Atena: Sistema Inteligente para Segurança dos Espaços

Gustavo Baraldi dos Santos[1], **Vitória Paula Aparecida da Silva**[1], **Késsia Rita da Costa Marchi**[1], **Daniela Eloise Flôr**[1]

[1]IFPR – Instituto Federal do Paraná - Campus Paranavaí
Rua José Felipe Tequinha, 1400 – Jardim das Nações – Paranavaí – PR – Brasil

20223019575@estudantes.ifpr.edu.br, 20223020244@estudantes.ifpr.edu.br,

kessia.marchi@ifpr.edu.br, daniela.flor@ifpr.edu.br

Abstract. *This paper presents the development of an intelligent security monitoring and automation system based on computer vision and artificial intelligence (AI). The solution offers an affordable system capable of performing facial recognition, basic differentiation between humans and animals, and generating automated reports and notifications. These functionalities ensure greater accuracy in access control and surveillance of indoor and outdoor environments. Furthermore, the work highlights the application of computer vision for real-time interpretation of images and videos, enabling the identification of authorized personnel and detection of anomalies. Finally, it discusses automated report storage functionalities and notification delivery via multiple digital channels, aiming to facilitate faster responses and more effective security strategies.*

Resumo. *Este artigo apresenta o desenvolvimento de um sistema inteligente de monitoramento e automação de segurança baseado em visão computacional e inteligência artificial (IA). A solução oferece um sistema acessível capaz de realizar reconhecimento facial, diferenciar de forma básica humanos de animais e gerar relatórios e notificações automáticas. Tais funcionalidades garantem maior precisão no controle de acesso e na vigilância de ambientes internos e externos. Além disso, o trabalho destaca a aplicação da visão computacional para a interpretação de imagens e vídeos em tempo real, possibilitando a identificação de pessoas autorizadas e a detecção de anormalidades. Por fim, discutem-se as funcionalidades de armazenamento automatizado dos relatórios e o envio de notificações por diferentes canais digitais, com o objetivo de permitir respostas mais rápidas e estratégias de segurança mais eficazes.*

1. Introdução

O avanço da inteligência artificial (IA) tem impulsionado o desenvolvimento de sistemas inteligentes, com aprimoramentos constantes e crescente acesso público a modelos de aprendizado de máquina e visão computacional. Esse cenário favorece a implementação de soluções inovadoras por empresas, *startups* e também em projetos de caráter acadêmico, como os desenvolvidos em cursos técnicos e superiores. A versatilidade dos modelos de IA, como os baseados em reconhecimento facial e detecção de padrões, os torna promissores para aplicações em áreas como controle de acesso, segurança patrimonial, análise comportamental e identificação de objetos ou pessoas em ambientes monitorados.

Sistemas de segurança estão cada vez mais presentes no mercado e oferecem funcionalidades importantes. No entanto, sua implementação em determinados contextos nem sempre são economicamente viáveis ou permitem a personalização desejada ou a integração com algoritmos de inteligência artificial de forma aberta e exploratória, o que motivou a investigação proposta neste projeto.

Este trabalho consiste em um estudo preliminar voltado à avaliação da viabilidade técnica de um sistema de monitoramento baseado em IA, utilizando a plataforma embarcada ESP32-CAM integrada aos modelos YOLOv5 (para detecção de objetos) e DeepFace (para reconhecimento facial). A proposta envolve o uso de ferramentas e bibliotecas gratuitas como Python, Arduino, HTML e Supabase, com o objetivo de automatizar o controle de acesso por meio da identificação de indivíduos previamente cadastrados, gerar relatórios e alertas e explorar soluções de baixo custo aplicáveis a diferentes contextos, especialmente os educacionais, domésticos ou de pequenos empreendimentos.

Apesar das expectativas e projeções, a solução proposta possui limitações inerentes à capacidade de processamento em tempo real do hardware embarcado e ao desempenho em condições que não sejam ideais de iluminação e conectividade. Ainda assim, o projeto busca contribuir para a formação técnica dos estudantes envolvidos e ampliar o repertório de aplicações possíveis com visão computacional no contexto educacional.

2. Fundamentação Teórica

A segurança pública é um dos temas mais debatidos na sociedade brasileira, dada sua relevância para a garantia dos direitos fundamentais do cidadão. No Brasil, a atuação do Estado nessa área enfrenta desafios históricos, como a elevada criminalidade, a desigualdade social e a limitação de recursos [Fórum Brasileiro de Segurança Pública 2025]. A Constituição Federal de 1988 estabelece que a segurança pública é dever do Estado e direito e responsabilidade de todos [Brasil 1988]. Diante desse cenário, torna-se fundamental buscar soluções inovadoras e integradas para responder de forma eficaz às demandas sociais [Fórum Brasileiro de Segurança Pública 2025].

Imerso nessa área, o sistema Detecta de São Paulo integra bancos de dados da polícia militar, além de dados do Detran e do Instituto de Identificação. Trata-se de "Uma Solução de Software, com interface Web, composta por uma infraestrutura de servidores que realizam funções inteligentes de correlacionamento de diversos tipos de eventos de interesse de segurança pública com as informações das bases de dados integradas à solução: Veículos, Pessoas (civil e criminal), Atendimento 190, etc." [Governo do Estado de São Paulo 2025]

Tais soluções não se restringem apenas à esfera policial, mas podem ser aplicadas em ambientes estratégicos de grande circulação, como aeroportos, onde sistemas de reconhecimento facial vêm sendo testados para combater o tráfico e o contrabando.

Embora tecnologias como por exemplo o sistema Detecta tenham trazido avanços importantes, a segurança pública no Brasil ainda enfrenta muitos obstáculos que permitam que essas soluções possam ser aplicadas de forma certa. Cita-se como um dos principais desafios a falta de recursos financeiros e de infraestrutura adequada, que acaba limitando não só o alcance, mas também a manutenção dessas tecnologias de forma adequada. Além disso, a fragmentação que ocorre entre os diferentes órgãos de segurança e a pouca integração entre eles comprometem a aplicação de ações conjuntas.

Esses desafios mostram que, apesar dos avanços tecnológicos, ainda existe um longo caminho a percorrer para que a tecnologia realmente transforme a segurança pública do país. Por isso, é fundamental o desenvolvimento de soluções que sejam acessíveis, integradas e adequada às realidades locais no país.

Logo, o principal objetivo deste trabalho foi implementar um sistema de monitoramento usando ferramentas simples e de baixo custo, integradas a IAs pré treinadas como o Yolo e DeepFace, para possibilitar uma opção viável, acessível e versátil. Foi usada a linguagem Python em programas gratuitos para desenvolver esse projeto junto com o dispositivo de menor custo usado, a ESP32-CAM, com isso abrindo-se um espaço para futuras implementações, levando à uma constante atualização. Com funções como a diferenciação de humano e objetos, automatização do processo de acesso controlado e o envio de relatórios e alertas locais. Logo, oferecendo uma solução que facilite o dia a dia, ajude a prevenir riscos e que traga mais tranquilidade para quem gerencia espaços ou para cidadãos que desejam segurança versátil e acessível em suas casas.

A inteligência artificial está entre as maiores transformações tecnológicas do século XXI. Segundo [Russell and Norvig 2016], seu impacto é comparável ao da eletricidade, transformando a forma como interagimos com dados, imagens e decisões automatizadas. No contexto da segurança, destaca-se a visão computacional, tecnologia que permite aos computadores interpretar imagens em tempo real e extrair informações relevantes do ambiente — como reconhecer rostos, detectar movimentações ou identificar objetos com base em padrões visuais aprendidos. [Szeliski 2010] destaca que a visão computacional permite que os sistemas computacionais não apenas “vejam”, mas compreendam as imagens a partir do contexto, o que impulsiona avanços em áreas como monitoramento inteligente e controle de acesso.

Neste projeto, foram utilizadas duas abordagens complementares de IA, processadas externamente em um computador conectado ao módulo embarcado ESP32-CAM: o YOLOv5 (*You Only Look Once*), voltado ao reconhecimento geral de objetos, e o DeepFace, voltado ao reconhecimento facial de pessoas previamente cadastradas. A escolha dessas tecnologias se deve à sua complementaridade, à compatibilidade com bibliotecas gratuitas em Python, e à sua viabilidade de integração com sistemas de baixo custo.

O YOLOv5 é uma rede neural convolucional para detecção de objetos em tempo real. Essa tecnologia divide a imagem em grades e, em cada célula, identifica a presença de objetos, classificando-os e estimando sua posição com base em coordenadas (x, y, largura, altura). O sistema retorna uma lista de elementos presentes na imagem com suas respectivas classificações e níveis de confiança. Neste trabalho, o YOLOv5 foi empregado para distinguir pessoas de outros elementos, como animais ou objetos fixos, aumentando a precisão do monitoramento mesmo em ambientes simples, foco da solução proposta.

A Tabela 1 apresenta uma comparação de performance entre diferentes versões do YOLO em diversos ambientes de teste, com destaque para a versão YOLOv5, adotada neste trabalho. Os *datasets* utilizados refletem diferentes cenários de aplicação: *Weed's public* representa ambientes com vegetação densa e iluminação natural variável; *Home Lawn* simula um ambiente residencial; *Baseball Field* caracteriza espaços abertos com movimentação e variações de luz; e *Manila grass* apresenta uma superfície homogênea com baixa complexidade visual.

Tabela 1. Comparação das versões do YOLO em diferentes ambientes.

Model	Dataset	P	R	mAP@0.5	mAP@0.5:0.95	Inference (ms)
EfficientDet	'Weeds' public	0.9133	0.9273	0.9426	0.7023	44.3
	Home Lawn	0.5982	0.5149	0.5195	0.4155	52.0
	Baseball Field	0.6138	0.7069	0.6614	0.4136	54.2
	Manila grass	0.6047	0.6954	0.5691	0.4369	50.1
YOLOv5m	'Weeds' public	0.9433	0.9663	0.9772	0.7828	16.2
	Home Lawn	0.6331	0.5272	0.5399	0.4263	19.2
	Baseball Field	0.6856	0.8126	0.7135	0.4716	24.1
	Manila grass	0.6411	0.5433	0.6412	0.5007	18.7
YOLOv6l	'Weeds' public	0.9442	0.9494	0.9747	0.7612	22.8
	Home Lawn	0.7836	0.6446	0.7057	0.5022	32.5
	Baseball Field	0.6098	0.4691	0.5379	0.4108	47.9
	Manila grass	0.5865	0.7571	0.7014	0.5248	26.6
YOLOv7	'Weeds' public	0.9265	0.9627	0.9745	0.7685	16.1
	Home Lawn	0.7118	0.6454	0.7108	0.5209	29.1
	Baseball Field	0.6223	0.7579	0.6379	0.4009	35.6
	Manila grass	0.6549	0.7571	0.6461	0.4614	27.5
YOLOv8l	'Weeds' public	0.9476	0.9610	0.9795	0.8123	34.0
	Home Lawn	0.6567	0.6422	0.6564	0.4721	37.4
	Baseball Field	0.6672	0.6474	0.6459	0.4312	19.7
	Manila grass	0.7635	0.6519	0.7589	0.5296	36.6

*Inference time refers to the average time it takes for the model to detect weeds in a single digital image.

O YOLOv5m demonstrou um excelente equilíbrio entre precisão (P), revocação (R) e tempo de inferência, destacando-se especialmente em ambientes mais realistas, como *Baseball Field* e *Weeds' public*. Esses resultados reforçam sua adequação para sistemas embarcados de monitoramento como o ATENA, que exigem modelos leves, rápidos e eficazes mesmo sob limitações de hardware. Sua escolha visa otimizar a detecção em tempo real, contribuindo para estratégias mais eficientes de vigilância e segurança em contextos urbanos e institucionais.

O DeepFace é uma ferramenta de reconhecimento facial baseada em redes neurais profundas *(Deep Neural Networks - DNNs)*. Redes neurais profundas são arquiteturas complexas de Inteligência Artificial que imitam a estrutura cerebral, utilizando várias camadas ocultas empilhadas para processar dados. Essa profundidade permite que elas aprendam, de forma hierárquica e automática, as características *(features)* mais abstratas e complexas dos dados (como o reconhecimento de um rosto ou a tradução de idiomas), sendo a base do campo de estudo conhecido como Aprendizado Profundo *(Deep Learning)*.

O processo tem início com a aplicação do classificador *HaarCascade*, que detecta rostos em uma imagem por meio da análise de janelas e das chamadas *features*, tal como o contraste entre olhos e bochechas. Em seguida, a imagem da face é recortada, redimensionada e analisada por uma rede que identifica traços únicos como a posição de olhos, boca e nariz. Esses traços são convertidos em vetores de características que permitem a comparação com rostos previamente armazenados. O grau de similaridade pode ser ajustado diretamente no código-fonte, tornando o sistema mais ou menos rigoroso conforme a necessidade.

A Figura 1 apresenta exemplos de configurações adicionais do DeepFace, como predição de idade, emoção, gênero e raça, ilustrando sua flexibilidade para lidar com diferentes expressões faciais, variações de qualidade da imagem, enquadramento e iluminação em tempo real.

Figura 1. Exemplo de configurações adicionais do DeepFace envolvendo as features.

3. Metodologia

A elaboração do sistema proposto foi orientada pelas etapas do Processo de Desenvolvimento de Produtos (PDP), conforme [Rozenfeld et al. 2006], adaptado ao contexto de sistemas embarcados em ambiente acadêmico. Essa abordagem estrutura as atividades desde a identificação do problema até a arquitetura e validação da solução, mesmo em projetos de escopo reduzido. Também foram considerados os critérios de [Brown 2020], que orientam o desenvolvimento de soluções tecnológicas com base na análise do público-alvo, nas funcionalidades esperadas e nos requisitos funcionais e não funcionais. As etapas foram organizadas como segue:

3.1. Geração do conceito

Realizou-se uma análise inicial das demandas do projeto, identificando-se a necessidade de um sistema de controle de acesso por reconhecimento facial com baixo custo e fácil implementação. A ESP32-CAM foi escolhida por ser um microcontrolador com câmera integrada e conectividade Wi-Fi, compatível com bibliotecas amplamente utilizadas. Também foram pesquisadas soluções semelhantes e referências em segurança patrimonial, reforçando a importância da acessibilidade e da usabilidade.

3.2. Planejamento e identificação

Foram definidos os requisitos funcionais (como a captura e envio de imagens, o reconhecimento facial e o registro de eventos) e os não funcionais (como confiabilidade, tempo de resposta e simplicidade da interface). A arquitetura adotada foi do tipo cliente-servidor, na qual a ESP32-CAM envia imagens por protocolo HTTP para um servidor em Python, que realiza o reconhecimento facial com a biblioteca face-recognition. Os dados de usuários são armazenados em um banco Supabase, e uma interface HTML foi desenvolvida para permitir o controle básico do sistema, como ativar ou desativar o reconhecimento.

3.3. Arquitetura e Operação

Com base nos requisitos funcionais e não funcionais definidos na etapa anterior, a arquitetura geral do sistema foi delineada de forma a permitir uma operação eficiente mesmo com hardware de baixo custo. A solução adotada segue o modelo cliente-servidor: a placa ESP32-CAM realiza a captura de imagens e as envia, via protocolo HTTP, para um servidor local programado em Python, onde é realizado o processamento das imagens utilizando as bibliotecas YOLOv5 (para detecção geral de objetos) ou DeepFace (para reconhecimento facial).

Durante os testes, a ESP32-CAM operou em resolução de 680×480 pixels, que apresentou bom equilíbrio entre desempenho e qualidade de imagem. Observou-se leve aquecimento do módulo após períodos prolongados de uso, contudo, sem comprometer a estabilidade do sistema. Para futuras versões, pretende-se avaliar o uso de dissipadores de calor ou microcoolers, a fim de permitir o funcionamento contínuo em resoluções superiores e sob maior demanda de processamento.

No servidor, as imagens recebidas são comparadas com registros armazenados previamente em um banco de dados Supabase. Caso haja correspondência, o acesso é validado; do contrário, o sistema registra o evento como tentativa não autorizada. O banco também armazena as ocorrências para consultas futuras, favorecendo a rastreabilidade dos acessos. A interface foi desenvolvida em HTML, com comandos básicos de ativação, desligamento e ajuste do intervalo de verificação.

A Figura 2 destaca a integração entre os componentes cliente-servidor para uma operação eficiente em hardware de baixo custo.

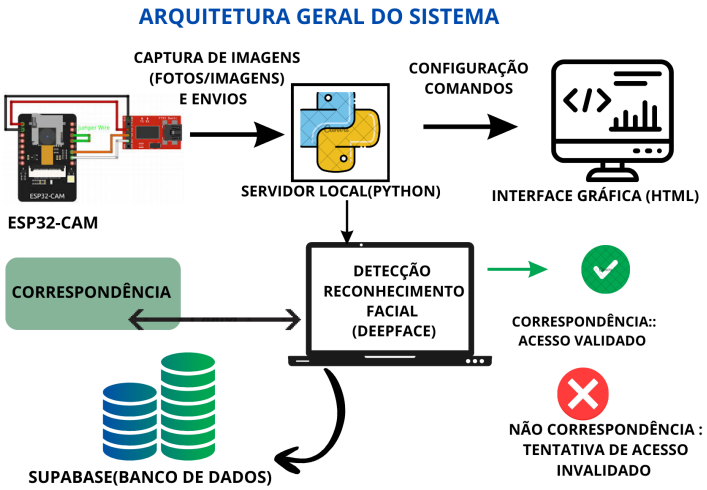

Figura 2. Arquitetura geral do sistema de controle de acesso com ESP32-CAM, servidor Python e IA integrada.

Após a integração dos módulos, a operação do sistema foi testada em ambiente controlado, considerando diferentes condições de iluminação, distâncias da câmera, ângulos faciais e perfis cadastrados e não cadastrados. Foram observados o tempo médio de resposta, a taxa de acerto nas identificações, a estabilidade da comunicação entre os dispositivos e comportamento do sistema diante de falhas de conexão.

Mesmo com limitações, como a redução de precisão em ambientes com muita luz ou sombra acentuada, o protótipo demonstrou viabilidade de uso em cenários simples de controle de acesso, oferecendo resultados promissores para futuras iterações.

4. Articulação entre as Tecnologias Empregadas

O sistema de controle de acesso desenvolvido adota uma arquitetura do tipo cliente-servidor. O dispositivo embarcado ESP32-CAM, escolhido por sua integração entre microcontrolador e câmera, é responsável pela captura das imagens e pela transmissão, via protocolo HTTP, ao servidor local. A conectividade Wi-Fi e o suporte à programação via Arduino IDE tornam a ESP32-CAM uma opção viável para soluções de baixo custo.

O servidor foi implementado em linguagem Python, com uso das bibliotecas OpenCV (para manipulação de imagens), Flask (para comunicação via API) e face-recognition (para reconhecimento facial). As imagens recebidas são comparadas com registros previamente cadastrados no banco de dados Supabase. Quando uma correspondência é identificada, o sistema autentica o usuário. Caso contrário, o acesso é negado e o evento é armazenado para rastreabilidade futura.

A interface web, desenvolvida em HTML, permite ao operador executar comandos básicos, como ativar/desativar o sistema e ajustar o intervalo de varredura. A lógica de operação permite que o sistema alterne entre os modos de IA conforme definido no código. Quando o modo yolo é ativado, a IA YOLOv5 detecta elementos visuais na imagem e os rotula com caixas delimitadoras, atualizadas em tempo real para simular continuidade de vídeo. Já no modo DeepFace, a detecção facial utiliza o algoritmo *HaarCascade* para identificar feições e redes neurais convolucionais (CNNs) para realizar a verificação por similaridade vetorial.

A Figura 3 ilustra a lógica que ativa a detecção de objetos utilizando a biblioteca YOLOv5, configurando a alternância de quadros com base no modo selecionado.

```
if modo_ia == "yolo":
    # ---------------- YOLO ----------------
    results = model(frame, verbose=False)[0]
    last_results = []
    for box in results.boxes:
        x1, y1, x2, y2 = map(int, box.xyxy[0])
        conf = float(box.conf[0])
        cls = int(box.cls[0])
        label = model.names[cls]
        last_results.append((x1, y1, x2, y2, label, conf))
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
        cv2.putText(frame, f'{label} {conf:.2f}', (x1, y1 - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
```

Figura 3. Código para ativação da detecção com YOLOv5.

Esse trecho permite que a detecção de objetos seja executada dinamicamente, atualizando os quadros capturados para simular a continuidade do vídeo, especialmente em contextos com movimentação na cena.

A Figura 4 apresenta o trecho do código responsável pela ativação do reconhecimento facial utilizando a biblioteca DeepFace, que realiza a detecção e comparação de rostos com base nas imagens previamente cadastradas.

```
            try:
                face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
                gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                faces = face_cascade.detectMultiScale(gray, 1.3, 5)

                for (x, y, w, h) in faces:
                    rosto_crop = frame[y:y+h, x:x+w]
                    if rosto_crop.size == 0:
                        continue

                    resultado = DeepFace.represent(
                        rosto_crop,
                        model_name="ArcFace",
                        enforce_detection=False
                    )[0]
                    embedding_teste = resultado["embedding"]

                    global ultimo_nome
                    ultimo_nome = "Desconhecido"
                    maior_similaridade = 0

                    for usuario in rostos_cadastrados:
                        similarity = np.dot(usuario["embedding"], embedding_teste) / (
                            np.linalg.norm(usuario["embedding"]) * np.linalg.norm(embedding_teste)
                        )
                        if similarity > 0.65 and similarity > maior_similaridade:
                            ultimo_nome = usuario["nome"]
                            maior_similaridade = similarity

                    cv2.putText(frame, f"{ultimo_nome} ({maior_similaridade:.2f})",
                                (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
                                (0, 255, 0) if ultimo_nome != "Desconhecido" else (0, 0, 255), 2)
                    cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

            except Exception as e:
                logging.error("[DeepFace] Erro ao analisar frame: %s", e)

        last_detection_time = now
        last_frame = frame.copy()

    elif last_frame is not None:
        frame = last_frame.copy()

    ret, buffer = cv2.imencode('.jpg', frame)
    if not ret:
        continue

    frame_bytes = buffer.tobytes()
    yield (b'--frame\r\n'
           b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')

stream.release()
```

Figura 4. Código para ativação do reconhecimento facial com DeepFace.

Esse bloco condicional ativa o modelo DeepFace e executa a detecção facial por meio do *HaarCascade*. A comparação do vetor gerado com os dados armazenados permite classificar a imagem como autorizada ou desconhecida, conforme o grau de semelhança definido no sistema. A Figura 5 ilustra a montagem do hardware baseada na ESP32-CAM, utilizada durante os testes em ambiente controlado para validação do sistema.

A escolha da ESP32-CAM se deu por sua integração entre microcontrolador e câmera, além da compatibilidade com comunicação Wi-Fi, o que a torna adequada para aplicações embarcadas de reconhecimento visual com baixo custo. A montagem priorizou simplicidade e reutilização de componentes de fácil acesso.

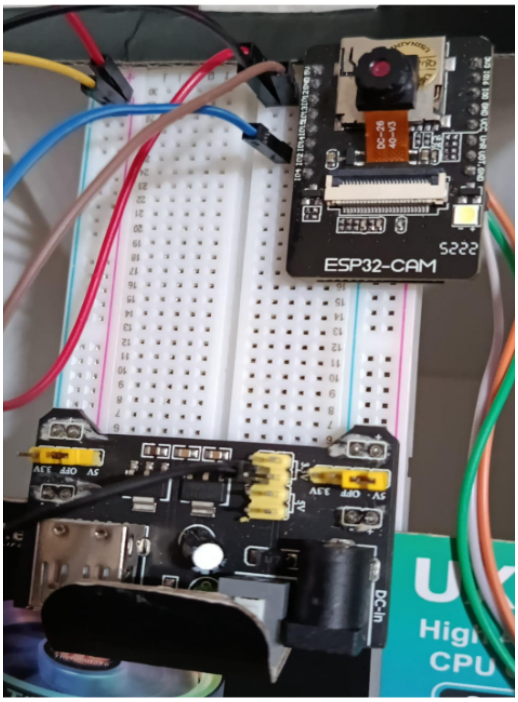

Figura 5. Montagem do protótipo Atena com ênfase na ESP32-CAM.

5. Resultados e Discussão

Os testes realizados com o protótipo demonstraram que a placa ESP32-CAM apresenta aquecimento progressivo quando mantida ativa por longos períodos, especialmente em resoluções mais altas. Considerando a proposta de uso em ambientes simples e controlados, optou-se por operar com configurações de vídeo intermediárias, suficientes para a identificação facial e que reduzem o desgaste do hardware.

Em relação ao desempenho das inteligências artificiais, observou-se que o modelo YOLOv5 obteve resultados satisfatórios para a detecção e rotulagem de objetos em áreas internas e com iluminação moderada. No entanto, situações de variação luminosa e monitoramento contínuo podem interferir na precisão da detecção. Já o DeepFace apresentou variações de desempenho de acordo com o valor de similaridade configurado no código. Utilizando um limiar de 0.80, o sistema conseguiu reconhecer rostos cadastrados com boa precisão, especialmente quando posicionados em ambientes com iluminação adequada.

Além disso, foi realizado um estudo específico para ajustar os parâmetros de detecção facial, com o objetivo de equilibrar desempenho, precisão e velocidade de processamento. Esses ajustes foram baseados em testes práticos, que consideraram taxas de erro como falsos positivos e falsos negativos em diferentes situações de captura.

Inicialmente, o processo envolveu testes empíricos com diferentes rostos incluindo o do próprio desenvolvedor, de familiares e imagens aleatórias obtidas na internet,totalizando dez rostos, a fim de observar o comportamento do algoritmo diante de variações faciais e luminosas. Durante esses testes, o parâmetro de tolerância de similaridade foi ajustado manualmente, observando que valores menores tornavam o sistema mais permissivo, enquanto valores maiores aumentavam a precisão, exigindo correspondência quase exata entre as feições analisadas.

Também foram consideradas variáveis ambientais que influenciam diretamente a eficiência do reconhecimento, como intensidade luminosa, distância entre o indivíduo e a câmera, além do ângulo de enquadramento. Observou-se que, com a resolução de 680×480 pixels, o sistema apresentou melhor desempenho em distâncias de até aproximadamente um metro e meio, demonstrando que a qualidade e a proximidade da imagem impactam diretamente a confiabilidade da detecção.

Com base nesses testes, encontramos um ponto de equilíbrio na configuração, que garantiu precisão sem deixar o sistema lento. Além disso, o estudo deixou claro que os parâmetros precisam ser revisados de tempos em tempos, especialmente em lugares com mudanças de iluminação ou múltiplos usuários.

O tempo médio de reconhecimento facial com DeepFace, nas condições de teste, foi de aproximadamente 5 segundos. O processamento da IA foi realizado em um computador com processador Intel i5-10400f, 16 GB de RAM e SSD de 240 GB, utilizando o ambiente Visual Studio Code com linguagem Python. A arquitetura modular do sistema permite ao usuário configurar o intervalo entre reconhecimentos faciais, definido nesta versão entre 0,1 e 10 segundos, garantindo maior controle sobre o consumo de recursos.

A interface desenvolvida para o sistema oferece interações simples com o usuário, como a ativação e desativação dos modos de reconhecimento, ajuste do intervalo de detecção e visualização em tempo real da imagem processada. Na Figura 6, observa-se o modo YOLOv5 em funcionamento, com a identificação de elementos na cena por meio de quadrados verdes e rótulos com o nome do objeto e a confiabilidade da predição. Nesse exemplo, a IA identificou um indivíduo (*person*) e uma cadeira (*chair*), com valores de confiança de 0,58 e 0,32, respectivamente.

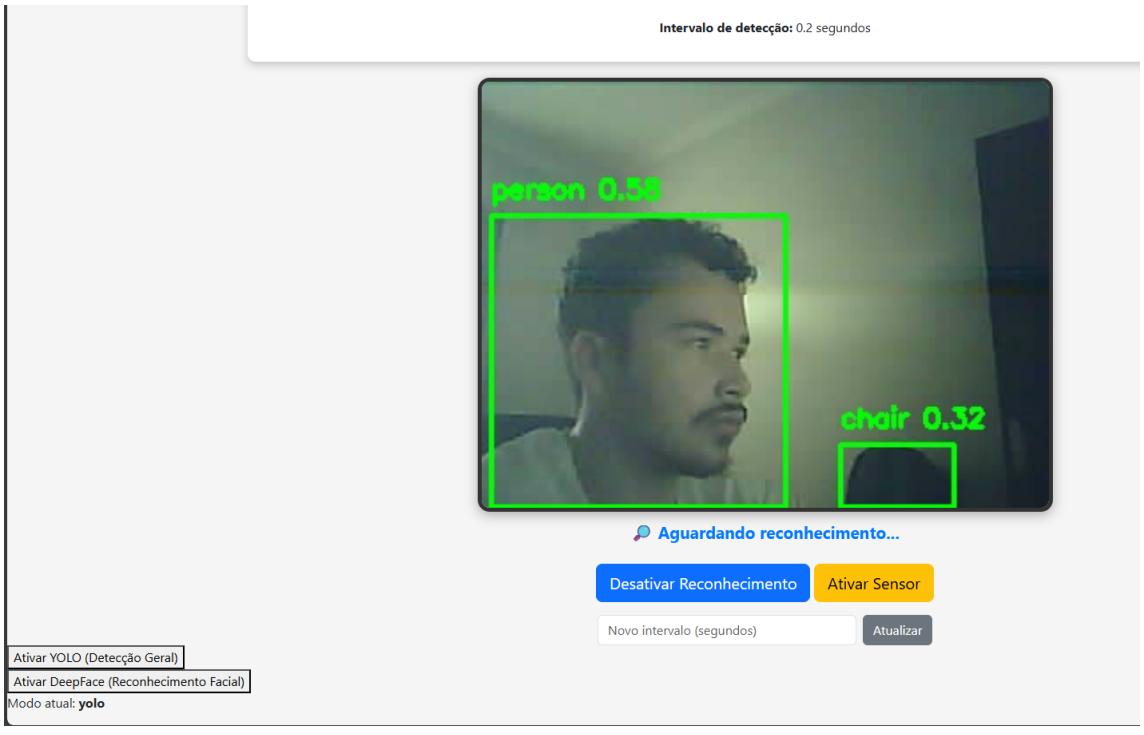

Figura 6. Interface do sistema no modo YOLOv5, com detecção de múltiplos elementos e indicação de confiabilidade.

Já na Figura 7, o protótipo do sistema apresenta o funcionamento do modo DeepFace, no qual o rosto de um indivíduo previamente cadastrado é reconhecido. O sistema retorna o nome correspondente e um valor de similaridade (neste caso, 0,98), evidenciando um bom desempenho do reconhecimento facial em condições favoráveis de iluminação e posicionamento.

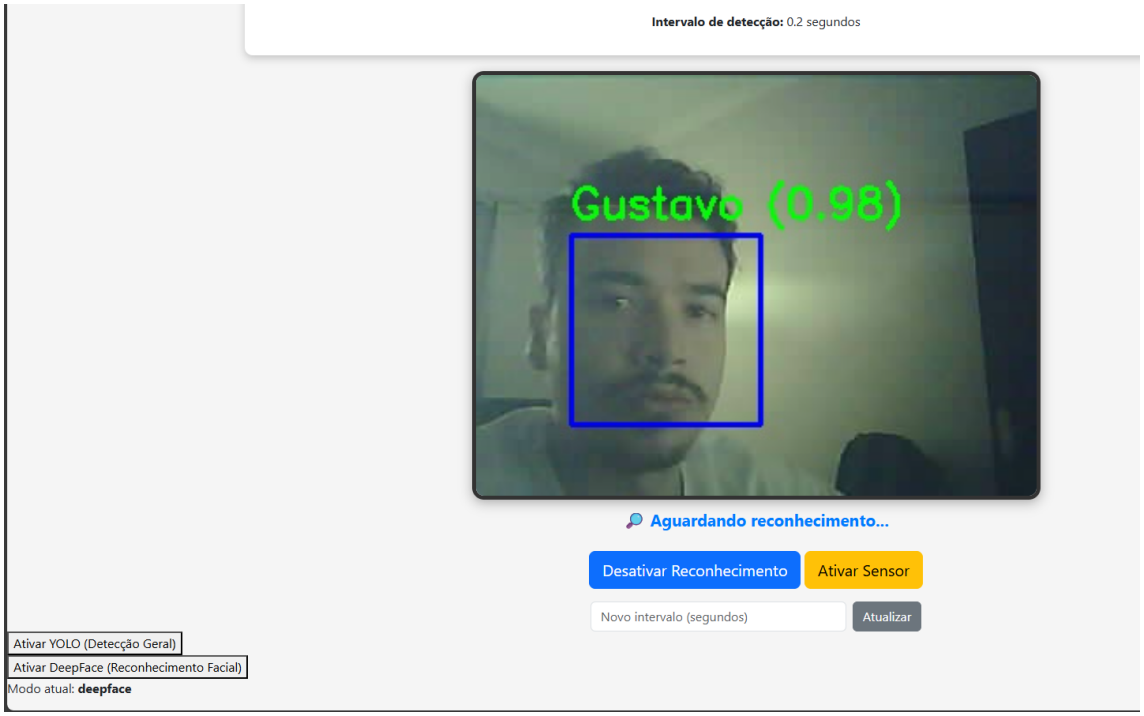

Figura 7. Interface do sistema no modo DeepFace, com identificação de indivíduo previamente cadastrado e valor de similaridade.

6. Considerações finais

O desenvolvimento do sistema de controle de acesso proposto neste trabalho demonstrou ser tecnicamente viável, mesmo com o uso de um hardware embarcado de baixo custo como a ESP32-CAM. A integração com ferramentas livres, como Python, Supabase e bibliotecas de IA (YOLOv5 e DeepFace), permitiu a implementação de funcionalidades típicas de sistemas de segurança inteligentes, com desempenho aceitável em condições controladas.

Os resultados obtidos evidenciam a compatibilidade entre custo e funcionalidade, demonstrando que é possível construir soluções acessíveis e eficientes para monitoramento e autenticação visual em pequenos ambientes residenciais, educacionais, comerciais e institucionais. Além disso, o projeto atende os Objetivos de Desenvolvimento Sustentável (ODS), em especial o objetivo 9 (Indústria, Inovação e Infraestrutura), ao explorar o potencial de tecnologias abertas e acessíveis, e o objetivo 4 (Educação de Qualidade), ao fornecer uma solução replicável, expansível e de alto valor didático para o ensino técnico e superior.

7. Trabalhos Futuros

Como desdobramentos do presente estudo, é possível explorar a adição de sensores de movimento ao sistema, com o objetivo de acionar a câmera apenas quando houver atividade detectada no ambiente, otimizando o consumo energético e ampliando a vida útil do dispositivo.

Outra possibilidade seria realizar o processamento das imagens na nuvem (Cloud Computing), aproveitando a alta velocidade de *upload* atual. Isso permitiria maior poder de processamento a custo reduzido, ampliando a escalabilidade do sistema. Também se considera a implementação de autenticação multifatorial, o aprimoramento da interface gráfica para facilitar o uso por diferentes públicos e a realização de testes em ambientes mais complexos, com maior fluxo de pessoas.

Por fim, o sistema poderá ser explorado em projetos de ensino, tanto para fins de aprendizagem sobre prototipação de sistemas embarcados com IA quanto para discussão de temáticas relacionadas à segurança digital, ética no uso de reconhecimento facial e acessibilidade tecnológica.

Referências

Brasil (1988). *Constituição da República Federativa do Brasil de 1988*. Senado Federal, Brasília.

Brown, T. (2020). *Design Thinking: uma metodologia poderosa para decretar o fim das velhas ideias*. Alta Books.

Fórum Brasileiro de Segurança Pública (2025). *Anuário Brasileiro de Segurança Pública 2024–2025*. Fórum, São Paulo.

Governo do Estado de São Paulo (2025). Cartilha de adesão ao sistema detecta – v3.0.

Rozenfeld, H., Forcellini, F. A., Amaral, D. C., Toledo, J. C. d., Silva, S. L. d., Allipprandini, D. H., and Scalice, R. K. (2006). *Gestão de desenvolvimento de produtos: uma referência para melhoria de processo*. Saraiva, São Paulo.

Russell, S. and Norvig, P. (2016). *Inteligência Artificial*. Pearson, 3 edition.

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.