Revista PET BRASIL

PETRECOGNITION: SISTEMA DE ACESSO INTELIGENTE

PETRECOGNITION: INTELLIGENT ACCESS SYSTEM

Felipe Meier Martins *(fma.peteel@gmail.com)*;
João Pedro de Souza Fernandes Branco *(jpb.peteel@gmail.com)*.
Universidade Federal de Santa Catarina

Dr. André Luís Kirsten
kirsten.andre@ufsc.br
Universidade Federal de Santa Catarina

Artigo

Resumo:
Este artigo trata do projeto PETrecognition, uma inovação proposta pelos bolsistas para acesso à sala PET. Desenvolvido em Python e utilizando bibliotecas como: Dlib, Face Recognition, OpenCV e Numpy, o projeto incorpora um Arduino Uno para controlar o acesso à sala por meio de redes neurais convolucionais. O reconhecimento facial combina câmeras, análise de imagens e Inteligência Artificial IA para capturar e comparar características únicas, como a distância entre olhos e largura do nariz. Tem várias aplicações, incluindo segurança, controle de acesso e verificação de identidade em setores financeiros e telecomunicações, mas enfrenta controvérsias relacionadas à privacidade, discriminação e erros de identificação.

Palavras-chave: Reconhecimento Facial; Redes Neurais; Controle de Acesso.

Abstract:
This article deals with the PETrecognition project, an innovation proposed by the scholarship holders for access to the PET room. Developed in Python and using libraries such as: Dlib, Face Recognition, OpenCV and Numpy, the project incorporates an Arduino Uno to control access to the room through convolutional neural networks. Facial recognition combines cameras, image analysis and AI Artificial Intelligence to capture and compare unique features such as eye distance and nose width. It has multiple applications, including security, access control and identity verification in financial and telecommunications sectors, but faces controversies related to privacy, discrimination and misidentification.

Keywords: Facial Recognition; Neural Network; Access Control; Deep Learning.

1. Introdução

Uma das metodologias do grupo PET EEL é a integração dos novos bolsistas em projetos internos, de forma a atuarem em uma área de seu interesse. Sendo assim, a área escolhida pelos autores para trabalhar com o projeto interno foi a de reconhecimento facial. Essa escolha deu-se procurando um equilíbrio de interesses entre programação e aprendizado de máquina (*machine learning*).

O reconhecimento facial é uma área de pesquisa em constante evolução, com aplicações abrangendo desde segurança até interação humano-computador. Com base nesse contexto dinâmico, o presente projeto visa capacitar os novos bolsistas, proporcionando-lhes aprendizado em programação em linguagem *Python* e desenvolvimento de um modelo de reconhecimento facial.

As tecnologias de reconhecimento facial trazem consigo uma série de aprimoramentos da eficiência em áreas como processos de identificação, controle de acesso e segurança em geral. Além disso, a implementação de um modelo de reconhecimento facial promoverá praticidade e modernidade no contato entre acadêmicos e pessoas interessadas na equipe.

Outra motivação essencial para a execução deste projeto é utilizar o modelo de reconhecimento facial como uma estratégia de divulgação do PET EEL, Programa de Educação Tutorial da Engenharia Elétrica da UFSC (Universidade Federal de Santa Catarina).

2. Desenvolvimento

Nessa seção se abordará o desenvolvimento do projeto, analisando os procedimentos gradativamente, a partir da etapa de estruturação do código e programação, até a etapa de teste no circuito já existente da fechadura eletrônica do escritório.

2. 1. Programação em Python

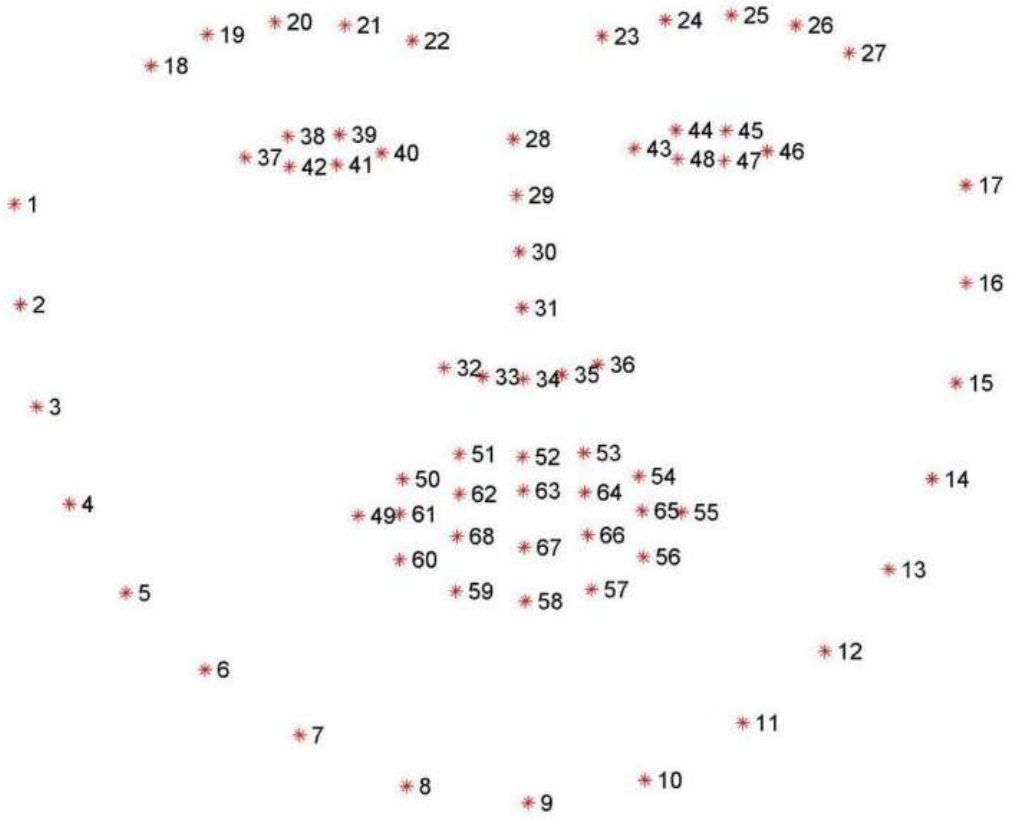
Para dar início ao projeto, se pesquisou sobre os materiais a serem estudados e utilizados. Em seguida, estudou-se um curso sobre detecção e reconhecimento facial ministrado por Jones Granatyr, disponível na plataforma de cursos Udemy (GRANATYR, Jones; ALVES, Gabriel, 2023). Este treinamento proporcionou um aprofundamento no conceito de redes neurais, visão computacional e técnicas para reconhecimento facial.

Para a parte do código, o ambiente utilizado foi a IDE do Pycharm, na qual várias bibliotecas foram fundamentais para viabilizar o projeto, incluindo Dlib, Face Recognition, OpenCV e Numpy.

Na estruturação do código, dividiu-se o trabalho em três arquivos: um para o codificação da base de fotos (database), extraindo 68 pontos/coordenadas de interesse na face (landmarks), com a

ajuda da biblioteca Dlib (Figura 1), esses pontos são identificados a partir de modelo pré-treinado de rede neural onde foi usado o dataset iBUG 300-W, dataset corresponde às amostras iniciais que auxiliam o treinamento de um algoritmo para o aprendizado de máquina. O segundo arquivo foi designado para processamento do tamanho do frame, com a finalidade de se ter um melhor processamento dos dados em um tamanho de frame pré determinado e outro para analisar cada frame e indicar se houve reconhecimento. Realizou-se essa divisão como forma de organizar o desenvolvimento do projeto e os códigos envolvidos.

Figura 1. 68 Pontos de extração de pontos de interesse para detecção facial – Dlib



Fonte: Adaptado de MEIJERINK, Carlijn, 2021. “Facial landmark detection under challengingconditions"

Nesse primeiro arquivo, aplicou-se uma interação para o código abrir o arquivo de fotos e ler foto a foto, devido à necessidade de um conjunto de imagens para o reconhecimento facial ter mais sucesso e segurança. Assim, obteve-se a descrição facial de cada imagem atribuindo o nome do arquivo à pessoa que deve ser identificada, gerando um arquivo do tipo pickle com uma array de codificação de informações.

O segundo arquivo implementou-se para processar o tamanho do frame para o melhor desempenho do algoritmo para reconhecer o rosto pela webcam.

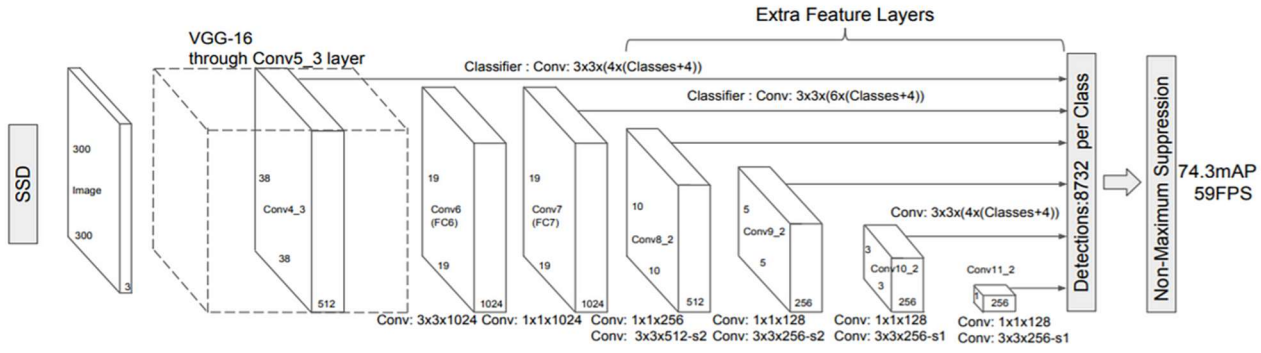
Por fim, o terceiro arquivo é responsável por analisar cada frame: sua primeira tarefa é analisar se algum rosto é detectado. A detecção é feita a partir de uma rede neural convolucional, a partir do método Single Shot Multibox Detector (SSD) (LIU, Wei. et al., 2016).

O SSD tem como base uma rede neural convolucional que no caso utilizamos a VGG-16. Por meio da Figura 2, vemos que uma imagem, frame da câmera, já formatada para os parâmetros padrões do algoritmo de 300 x 300 pixels com 3 canais de de cores, o RGB (red, green, blue), representada pelo primeiro retângulo, é introduzida na rede VGG-16, representada pelos demais retângulos e nomeadas por Conv.

Adiciona-se estrutura auxiliares na rede para detectar características como o Bouding Boxes, método capaz de extrair coordenadas de localização para geração de retângulos, indicando a detecção e realizando a classificação do objeto para detecção em um passo da rede neural. Nesse algoritmo é utilizado uma técnica de regressão chamada Multibox, a qual é capaz de retornar valores numéricos previstos para gerar as localizações dos bounding boxes.

E no final, se recebe as detecções, além de uma técnica para seleção da melhor detecção, ilustrada por Non-Maximum Suppression. Em relação a acurácia do método, constata-se que a detecção ocorre, em um input de 300x300 pixels em 74,3% mAP (mean average precision), que decorre da precisão média da qualidade de uma classificação de multi-classes, já um modelo com entrada de 512x512 pixels é observado uma acurácia maior de 76,9% mAP, porém é preferível o uso da primeira entrada citada, visto que a segundo necessita de um tamanho de frame maior, o que auxilia tanto em questões de velocidade de processamento, quanto em acurácia.

Figura 2. Single Shot Detector (SSD)



Fonte: LIU, Wei. "SSD: Single Shot MultiBox Detector"

Em caso afirmativo, o código analisa se o rosto em questão está cadastrado na base de fotos ou não. Essa análise ocorre a partir do cálculo da distância entre a array de códigos dos rostos

cadastrados em relação aos detectados pela webcam, extraindo o arquivo com melhor aproximação com o que é reconhecido com a webcam.

Denotado pela função do numpy np.argmin (Figura 3), é possível registrar o índice que corresponde ao valor mínimo. Essa distância "threshold" (cuja tradução mais próxima do inglês é limite) pode ser determinada por uma tolerância, de maneira que não seja muito alta, para que qualquer pessoa não cadastrada seja reconhecida, e nem muito baixa, para que uma pessoa cadastrada não seja reconhecida. A partir disso, conclui-se o código do reconhecimento facial.

Um exemplo de função oriunda das bibliotecas mencionadas acima, essencial para a realização do projeto é np.argmin do Numpy (Figura 3). Esta função tem a capacidade de achar a imagem dentro do database que mais se assemelha com a imagem capturada pelo frame da câmera. Em seguida, calcula a média da distância entre as duas imagens, denominado de perda.

Sob essa perspectiva, obtendo a diferença entre a imagem mais próxima com o frame, torna-se possível comparar essa distância com um threshold (limite), possibilitando definir valores numéricos de referência para a tolerância, de maneira que não seja muito alta, para que uma pessoa não cadastrada seja reconhecida, e nem muito baixa, para que uma pessoa cadastrada não seja reconhecida. A partir disso, conclui-se o código do reconhecimento facial.

Figura 3. Função *recognize_faces*

```
def recognize_faces(image, list_encodings, list_names, resizing=0.25, tolerance=0.2): #função para reconhecimento facial
    image = cv2.resize(image, (0,0), fx=resizing, fy=resizing)

    img_rgb = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    face_locations = face_recognition.face_locations(img_rgb)
    face_encodings = face_recognition.face_encodings(img_rgb, face_locations)
    name = "Not identified"
    conf_values = []

    for encoding in face_encodings:
        face_distances = face_recognition.face_distance(list_encodings, encoding)
        best_match_index = np.argmin(face_distances) #imagem mais próxima do data base

        if face_distances[best_match_index] < tolerance: #caso a distancia da imagem <tolerância
            name = list_names[best_match_index] #incluir a lista de nomes em nome
        conf_values.append(face_distances[best_match_index]) #configuração para ter a lista de valores de confiança

    face_locations = np.array(face_locations)
    face_locations = face_locations / resizing
    return face_locations.astype(int), name, conf_values
```

Fonte: Arquivo do grupo PET

2. 2. Integração do Hardware

Uma vez o reconhecimento funcionando, começaram-se os estudos e testes para o acionamento e integração do relé acoplado à fechadura eletrônica.

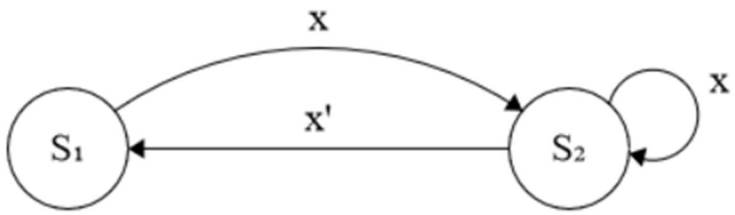
Utilizou-se a biblioteca pyfirmata para a interface do código em Python no microcontrolador Arduino, possibilitando a instrução dos comandos do microcontrolador a partir do código em Python, a partir do protocolo Firmata na IDE do Arduino.

Com essa etapa concluída, precisou-se apenas enviar o sinal de acionamento para o relé quando uma face era reconhecida. O papel desse componente no projeto é justamente abrir o circuito da fechadura eletrônica quando ocorre o reconhecimento, desmagnetizando a chapa da porta e permitindo a passagem de pessoas.

O relé permanecia acionado enquanto o rosto era reconhecido, uma vez que o reconhecimento era interrompido, o circuito da fechadura eletrônica se fechava novamente, trancando a porta.

Pode-se verificar o funcionamento geral do projeto no diagrama de máquina de estado finita (Figura 4), sendo S1 relacionada a fechadura trancada, S2 a fechadura destrancada e o x é o comando de acionamento do relé.

Figura 4. Diagrama do circuito digital, máquina de estado finita

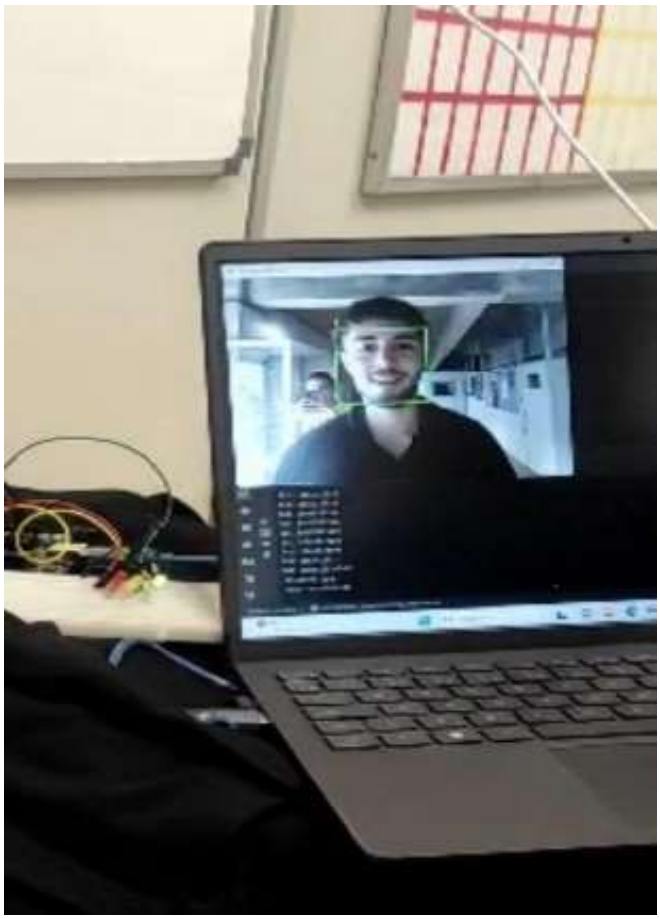


Fonte: Arquivo do grupo PET

2. 3. Integrando o projeto no escritório

Com o acionamento do relé funcionando, necessitou-se apenas a implementação do projeto no circuito da fechadura eletrônica do escritório (Figura 5).

Figura 5. Primeiras implementações na porta do escritório.



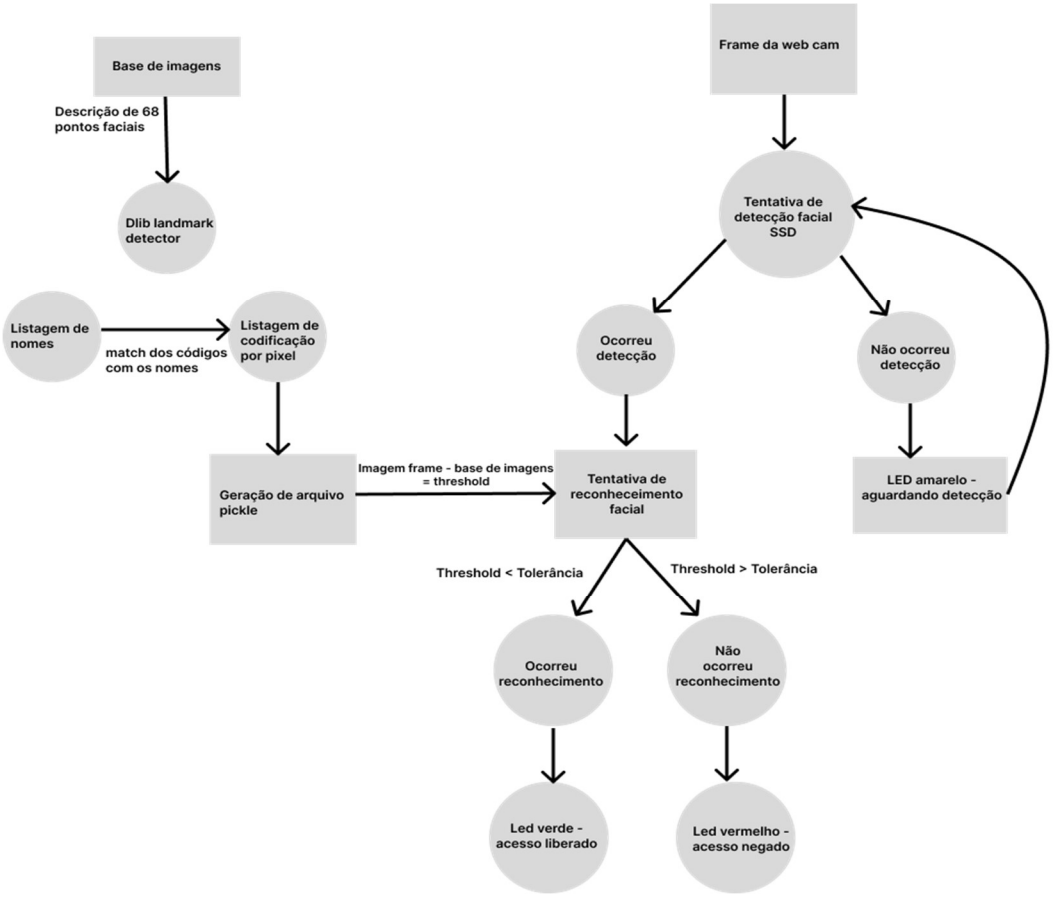
Fonte: Arquivo do grupo PET

Visto que a abertura do circuito é comandada por um sinal digital, foi implementada uma porta lógica OR , de maneira que tanto o acionamento de um botão (para quem estava no interior do escritório sair), quanto o reconhecimento facial eram capazes de abrir a porta de maneira independente.

É possível entender a integração do projeto com um circuito de fechadura eletrônica padrão, ao conectar o pino de saída do sinal de acionamento do relé, quando ocorria reconhecimento facial, junto ao fio que saia do botão de desmagnetização da porta (OPEN) e ligar o GND do relé à referência da fonte de alimentação do circuito.

Assim, os testes obtiveram sucesso ao conseguir abrir a porta por reconhecimento facial, conforme (Figura 6).

Figura 6. Fluxograma do processo de reconhecimento facial



Fonte: Arquivo do grupo PET

3. Conclusão

3.1 Resultados

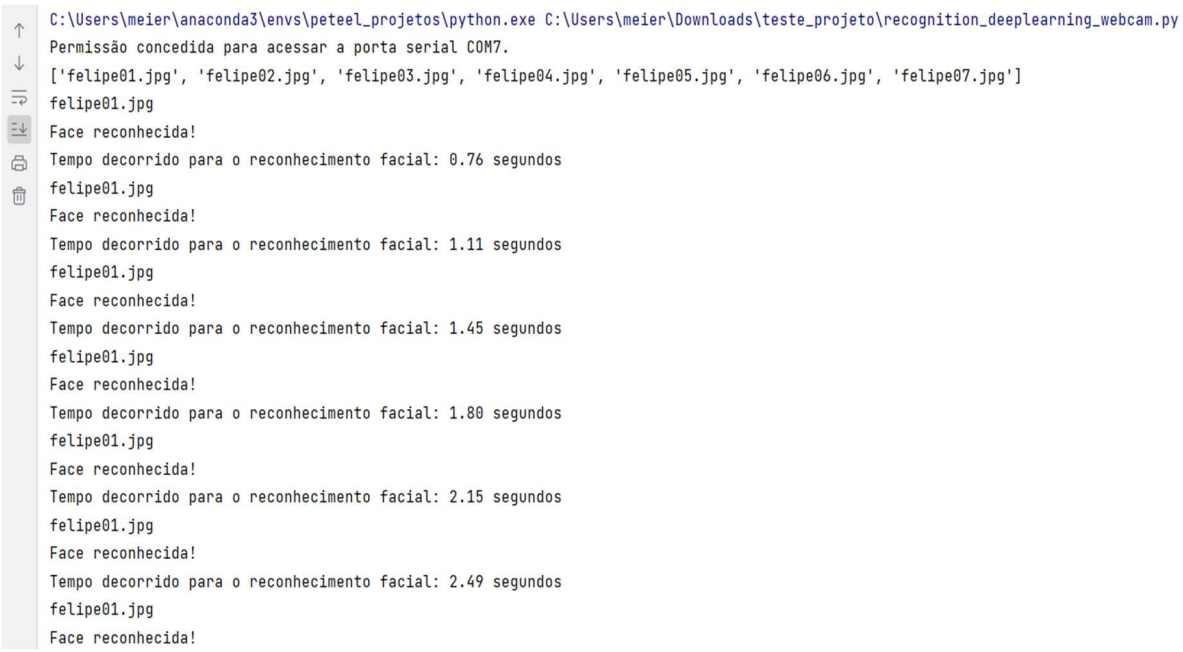
De pontos positivos percebe-se que ao utilizar um Arduino UNO para o acionamento do relé e posterior destrancamento da porta o projeto se torna dinâmico. Uma vez que a latência para o comando ser instruído e por fim efetivado é pequena, em média 350 milissegundos. Conforme a Figura 7, os arquivos que aparecem no terminal tais como felipe01.jpg, felipe02.jpg e etc são as fotos selecionadas no dataset, o aparecimento de cada um corresponde a qual arquivo mais teve proximidade com o reconhecimento na webcam. Em comparação, um sistema comercial observa-se uma média de latência de 100 milissegundos.

Em conclusão, após diversos testes com um database com apenas uma pessoa cadastrada por meio da submissão de algumas fotos do indivíduo. Percebeu-se o overconfidence, no português, excesso de confiança do modelo de classificador (SSD). Houve a visualização desta problemática devido o reconhecimento de perfis parecidos, cor de olhos e cabelos, com a pessoa teste.

Infelizmente, no momento é inviável a descoberta concreta desta problemática, porém, há considerações de fatores desde a quantidade de imagens no database até o treinamento do modelo. Como solução momentânea, teve-se a alteração do threshold para um valor maior, aceitando apenas decisões com acurácias maiores de 80%.

Outra análise é a respeito do reconhecimento haver dificuldades durante a diferenciação de pessoas apenas em uma margem satisfatória em uma luz.

Figura 7. Tempo decorrido entre reconhecimentos faciais



Fonte: Arquivo do grupo PET

3.2 Conclusões Finais

Portanto, por mais que o projeto PET recognition ainda não esteja em sua fase final, pode-se dizer que, além dos projetistas se adentrar pela primeira vez no universo de visão computacional e desenvolverem sua lógica de programação, uma das metas para o projeto, a tecnologia já poderia ser usado para outros fins além da complexidade que exige um sistema de acesso. Um bom exemplo é um despertador que só é desativado uma vez que o indivíduo tem o rosto reconhecido por uma câmera, não precisando de um planejamento mais robusto de hardware.

Outro ponto de melhoria se concentra no fator de segurança. Visto que, com base na literatura, o algoritmo de deep learning do dlib analisa apenas pontos na face do indivíduo, o reconhecimento perde, de certa forma, a noção de profundidade, que pode possibilitar alguém mal-intencionado o acesso ao escritório pela foto de uma face que o algoritmo reconheceria. Dessa maneira, pode-se solucionar esse problema de determinados meios: uma opção é analisar um

algoritmo que possa fazer a leitura dos frames da webcam excluindo possíveis imagens geradas pelo celular ou até implementar uma lógica de permitir o acesso após uma amostra maior de frames reconhecidos, tornando o projeto mais confiável. Outra maneira de solucionar o problema é o planejamento de uma lâmpada anexa à câmera, de forma que, ao acontecer tentativas de acesso via foto, a reflexão da luz com o celular impossibilite a detecção da face dada baixa resolução.

Em termos de objetivo, o projeto alcançou os resultados esperados, uma vez que o sistema de reconhecimento facial tende a ter empecilhos no que diz respeito à segurança mínima. Além disso, foi o primeiro contato dos projetistas com as bibliotecas citadas, o que possibilitou a evolução na lógica de programação e aprendizados iniciais nas redes neurais.

4. Agradecimentos

Os autores agradecem e reconhecem o suporte oferecido pelo Programa de Educação Tutorial (PET) do Ministério da Educação do Brasil.

Referências

ALVES, Luiz. “Implementação de um sistema de fechadura eletrônica usando arduino.” Acesso em: 29 de agosto de 2023.

"ESP32-CAM - Controle de Acesso com Reconhecimento Facial e Jarvis - IeC119." Disponível em: https://www.youtube.com/watch?v=MGPL10N9YmM&t=703s. Acesso em: 29 de agosto de 2023.

"ESP32 com Câmera e Reconhecimento Facial." Disponível em: https://www.youtube.com/watch?v=915jxGwLxxI. Acesso em: 29 de agosto de 2023.

GRANATYR, Jones; ALVES, Gabriel. "Detecção e Reconhecimento Facial com Python." Disponível em: https://www.udemy.com/course/deteccao-reconhecimento-facial-python/. Acesso em 29 de agosto de 2023.

KHAN, Maliha; CHAKRABORTY, Sudeshna; ASTYA, Rani; KHEPRA, Shaveta. “Face Detection and Recognition Using OpenCV”. Acesso em: 29 de agosto de 2023.

LIU, Wei. “SSD: Single Shot MultiBox Detector”. Acesso em: 29 de agosto de 2023.

M Haan, O Pascalis, MH Johnson. Specialization of neural mechanisms underlying face recognition in human infants - Journal of cognitive neuroscience, 2002 - direct.mit.edu. Acesso em: 29 de agosto de 2023.

NOGUEIRA, Gustavo; SANTOS, Felipe. “Desenvolvimento de protótipo de fechadura eletrônica com Reconhecimento Facial”. Acesso em: 29 de agosto de 2023.

"Primeiro Projeto de Reconhecimento Facial com ESP32 e Inteligência Artificial - Eletrônica Fácil." Disponível em: https://www.youtube.com/watch?v=Fdia1FPgsOY. Acesso em: 29 de agosto de 2023.