_recognition.face_encodings(imagem, locais)

            if not encodings:
                print(f"AVISO: Nenhum rosto detectado em '{arquivo}' (cadastro). Imagem
ignorada.")
                continue

            # Usa apenas o primeiro rosto encontrado
            codificacoes_conhecidas.append(encodings[0])
            nomes_conhecidos.append(nome_pessoa)

        except Exception as e:
            print(f"AVISO: Não foi possível processar '{arquivo}' (cadastro). Erro: {e}")

    if not codificacoes_conhecidas:
        raise RuntimeError("Nenhuma codificação válida foi carregada da base.")

    print(f"Total de embeddings carregados: {len(codificacoes_conhecidas)}")
    print(f"Pessoas da base: {sorted(set(nomes_conhecidos))}")
```

```
    return codificacoes_conhecidas, nomes_conhecidos



def prever_identidade_dlib(codificacao_teste: np.ndarray,
                           codificacoes_conhecidas: List[np.ndarray],
                           nomes_conhecidos: List[str]) -> Tuple[str, float]:
    """Retorna o nome do cadastro mais próximo e a distância euclidiana."""
                distancias = face_recognition.face_distance(codificacoes_conhecidas,
codificacao_teste)
    indice_melhor = int(np.argmin(distancias))
    return nomes_conhecidos[indice_melhor], float(distancias[indice_melhor])



def executar_cenario_dlib(nome_cenario: str,
                          caminho_cenario: str,
                          codificacoes_conhecidas: List[np.ndarray],
                          nomes_conhecidos: List[str]) -> dict:
    print("\n==========================================")
    print(f"Processando: {nome_cenario}...")
    print(f"Pasta: {caminho_cenario}")
    print("==========================================")

    total_testes = 0
    total_acertos = 0
    falhas_deteccao = 0
    erros_identidade = 0
    tempo_inicio = time.time()

    if not os.path.exists(caminho_cenario):
        print(f"AVISO: Pasta não encontrada {caminho_cenario}. Pulando este cenário.")
        return {}

    arquivos = [a for a in sorted(os.listdir(caminho_cenario)) if eh_imagem(a)]
    if not arquivos:
                          print(f"Nenhuma imagem de teste válida foi encontrada em
'{caminho_cenario}'.")
```

```
        return {}

    for nome_arquivo_teste in arquivos:
        nome_esperado = extrair_nome_pessoa(nome_arquivo_teste)
        caminho_imagem_teste = os.path.join(caminho_cenario, nome_arquivo_teste)
        total_testes += 1

        try:
            frame = face_recognition.load_image_file(caminho_imagem_teste)
                                    locais_rostos = face_recognition.face_locations(frame,
model=MODELO_DETECCAO_DLIB)
            codificacoes = face_recognition.face_encodings(frame, locais_rostos)
        except Exception as e:
            print(f"  [FALHA]  Não foi possível ler/processar {nome_arquivo_teste}. Erro:
{e}")
            falhas_deteccao += 1
            continue

        if not codificacoes:
            falhas_deteccao += 1
              print(f"  [ERRO]   {nome_arquivo_teste}: Esperado '{nome_esperado}', mas
NENHUM ROSTO FOI DETECTADO.")
            continue

        codificacao_teste = codificacoes[0]
        nome_previsto, distancia = prever_identidade_dlib(
            codificacao_teste,
            codificacoes_conhecidas,
            nomes_conhecidos,
        )

        if nome_previsto == nome_esperado:
            total_acertos += 1
            print(
                f"  [ACERTO] {nome_arquivo_teste}: Esperado '{nome_esperado}', "
                f"Encontrado '{nome_previsto}' (Distância: {distancia:.4f})"
            )
        else:
```

```
            erros_identidade += 1
            print(
                f"  [ERRO]   {nome_arquivo_teste}: Esperado '{nome_esperado}', "
                f"Encontrado '{nome_previsto}' (Distância: {distancia:.4f})"
            )

    tempo_total = time.time() - tempo_inicio
    acuracia = (total_acertos / total_testes * 100) if total_testes > 0 else 0.0
     taxa_deteccao = ((total_testes - falhas_deteccao) / total_testes * 100) if total_testes
> 0 else 0.0

    print(f"\nResultado do '{nome_cenario}':")
    print(f"  - Total de imagens: {total_testes}")
    print(f"  - Acurácia (rank-1): {acuracia:.2f}% ({total_acertos} de {total_testes})")
    print(f"  - Taxa de detecção: {taxa_deteccao:.2f}%")
    print(f"  - Falhas de detecção: {falhas_deteccao}")
    print(f"  - Erros de identidade: {erros_identidade}")
    print(f"  - Tempo de execução: {tempo_total:.2f} segundos")

    return {
        "acuracia": acuracia,
        "taxa_deteccao": taxa_deteccao,
        "falhas_deteccao": falhas_deteccao,
        "erros_identidade": erros_identidade,
        "total_testes": total_testes,
        "tempo_total": tempo_total,
    }



def main() -> None:
    try:
                                        codificacoes_conhecidas,      nomes_conhecidos      =
carregar_base_dlib(PASTA_IMAGENS_CONHECIDAS)
    except Exception as e:
        print(f"Erro ao carregar a base: {e}")
        return
```

```
    resultados_finais = {}

    print("\n--- INICIANDO TESTES DE ACURÁCIA JUSTA (DLIB) ---")
    print("Critério: closed-set rank-1 (sem threshold de 'desconhecido').")

    for nome_cenario, caminho_cenario in CENARIOS_DE_TESTE.items():
        resultado = executar_cenario_dlib(
            nome_cenario,
            caminho_cenario,
            codificacoes_conhecidas,
            nomes_conhecidos,
        )
        if resultado:
            resultados_finais[nome_cenario] = resultado


print("\n\n============================================================")
    print("--- RESULTADOS FINAIS DLIB (COMPARAÇÃO JUSTA) ---")

print("============================================================")
    for cenario, dados in resultados_finais.items():
        print(
            f"{cenario}: "
            f"Acurácia={dados['acuracia']:.2f}% | "
            f"Detecção={dados['taxa_deteccao']:.2f}% | "
            f"FalhasDet={dados['falhas_deteccao']} | "
            f"ErrosID={dados['erros_identidade']}"
        )

print("============================================================")


if __name__ == "__main__":
    main()
```

Fonte: autoria própria (2026).

Listagem 3 - arcface_retinaface_teste.py (avaliação de acurácia por cenário de iluminação pelo Arcface usando RetinaFace)

```
# -*- coding: utf-8 -*-
"""
Script de Experimento Comparativo: ArcFace + RetinaFace
Versão mais enxuta e mais rápida que a anterior.

Melhorias mantidas:
1) detector fixo: RetinaFace
2) pré-processamento leve para baixa luz
3) cadastro por identidade usando centroide dos embeddings
4) avaliação closed-set rank-1

Critério:
- sem classe "desconhecido"
- acerto somente quando nome previsto == nome esperado
"""

import os
import time
from collections import defaultdict

import cv2
import numpy as np
from deepface import DeepFace
from scipy.spatial.distance import cosine

# ============================================================
# CONFIGURAÇÕES
# ============================================================
CAMINHO_IMAGENS_CONHECIDAS = "imagens_conhecidas"
PASTA_RAIZ_TESTES = "imagens_teste"

MODELO = "ArcFace"
DETECTOR_BACKEND = "retinaface"

# Se quiser deixar mais rápido ainda, coloque False
```

```
USAR_CLAHE = True

CENARIOS_DE_TESTE = {
    "Cenário Ideal": os.path.join(PASTA_RAIZ_TESTES, "Ideal"),
    "Baixa Luz": os.path.join(PASTA_RAIZ_TESTES, "Baixa_luz"),
    "Baixa Resolução": os.path.join(PASTA_RAIZ_TESTES, "Baixa_resolu"),
    "Contraluz": os.path.join(PASTA_RAIZ_TESTES, "Contra_luz"),
}

# ============================================================
# FUNÇÕES AUXILIARES
# ============================================================
def extrair_nome_identidade(nome_arquivo):
    """
    Extrai o nome esperado a partir do arquivo.
    Exemplo:
    Andre_Ideal1.jpg -> andre
    Higor_Baixa_luz10.jpg -> higor
    """
    base = os.path.splitext(os.path.basename(nome_arquivo))[0]
    return base.split("_")[0].lower()


def aplicar_clahe(img_bgr):
    """
    Melhora contraste em imagens escuras sem exagerar muito.
    """
    lab = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2LAB)
    l, a, b = cv2.split(lab)

    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    l2 = clahe.apply(l)

    lab2 = cv2.merge((l2, a, b))
    return cv2.cvtColor(lab2, cv2.COLOR_LAB2BGR)


def gerar_variantes_imagem(img_bgr):
```

```
    """
    Gera poucas variantes para não ficar lento.
    """
    variantes = [img_bgr]

    if USAR_CLAHE:
        try:
            variantes.append(aplicar_clahe(img_bgr))
        except Exception:
            pass

    return variantes


def representar_imagem_retinaface(img_bgr):
    """
    Tenta extrair embedding usando somente RetinaFace.
    Testa a imagem original e, opcionalmente, uma versão com CLAHE.
    Retorna:
      - embedding médio
      - quantidade de tentativas bem-sucedidas
    """
    variantes = gerar_variantes_imagem(img_bgr)
    embeddings = []

    for variante in variantes:
        try:
            reps = DeepFace.represent(
                img_path=variante,
                model_name=MODELO,
                detector_backend=DETECTOR_BACKEND,
                enforce_detection=True,
                align=True,
                normalization="ArcFace",
            )

            if reps and "embedding" in reps[0]:
                embeddings.append(np.array(reps[0]["embedding"], dtype=np.float32))
```

```
            except Exception:
                continue

        if not embeddings:
            return None, 0

        emb_medio = np.mean(np.vstack(embeddings), axis=0)
        return emb_medio, len(embeddings)


    def carregar_imagem_bgr(caminho):
        return cv2.imread(caminho)


    # ============================================================
    # PASSO 1: CADASTRO ROBUSTO POR IDENTIDADE
    # ============================================================
    if not os.path.exists(CAMINHO_IMAGENS_CONHECIDAS):
            print(f"Erro: a pasta '{CAMINHO_IMAGENS_CONHECIDAS}' não foi
encontrada.")
        raise SystemExit(1)

    print("Carregando base de dados com ArcFace + RetinaFace...")
    print(f"Detector backend fixo: {DETECTOR_BACKEND}")
    print(f"CLAHE ativado: {USAR_CLAHE}")

    embeddings_por_pessoa = defaultdict(list)
    falhas_cadastro = []

    for arquivo in sorted(os.listdir(CAMINHO_IMAGENS_CONHECIDAS)):
        caminho = os.path.join(CAMINHO_IMAGENS_CONHECIDAS, arquivo)
        img = carregar_imagem_bgr(caminho)

        if img is None:
            falhas_cadastro.append((arquivo, "imagem inválida"))
            continue

        nome = extrair_nome_identidade(arquivo)
```

```
            emb, tentativas_ok = representar_imagem_retinaface(img)

            if emb is None:
                falhas_cadastro.append((arquivo, "nenhum rosto detectado"))
                continue

            embeddings_por_pessoa[nome].append(emb)
                        print(f"[CADASTRO OK] {arquivo} -> identidade='{nome}' |
tentativas_ok={tentativas_ok}")

        if falhas_cadastro:
            print("\nFalhas no cadastro:")
            for arq, motivo in falhas_cadastro:
                print(f" - {arq}: {motivo}")

        if not embeddings_por_pessoa:
            print("Erro: nenhuma identidade válida foi cadastrada.")
            raise SystemExit(1)

        nomes_conhecidos = []
        centroides_conhecidos = []

        for nome, lista_embs in embeddings_por_pessoa.items():
            centroide = np.mean(np.vstack(lista_embs), axis=0)
            nomes_conhecidos.append(nome)
            centroides_conhecidos.append(centroide)

        print("\nIdentidades cadastradas:")
        for nome, embs in embeddings_por_pessoa.items():
            print(f" - {nome}: {len(embs)} embedding(s) usado(s)")

        # ============================================================
        # PASSO 2: EXECUÇÃO DOS TESTES
        # ============================================================
        resultados_finais = {}

        print("\n--- INICIANDO TESTES DE ACURÁCIA JUSTA (ARCFACE +
RETINAFACE) ---")
```

```
print("Critério: closed-set rank-1 (sem threshold de 'desconhecido').")

for nome_cenario, caminho_cenario in CENARIOS_DE_TESTE.items():
    print("\n==========================================")
    print(f"Processando: {nome_cenario}...")
    print(f"Pasta: {caminho_cenario}")
    print("==========================================")

    if not os.path.exists(caminho_cenario):
        print(f"AVISO: pasta não encontrada. Pulando: {caminho_cenario}")
        continue

    arquivos = sorted(os.listdir(caminho_cenario))
    if not arquivos:
        print(f"Nenhuma imagem encontrada em '{caminho_cenario}'.")
        continue

    total_testes = 0
    total_acertos = 0
    falhas_deteccao = 0
    erros_identidade = 0
    tempo_inicio = time.time()

    for nome_arquivo_teste in arquivos:
        caminho_teste = os.path.join(caminho_cenario, nome_arquivo_teste)
        img = carregar_imagem_bgr(caminho_teste)

        if img is None:
            print(f"  [FALHA]  {nome_arquivo_teste}: imagem inválida.")
            continue

        total_testes += 1
        nome_esperado = extrair_nome_identidade(nome_arquivo_teste)

        emb_teste, tentativas_ok = representar_imagem_retinaface(img)

        if emb_teste is None:
            falhas_deteccao += 1
```

```
                print(f"  [ERRO]   {nome_arquivo_teste}: Esperado '{nome_esperado}', mas
NENHUM ROSTO FOI DETECTADO.")
                continue

            distancias = [cosine(c, emb_teste) for c in centroides_conhecidos]
            idx = int(np.argmin(distancias))

            nome_encontrado = nomes_conhecidos[idx]
            menor_dist = float(distancias[idx])

            if nome_encontrado == nome_esperado:
                total_acertos += 1
                print(
                    f"  [ACERTO] {nome_arquivo_teste}: Esperado '{nome_esperado}', "
                    f"Encontrado '{nome_encontrado}' "
                    f"(Distância Cosseno: {menor_dist:.4f}, tentativas_ok={tentativas_ok})"
                )
            else:
                erros_identidade += 1
                print(
                    f"  [ERRO]   {nome_arquivo_teste}: Esperado '{nome_esperado}', "
                    f"Encontrado '{nome_encontrado}' "
                    f"(Distância Cosseno: {menor_dist:.4f}, tentativas_ok={tentativas_ok})"
                )

        tempo_total = time.time() - tempo_inicio

        if total_testes > 0:
            acuracia = (total_acertos / total_testes) * 100.0
            deteccao = ((total_testes - falhas_deteccao) / total_testes) * 100.0

            resultados_finais[nome_cenario] = {
                "acuracia": acuracia,
                "deteccao": deteccao,
                "falhas_deteccao": falhas_deteccao,
                "erros_identidade": erros_identidade,
                "tempo": tempo_total,
                "total_testes": total_testes,
```

```
                "total_acertos": total_acertos,
            }

            print(f"\nResultado do '{nome_cenario}':")
            print(f"  - Total de imagens: {total_testes}")
            print(f"  - Acurácia (rank-1): {acuracia:.2f}% ({total_acertos} de {total_testes})")
            print(f"  - Taxa de detecção: {deteccao:.2f}%")
            print(f"  - Falhas de detecção: {falhas_deteccao}")
            print(f"  - Erros de identidade: {erros_identidade}")
            print(f"  - Tempo de execução: {tempo_total:.2f} segundos")
        else:
                        print(f"Nenhuma imagem de teste válida foi encontrada em '{caminho_cenario}'.")

    # ============================================================
    # PASSO 3: RESUMO FINAL
    # ============================================================
    print("\n============================================================")
    print("--- RESULTADOS FINAIS ARCFACE + RETINAFACE ---")
    print("============================================================")

    for cenario, res in resultados_finais.items():
        print(
            f"{cenario}: "
            f"Acurácia={res['acuracia']:.2f}% | "
            f"Detecção={res['deteccao']:.2f}% | "
            f"FalhasDet={res['falhas_deteccao']} | "
            f"ErrosID={res['erros_identidade']}"
        )

    print("============================================================")
```

Fonte: autoria própria (2026).