UNIVERSIDADE PAULISTA – UNIP

BRUNO SANTOS DE OLIVEIRA / F34HAA6
JAYANE SILVA DURSO CORREA / G5486B6
NICK ARCANA FRONTANILLA / N9431C0
RAFAEL DE DEMO RAMOS / G0557G8
SABRINA DA SILVA / G550390

DESENVOLVIMENTO DE UM SISTEMA DE IDENTIFICAÇÃO E AUTENTICAÇÃO BIOMÉTRICA

SÃO PAULO
2024

UNIVERSIDADE PAULISTA – UNIP

DESENVOLVIMENTO DE UM SISTEMA DE IDENTIFICAÇÃO E AUTENTICAÇÃO BIOMÉTRICA

Trabalho de Atividades Práticas Supervisionadas para Graduação em Ciência da Computação apresentado à Universidade Paulista - UNIP

SÃO PAULO
2024

SUMÁRIO

1. OBJETIVO E MOTIVAÇÃO DO TRABALHO

O objetivo deste trabalho é realizar uma análise aprofundada das ferramentas de identificação e autenticação biométrica baseadas em impressões digitais, abordando seus fundamentos teóricos, metodologias, tecnologias envolvidas e aplicações práticas. A biometria por impressões digitais utiliza características únicas das impressões digitais de cada indivíduo para realizar a verificação de identidade, sendo amplamente empregada em diversos setores, como segurança da informação, controle de acesso físico, autenticação em dispositivos móveis e sistemas bancários.

Neste estudo, serão investigadas as principais tecnologias utilizadas nesses sistemas, como sensores de impressão digital, algoritmos de reconhecimento e as técnicas de armazenamento e proteção de dados biométricos. Além disso, será explorado o ciclo de funcionamento dessas ferramentas, desde a captura da imagem da impressão digital até o processo de comparação e verificação.

Serão também discutidas as vantagens da utilização desse método, como a conveniência e a precisão, além dos potenciais limitações, como a vulnerabilidade a ataques de falsificação e questões relacionadas à privacidade dos dados. Por fim, o trabalho irá abordar os avanços tecnológicos recentes que vêm sendo incorporados a essas ferramentas, com o intuito de aumentar a eficiência, a segurança e a proteção da privacidade dos usuários.

O objetivo de aplicações nesse ramo é, de forma clara e objetiva, comparar centenas de milhares de imagens presentes em um dataset, ou até em casos de trabalhar com uma alta escala de dados(imagens), usar um banco de dados. Para realizar caracterização e autenticação de imagens. Nesse caso específico, o programa realiza o “tratamento” de imagens que apresentam algum tipo de obstrução, (um borrão, distorção ou qualquer erro que comprometa de forma clara a identificação da imagem).

Exemplos como estes, podem ser encontrados em aplicações simples do nosso cotidiano, podemos citar aparelhos de identificação e autenticação de condomínios privativos, para controlar o acesso de moradores e visitantes de um edifício, por exemplo. Em casos como esses, um banco de dados gigantesco (equivalente a quantidade de moradores) é preenchida com dados

e informações necessárias para realizar a autenticação, como fotos das digitais e do rosto. E a partir de um algoritmo, assim como este apresentado nesse trabalho, realiza a sua autenticação.

Até em exemplos um pouco mais complexos, como o qual estávamos vivenciando a algumas semanas atrás, durante a eleição municipal, onde ao realizar o voto é comum o eleitor assinar a ficha, bem como realizar a leitura de sua digital. Tal processo é um dos exemplos que aplicações como estas estão presentes.

O reconhecimento de impressões digitais é uma das técnicas de autenticação biométrica mais antigas e confiáveis, amplamente adotada para identificar e autenticar indivíduos com precisão e segurança. Cada pessoa possui um padrão exclusivo de sulcos e cristas nas pontas dos dedos, tornando as impressões digitais uma característica biométrica única e altamente diferenciada. Essa tecnologia tem evoluído ao longo dos anos, passando de uma prática utilizada principalmente em investigações criminais para uma ferramenta essencial em aplicações de segurança digital, controle de acesso, e autenticação em dispositivos pessoais, como smartphones e computadores.

O funcionamento do reconhecimento de impressões digitais envolve a captura da imagem da impressão digital através de sensores especializados. Esses sensores podem ser ópticos, capacitivos ou ultrassônicos, cada um com diferentes métodos para detectar as minúcias da impressão. Após a captura, um algoritmo processa a imagem e identifica características únicas, como bifurcações e terminações de sulcos, que são então comparadas com dados armazenados em um banco de dados. Esse processo possibilita verificar a identidade do usuário de forma rápida e eficiente, oferecendo uma solução segura e conveniente para autenticação.

A popularidade do reconhecimento de impressões digitais deve-se a suas inúmeras vantagens, como alta precisão, velocidade e conveniência, especialmente em comparação com métodos tradicionais, como senhas e códigos PIN. Além disso, a exclusividade das impressões digitais de cada indivíduo proporciona um alto nível de segurança, dificultando tentativas de fraude. No entanto, esse método também enfrenta desafios e limitações, incluindo a vulnerabilidade a ataques de falsificação, desgaste da pele e

questões de privacidade quanto ao armazenamento e à proteção dos dados biométricos.

Nos últimos anos, avanços significativos têm transformado a tecnologia de reconhecimento de impressões digitais, graças ao uso de inteligência artificial, aprendizado de máquina e aprimoramentos em processamento de imagem. Esses avanços permitiram o desenvolvimento de sistemas mais robustos, capazes de lidar com variações nas condições das impressões digitais e melhorar a resiliência contra fraudes. Tais inovações também contribuíram para a redução de falsos positivos e falsos negativos, elevando ainda mais a precisão e a confiabilidade desses sistemas.

Considerando a crescente dependência da sociedade em sistemas digitais e a importância da segurança da informação, o reconhecimento de impressões digitais continua a ser um tema de grande relevância. Este trabalho explora as tecnologias envolvidas, as metodologias aplicadas, os recentes avanços e as limitações desse método de autenticação. Além disso, serão discutidas as tendências futuras no campo da biometria e o impacto do reconhecimento de impressões digitais na proteção de dados e na vida cotidiana, destacando seu papel fundamental na evolução da segurança digital.

2. INTRODUÇÃO

A biometria oferece uma alternativa segura e confiável em relação aos métodos tradicionais de autenticação. O desenvolvimento de um sistema biométrico pode contribuir significativamente para a melhoria da segurança e autenticação em diversas áreas.

A biometria trata-se do uso das características físicas ou biométricas, como a impressão digital ou o reconhecimento facial, para identificar uma pessoa. Considerando que as características de cada indivíduo servem como parâmetro, o objetivo principal seria a geração de “senhas únicas” correspondentes a estas características, para dificultar os acessos aos seus dados armazenados digitalmente em diversos dispositivos como o celular, instrumento presente no dia a dia de uma grande parcela das populações dos centros urbanos.

Nos últimos anos, a segurança da informação tem sido um dos temas centrais nas discussões sobre tecnologia e inovação, especialmente no contexto da proteção de dados pessoais e sensíveis. O avanço das tecnologias digitais e o aumento da quantidade de informações armazenadas em sistemas online têm gerado preocupações sobre o risco de acessos não autorizados, fraudes e invasões. Nesse cenário, métodos tradicionais de autenticação, como senhas e PINs, têm se mostrado cada vez mais vulneráveis a ataques cibernéticos, o que exige a busca por soluções mais eficazes e seguras. A biometria, como método de identificação e autenticação surge como uma alternativa promissora, devido à sua capacidade de utilizar características físicas e comportamentais únicas dos indivíduos, garantindo maior precisão e segurança no processo de verificação de identidade.

A autenticação biométrica envolve o uso de características biológicas, como impressões digitais, reconhecimento facial, íris e voz, para autenticar um usuário em sistemas digitais. Ao contrário das senhas, que podem ser facilmente esquecidas ou descobertas por meios fraudulentos, as características biométricas são praticamente impossíveis de replicar, proporcionando um alto nível de confiabilidade. Além disso, a biometria por ser mais conveniente acaba reduzindo a necessidade de memorizar informações complexas, como senhas longas e combinações de caracteres.

Neste contexto, o desenvolvimento de um sistema de identificação e autenticação biométrica se torna não apenas uma necessidade, mas uma exigência no que diz respeito à proteção de dados e ao fortalecimento da segurança digital. O objetivo deste trabalho é propor e desenvolver um sistema baseado em tecnologia biométrica para identificação e autenticação de usuários, com ênfase na implementação de algoritmos de reconhecimento de impressões digitais. A escolha da impressão digital como parâmetro biométrico baseia-se em sua ampla aplicação em sistemas de segurança e sua eficácia comprovada em termos de precisão e facilidade de implementação.

A implementação desse sistema visa oferecer uma solução mais segura e eficiente, possibilitando o controle de acessos em diversas áreas, como bancos, sistemas de saúde, dispositivos móveis e outros setores que demandam alta segurança. Além disso, busca-se proporcionar uma experiência mais amigável e ágil para os usuários, que poderão realizar a autenticação de maneira simples e rápida, sem a necessidade de lembrar ou digitar informações sensíveis.

A seguir, serão discutidos os principais conceitos e tecnologias envolvidos no processo de desenvolvimento do sistema, bem como os desafios e soluções encontradas ao longo da implementação. Este trabalho visa contribuir para o avanço da segurança digital, propondo uma solução condizente com o cenário tecnológico atual para a autenticação de usuários, com base em características biométricas únicas e de difícil falsificação.

3. FUNDAMENTOS DAS PRINCIPAIS TÉCNICAS BIOMÉTRICAS

Para iniciar as principais técnicas biométricas, vamos entender o que é biometria.

Biometria é o método de segurança que cada vez mais está presente em nossas vidas, é um dos caminhos mais seguros para a identificação de pessoas e proteção de dados. É usada de caixas automáticos em bancos a terminais de embarque em aeroportos. Até mesmo seu celular, tablet ou computador o método de identificação por digital ou reconhecimento facial.

Biometria é a medição de características físicas e comportamentais para identificação de indivíduos, usada em Segurança da Informação para controlar acesso físico, identificar criminosos e proteger dados sigilosos.

3.1 Tipos de biometrias

3.1.1 Impressão digital

O reconhecimento biométrico pela impressão digital é confiável e de baixo custo. Devido à constância das digitais, a única falha possível é a perda delas. Assim, é amplamente utilizado, sozinho ou em conjunto com outros métodos.

3.1.2 Reconhecimento facial

O reconhecimento facial mapeia um rosto em 3D ou 2D para desbloquear funções e identificar uma pessoa. No entanto, o método não é permanente, pois mudanças no rosto podem ocorrer, tornando a técnica imprecisa. Além disso, há preocupações com a coleta de dados sem consentimento, já que empresas e governos podem utilizá-los de forma invasiva.

3.1.3 Reconhecimento de íris

A biometria usando a íris é altamente confiável e menos invasiva do que a leitura da retina. Embora custosa, ela pode se tornar o método mais usado, superando a impressão digital. Modelos premium de celulares já oferecem scanners de íris para proteger dados.

3.1.4 Reconhecimento de voz

O reconhecimento por voz analisa parâmetros físicos e comportamentais para criar um perfil sonoro único, que pode servir como assinatura biométrica.

Apesar do baixo custo, a confiabilidade é limitada devido a ruídos e mudanças físicas.

Esses são alguns dos mais famosos tipos.

3.2 A segurança da biometria funciona?

A segurança biométrica usa características físicas únicas, como impressões digitais, para verificar a identidade das pessoas. Esses dados são salvos e comparados para garantir o acesso seguro a dispositivos e locais restritos. Em vez de senhas, o corpo de uma pessoa se torna a chave para permitir a entrada. Os scanners biométricos capturam essas informações para garantir a segurança e autenticidade do acesso.

A segurança biométrica é usada em muitas indústrias, como proteger documentos e objetos valiosos. Bancos como Citibank e Halifax usam reconhecimento de voz e batimentos cardíacos para identificar clientes, e a Ford está considerando sensores biométricos nos carros. Passaportes eletrônicos têm biometria, como impressão digital, íris e foto, para evitar acesso não autorizado. Prós e contras desses sistemas de segurança estão sendo observados.

3.3 Biometria e suas questões de identidade e privacidade

A autenticação biométrica traz conveniência, mas gera preocupações sobre privacidade devido ao potencial de coleta de dados pessoais sem consentimento. O reconhecimento facial está cada vez mais presente em cidades como na China e Londres, assim como em aeroportos ao redor do mundo, incluindo Dubai. Algumas cidades como Nova York, Chicago e Moscou utilizam câmeras de circuito interno conectadas a bancos de dados para combater o crime. Isso levanta temores sobre vigilância e possíveis abusos de dados. A tecnologia está em constante evolução, com a Carnegie Mellon University desenvolvendo câmeras capazes de ler íris a 10 metros de distância em multidões.

3.4 Formas de proteger a identidade biométrica

Com a preocupação em relação à segurança e privacidade, é importante adicionar camadas de proteção nos sistemas biométricos. A autenticação se torna mais difícil para invasores quando são necessários múltiplos métodos, como detecção de atividade (como o piscar) e correspondência de amostras criptografadas com usuários. Alguns sistemas também incluem características adicionais nos dados biométricos, como idade, gênero e altura, para dificultar acesso não autorizado. Um exemplo é o programa Aadhaar da Unique ID Authority of India, que utiliza várias etapas de autenticação, como leituras de íris, impressões digitais e reconhecimento facial. A autenticação em duas fases, combinando biometria, um token de hardware e uma senha, é uma abordagem poderosa, especialmente considerando o aumento dos dispositivos IoT. Utilizar um gerenciador de senhas também pode oferecer proteção adicional.

3.5 Etapas para a realização da biometria

Após entender o conceito da biometria e seus diversos tipos, o próximo passo é compreender como ela funciona. Para garantir segurança, é essencial ter uma base de dados de alta qualidade para comparar os traços biométricos. O sistema de análise biométrica utiliza machine learning para capturar e avaliar os dados do usuário, compará-los com a base de dados, conectar as informações a um indivíduo e, com base nisso, permitir ou impedir uma ação. Além da coleta de dados biométricos, é importante validá-los comparando-os com a base de dados. Cada sistema de tradução de dados biométricos tem seu próprio método e níveis de confiabilidade específicos. Por exemplo, na biometria facial, são extraídos até 80 pontos para comparação com a imagem escaneada. Com essas informações, fica mais fácil entender como funciona a biometria e os sistemas biométricos. Dúvidas sobre o cadastro de características físicas ou comportamentais são comuns, mas compreender o processo pode ajudar a dissipá-las.

Para cadastrar a biometria, é necessário passar pelas etapas de captura e análise dos dados biométricos para confirmar a identidade. Escolha o tipo de biometria, como facial ou impressão digital, e siga os passos apresentados para registrar o ponto no aplicativo da empresa. O cadastro das características deve

seguir um padrão para garantir a qualidade dos dados e evitar falhas de identificação no sistema de ponto biométrico.

Cada técnica biométrica possui vantagens e desafios únicos. A escolha da técnica ideal depende do nível de segurança exigido, do contexto de aplicação e das questões de privacidade envolvidas.

Essas fontes bibliográficas fornecem uma base sólida para entender o funcionamento, as aplicações e as inovações em cada tipo de biometria.

4. PLANO DE DESENVOLVIMENTO DA APLICAÇÃO

O projeto será inteiramente escrito usando a linguagem Python, assim como seus respectivos frameworks e bibliotecas. Entre os motivos da escolha de Python, dentro as outras duas opções, se deve principalmente a sua facilidade e clareza em escrever códigos como estes, além de bibliotecas já presentes no Python que facilitam ainda mais o polimento final da aplicação.

Ao desenvolver uma aplicação de reconhecimento de impressões digitais, a escolha da linguagem de programação é fundamental para garantir eficiência, rapidez no desenvolvimento e facilidade de manutenção. Python destaca-se como a melhor opção para este tipo de aplicação quando comparado a linguagens como C# e Java, por diversos motivos.

Python possui bibliotecas extensas e poderosas para processamento de imagens e visão computacional, como **OpenCV**, **Pillow (PIL)** e **NumPy**. Essas bibliotecas são maduras, bem documentadas e amplamente utilizadas na comunidade. No programa desenvolvido, o uso do OpenCV facilita a detecção de características nas impressões digitais e a comparação entre elas, tarefas que seriam mais complexas e verbosas em C# ou Java.

A sintaxe clara e concisa do Python permite que os desenvolvedores escrevam menos código para realizar tarefas complexas. Isso aumenta a produtividade e facilita a manutenção do código. No exemplo dado, a implementação da interface gráfica com **Tkinter** e a manipulação de imagens são feitas com menos linhas de código do que seria necessário em C# (com Windows Forms ou WPF) ou Java (com Swing ou JavaFX).

Além de possuir uma comunidade global ativa que contribui com bibliotecas, ferramentas e soluções para diversos problemas. A disponibilidade de recursos e suporte facilita a resolução de desafios durante o desenvolvimento. Além disso, bibliotecas específicas para biometria e reconhecimento de impressões digitais estão mais presentes no ecossistema Python.

O reconhecimento de padrões em impressões digitais pode ser aprimorado com técnicas de aprendizado de máquina. Python é a linguagem preferida para projetos de **machine learning** e **inteligência artificial**, oferecendo bibliotecas como **scikit-learn**, **TensorFlow** e **Keras**. Essa integração é mais direta em Python do que em C# ou Java, que podem

requerer bibliotecas de terceiros menos maduras ou integrações mais complexas.

Python é excelente para prototipagem rápida devido à sua natureza interpretada e dinâmica. Isso permite testar e iterar sobre o código mais rapidamente, reduzindo o tempo de desenvolvimento. Em C# e Java, o processo de compilação e a verbosidade da linguagem podem retardar essa etapa crucial.

Aplicações em Python são altamente portáveis e funcionam em diversos sistemas operacionais sem modificações significativas no código. Isso é vantajoso para distribuir a aplicação a usuários com diferentes ambientes (Windows, macOS, Linux). Embora C# e Java também sejam multiplataforma, podem apresentar incompatibilidades ou requerer ajustes específicos.

Python é propriamente conhecido por sua baixa curva de aprendizado, o que permite que equipes de desenvolvimento se tornem produtivas rapidamente. Para projetos que exigem colaboração ou integração de novos membros, essa característica é um diferencial importante.

Python suporta múltiplos paradigmas de programação (procedural, orientado a objetos, funcional), oferecendo flexibilidade na abordagem de resolução de problemas. Essa versatilidade pode ser limitada em C# e Java, que são predominantemente orientadas a objetos.

Por fim, a aplicação de reconhecimento de impressões digitais apresentada, beneficia-se significativamente das vantagens oferecidas por Python. A combinação de bibliotecas robustas para processamento de imagens, sintaxe simplificada, e uma comunidade ativa tornam Python a escolha ideal para este tipo de projeto. Embora C# e Java sejam linguagens poderosas e amplamente utilizadas, elas não oferecem a mesma agilidade e facilidade para tarefas específicas de processamento de imagens e visão computacional.

Optar por Python reduz o tempo de desenvolvimento, facilita a manutenção e permite que a equipe foque nas funcionalidades essenciais da aplicação, em vez de lidar com complexidades desnecessárias inerentes a linguagens mais verbosas ou com menor suporte para as bibliotecas necessárias.

5. ESTRUTURA E MÓDULOS USADOS

A aplicação fornecida é um programa em Python que realiza a correspondência de impressões digitais utilizando técnicas de visão computacional. O código está organizado em uma classe principal chamada **FingerprintMatcher** que integra a lógica de processamento de imagens com uma interface gráfica interativa para o usuário.

Os módulos usados neste projeto são:

- OpenCV(cv2): para processamento de imagens e implementação dos algoritmos de detecção e correspondência de características.
- OS(os): para interagir com o sistema de arquivos, permitindo navegar pelos diretórios e acessar os arquivos de imagem.
- Time(time): para medir o tempo de execução do processo de correspondência, fornecendo feedback de desempenho.
- Tkinter(tk): e seus submódulos: para construir a interface gráfica com o usuário (GUI), criando janelas, botões e outros elementos de interação.
- PIL(Pillow): para manipulação e conversão de imagens, facilitando o trabalho com diferentes formatos e a integração com o Tkinter.
- NumPy(numpy): para operações numéricas e manipulação eficiente de arrays, especialmente na conversão e processamento de imagens.

Dentro da classe FingerprintMatcher, o método __**init**__ configura a interface gráfica, definindo o título da janela, seu tamanho, propriedades de redimensionamento e estilo visual. Também carrega um ícone para a aplicação e cria os elementos da GUI, como labels para exibir mensagens ao usuário, frames para organizar os botões e os próprios botões que permitem selecionar uma imagem e executar o processo de correspondência. Além disso, inicializa variáveis que serão utilizadas para armazenar a imagem selecionada, o melhor score obtido, o nome do arquivo correspondente e as características detectadas.

O método **select_image** é responsável por permitir que o usuário selecione uma imagem de impressão digital a partir do sistema de arquivos. Utiliza uma caixa de diálogo fornecida pelo Tkinter para que o usuário navegue e escolha o arquivo desejado. A imagem selecionada é então carregada utilizando o PIL, convertida para o espaço de cores adequado e armazenada para uso posterior. Caso ocorra algum erro no carregamento da imagem ou o usuário não selecione um arquivo, são exibidas mensagens de erro ou informação correspondentes.

O método **run_fingerprint_matching** implementa o núcleo do processamento da aplicação. Inicialmente, verifica se uma imagem foi selecionada e se o diretório contendo as imagens de referência existe. Em seguida, inicia uma contagem de tempo para medir o desempenho do processo. O método então itera sobre um conjunto de arquivos de imagem presentes no diretório especificado, limitando-se aos primeiros 1000 arquivos para otimizar o tempo de execução. Para cada imagem de referência, o programa então:

1. Carrega a imagem e a converte para o formato apropriado para processamento.

2. Utiliza o algoritmo SIFT (Scale-Invariant Feature Transform) para detectar pontos-chave (keypoints) e calcular descritores das características tanto da imagem selecionada pelo usuário quanto da imagem de referência.

3. Emprega o **FlannBasedMatcher**, um algoritmo de correspondência eficiente, para encontrar correspondências entre os descritores das duas imagens.

4. Aplica o critério de Lowe para filtrar as correspondências, mantendo apenas as que atendem a uma relação de distância específica.

5. Calcula um score baseado no número de correspondências válidas em relação ao número mínimo de keypoints detectados entre as duas imagens.

6. Atualiza o melhor score e armazena as informações correspondentes caso o score atual seja superior ao mais bem encontrado até o momento.

Após processar todas as imagens de referência, o método **display_results** é chamado para exibir os resultados. Ele calcula o tempo total de execução e verifica se um match válido foi encontrado. Se houver, utiliza a função **cv2.drawMatches** para desenhar as correspondências entre a imagem selecionada e a melhor imagem de referência encontrada. A imagem resultante é redimensionada e convertida para um formato compatível com a interface gráfica, sendo então exibida ao usuário. Além disso, são atualizados os textos da interface para informar o nome do arquivo correspondente, o score obtido e o tempo de execução. Caso nenhum match tenha sido encontrado, uma mensagem informando tal fato é exibida.

O fluxo principal do programa é iniciado no **bloco if __name__ == "__main__":**, onde uma instância da classe FingerprintMatcher é criada e o loop principal da interface gráfica é iniciado com **root.mainloop()**. Isso mantém a aplicação em execução, aguardando a interação do usuário.

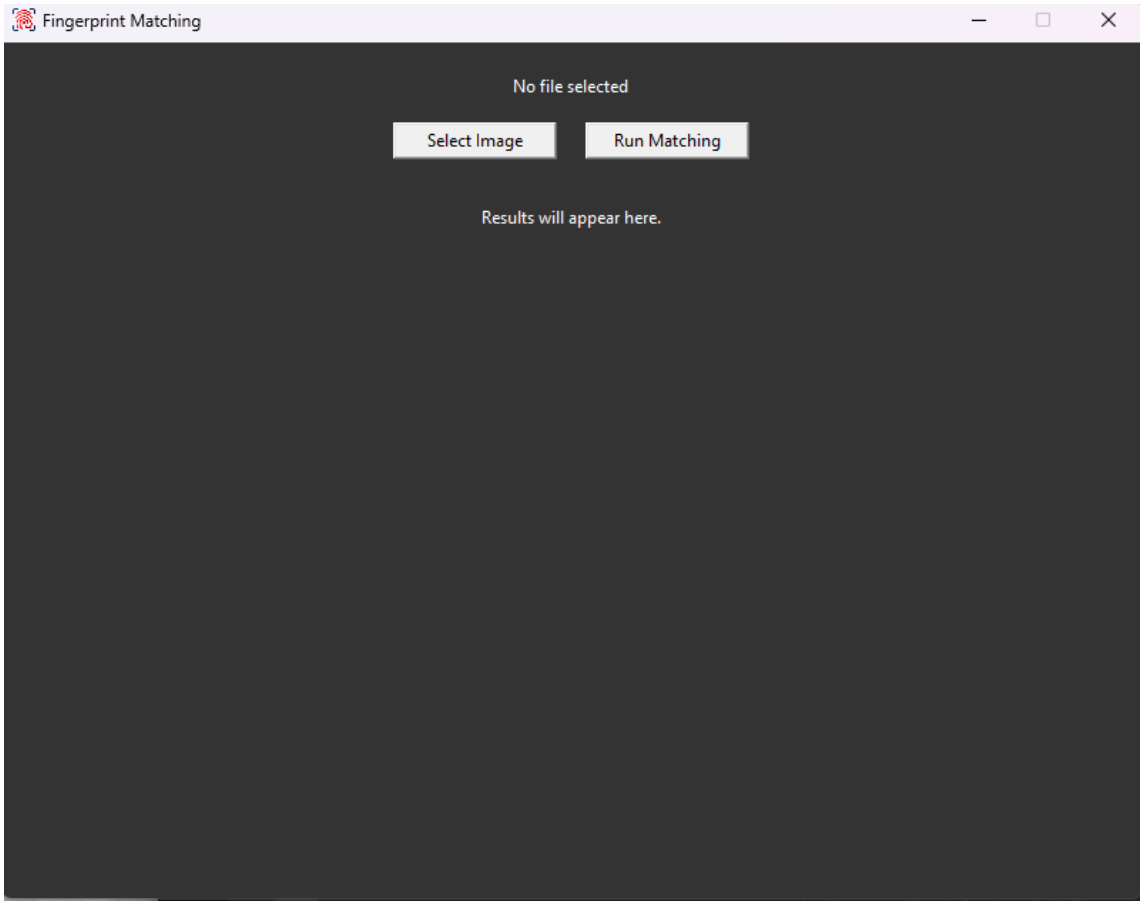
Em síntese, o código integra diversos módulos para oferecer uma aplicação funcional de correspondência de impressões digitais. O OpenCV é utilizado para as operações de processamento de imagem e algoritmos de detecção e correspondência. O Tkinter fornece os recursos para a criação de uma interface gráfica interativa. O PIL facilita o carregamento e manipulação das imagens, enquanto o NumPy auxilia nas operações numéricas necessárias.

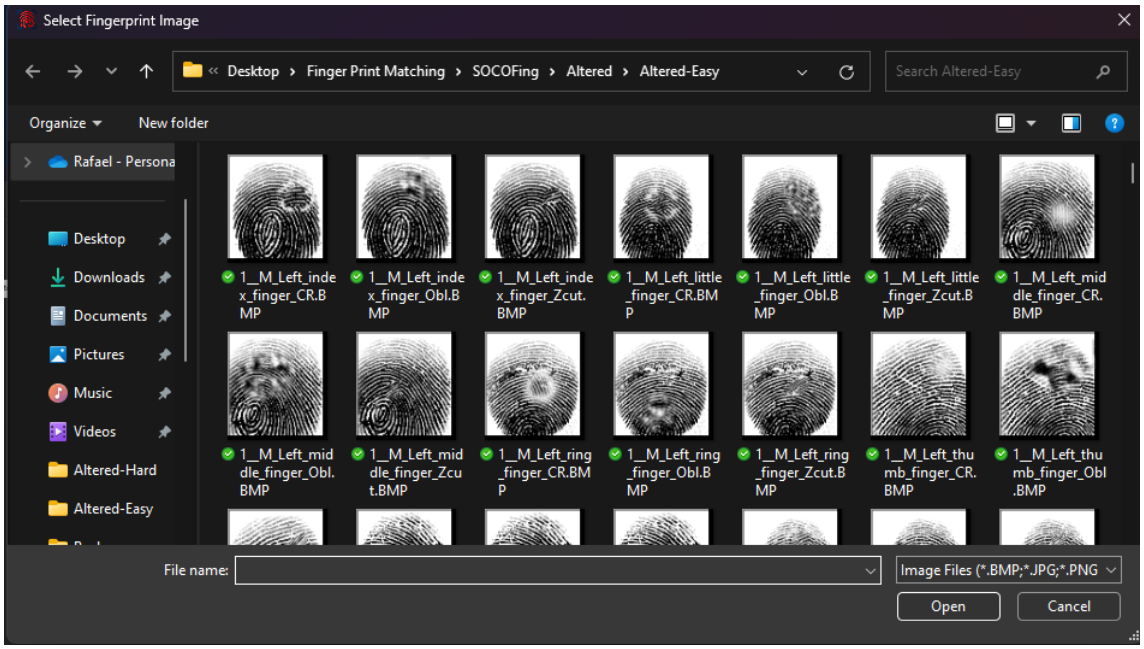
A aplicação funciona permitindo que o usuário selecione uma imagem de impressão digital que deseja comparar. O programa então processa essa imagem e a compara com um conjunto de imagens de referência armazenadas em um diretório específico. Utiliza algoritmos robustos de detecção de características e correspondência para identificar similaridades entre as impressões digitais. Ao final, exibe os resultados de forma visual, mostrando as correspondências encontradas e fornecendo informações sobre a melhor correspondência identificada.

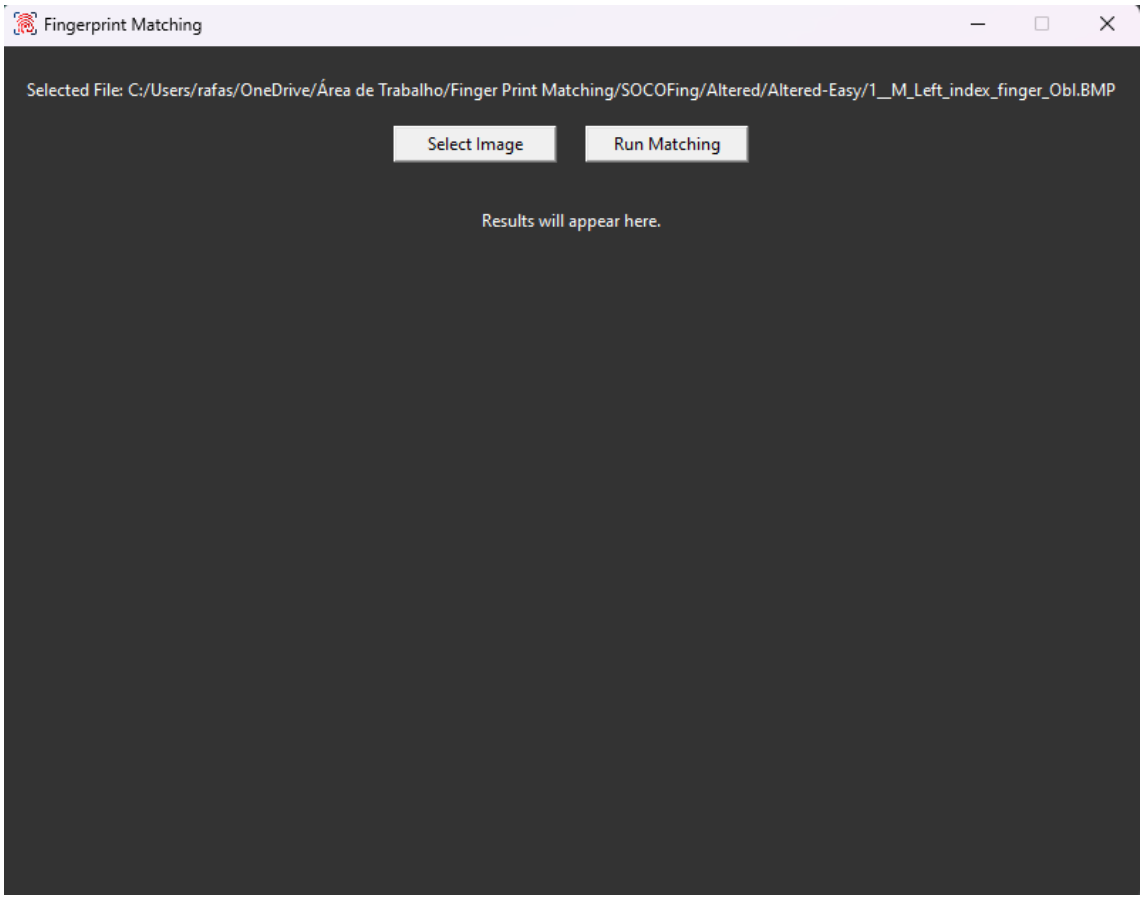
Este projeto exemplifica como a linguagem Python e suas bibliotecas podem ser utilizadas para desenvolver aplicações complexas de forma relativamente simples e eficiente. A estrutura modular do código facilita a

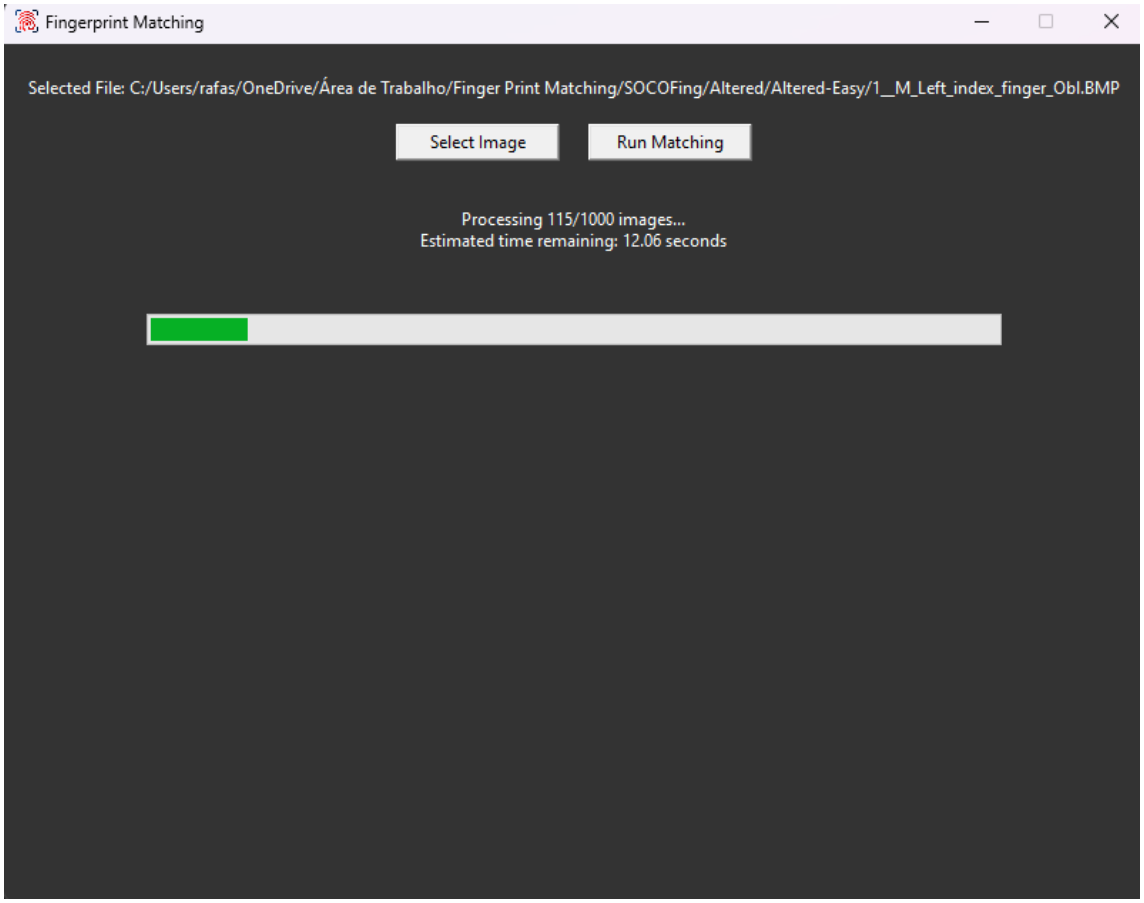
manutenção e expansão futura, permitindo que novos recursos sejam adicionados ou melhorias sejam feitas nos algoritmos de processamento ou na interface com o usuário.

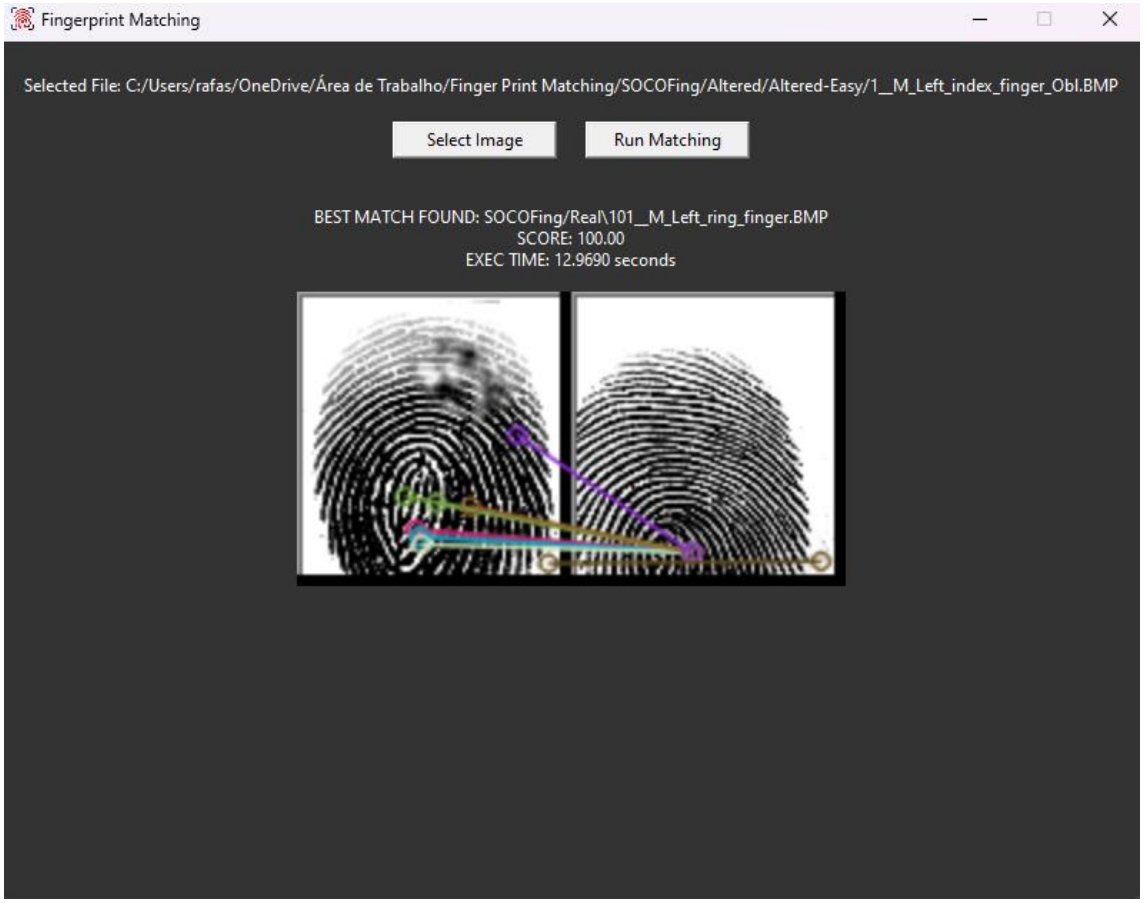
6. APRESENTAÇÃO DA APLICAÇÃO






Fingerprint Matching
Selected File: C:/Users/rafas/OneDrive/Área de Trabalho/Finger Print Matching/SOCOFing/Altered/Altered-Easy/1__M_Left_index_finger_Obl.BMP
Select Image
Run Matching
Results will appear here.


Fingerprint Matching
Selected File: C:/Users/rafas/OneDrive/Área de Trabalho/Finger Print Matching/SOCOFing/Altered/Altered-Easy/1__M_Left_index_finger_Obl.BMP
Select Image
Run Matching
Processing 115/1000 images...
Estimated time remaining: 12.06 seconds


Fingerprint Matching
Selected File: C:/Users/rafas/OneDrive/Área de Trabalho/Finger Print Matching/SOCOFing/Altered/Altered-Easy/1__M_Left_index_finger_Obl.BMP
Select Image
Run Matching
BEST MATCH FOUND: SOCOFing/Real\101__M_Left_ring_finger.BMP
SCORE: 100.00
EXEC TIME: 12.9690 seconds

7. RELATÓRIO COM AS LINHAS DE CÓDIGO

```
import cv2
import os
import time
import tkinter as tk
from tkinter import filedialog, messagebox
from tkinter import ttk
from PIL import Image, ImageTk
import numpy as np
import threading

class FingerprintMatcher:
    def __init__(self, root):
        self.root = root
        self.sample = None
        self.best_Score = 0
        self.best_filename = None
        self.best_image = None
        self.kp1, self.kp2, self.mp = None, None, None

        # GUI Configuration
        self.setup_gui()

    def setup_gui(self):
        # Window setup
        self.root.title("Fingerprint Matching")
        self.root.geometry("800x600")
        self.root.resizable(False, False)
        self.root.configure(bg='gray20')

        # Icon and Labels
        photo = tk.PhotoImage(file='img/fingerprint-scan.png')
        self.root.wm_iconphoto(False, photo)

        self.filename_label = tk.Label(self.root, text="No file selected", bg='gray20',
fg='white')
        self.filename_label.pack(pady=(20, 10))

        # Frame for buttons
        button_frame = tk.Frame(self.root, bg='gray20')
        button_frame.pack(pady=(5, 20))
```

```
        select_button = tk.Button(button_frame, text="Select Image",
command=self.select_image, width=15)
        select_button.pack(side=tk.LEFT, padx=10)

        run_button = tk.Button(button_frame, text="Run Matching",
command=self.run_fingerprint_matching_thread,
                               width=15)
        run_button.pack(side=tk.LEFT, padx=10)

        # Progress bar and result display text (initially hidden)
        self.progress = ttk.Progressbar(self.root, orient="horizontal", length=600,
mode="determinate")
        self.progress.pack(pady=10)
        self.progress.pack_forget()  # Hide the progress bar initially

        self.result_text = tk.StringVar()
        self.result_text.set("Results will appear here.")
        result_label_text = tk.Label(self.root, textvariable=self.result_text,
bg='gray20', fg='white')
        result_label_text.pack(pady=10)

        # Label to display the result image
        self.result_label = tk.Label(self.root, bg='gray20')
        self.result_label.pack(padx=20)

    def select_image(self):
        filepath = filedialog.askopenfilename(title="Select Fingerprint Image",
                                              filetypes=[("Image Files", "*.BMP;*.JPG;*.PNG")])
        if not filepath:
            messagebox.showinfo("Info", "No file selected")
            return

        # Load and display the selected image
        pil_image = Image.open(filepath).convert("RGB")
        self.sample = cv2.cvtColor(np.array(pil_image), cv2.COLOR_RGB2BGR)
        self.filename_label.config(text=f"Selected File: {filepath}")
        print(f"Image loaded successfully from: {filepath}\n")

    def run_fingerprint_matching_thread(self):
        """ Run fingerprint matching in a separate thread to avoid freezing the GUI.
"""
        if self.sample is None:
            messagebox.showerror("Error", "Please select an image first")
            return
```

```
        # Check if dataset path exists
        if not os.path.exists("SOCOFing/Real"):
            messagebox.showerror("Error", "Directory 'SOCOFing/Real' does not exist. Please check the path.")
            return

        # Show the progress bar when starting the process
        self.progress.pack(pady=10)

        # Start processing in a new thread
        threading.Thread(target=self.run_fingerprint_matching).start()

    def run_fingerprint_matching(self):
        self.best_Score = 0
        self.best_filename = None
        self.best_image = None
        self.kp1, self.kp2, self.mp = None, None, None

        # Retrieve image files and initialize progress
        image_files = self.get_image_files("SOCOFing/Real")[:1000]  # Limit to first 1000 images
        total_files = len(image_files)
        self.progress["maximum"] = total_files

        start_time = time.time()

        for i, file in enumerate(image_files):
            finger_print_img = self.load_image(file)
            if finger_print_img is None:
                continue  # Skip invalid images

            # Perform matching using SIFT and FLANN
            score, keypoints1, keypoints2, match_points = self.match_fingerprints(finger_print_img)

            # Update the best match if the score is higher
            if score > self.best_Score:
                self.best_Score, self.best_filename, self.best_image = score, file, finger_print_img
                self.kp1, self.kp2, self.mp = keypoints1, keypoints2, match_points

            # Update progress bar and estimate remaining time
            elapsed_time = time.time() - start_time
```

```
            remaining_time = (elapsed_time / (i + 1)) * (total_files - (i + 1))
            self.progress["value"] = i + 1
            self.result_text.set(
                f"Processing {i + 1}/{total_files} images...\nEstimated time remaining: {remaining_time:.2f} seconds")
            self.root.update_idletasks()

        # Hide the progress bar once the process is complete
        self.progress.pack_forget()

        self.display_results(start_time)

    def get_image_files(self, directory):
        """ Retrieve all valid image files from the specified directory. """
        return [os.path.join(directory, file) for file in os.listdir(directory) if
                file.lower().endswith(('.bmp', '.jpg', '.png'))]

    def load_image(self, file_path):
        """ Load an image from the specified path and return it as a BGR image. """
        try:
            pil_image = Image.open(file_path).convert("RGB")
            return cv2.cvtColor(np.array(pil_image), cv2.COLOR_RGB2BGR)
        except Exception as e:
            print(f"Warning: Could not load image {file_path}. Error: {e}")
            return None

    def match_fingerprints(self, fingerprint_image):
        """ Match the sample fingerprint with a given fingerprint image using SIFT and FLANN. """
        sift = cv2.SIFT_create()
        keypoints1, descriptor1 = sift.detectAndCompute(self.sample, None)
        keypoints2, descriptor2 = sift.detectAndCompute(fingerprint_image, None)

        if descriptor1 is None or descriptor2 is None:
            return 0, None, None, None

        # FLANN matching with ratio test
        flann = cv2.FlannBasedMatcher({'algorithm': 1, 'trees': 5}, {'checks': 50})
        matches = flann.knnMatch(descriptor1, descriptor2, k=2)
        match_points = [p for p, q in matches if p.distance < 0.7 * q.distance]

        # Calculate match score
        keypoints = min(len(keypoints1), len(keypoints2))
        score = (len(match_points) / keypoints) * 100 if keypoints > 0 else 0
```

```
        return score, keypoints1, keypoints2, match_points

    def display_results(self, start_time):
        """ Display the matching results on the GUI. """
        exec_time = time.time() - start_time
        if self.best_image is not None:
            result_img = self.create_result_image()
            self.show_result_image(result_img)
            self.result_text.set(
                f"BEST MATCH FOUND: {self.best_filename}\nSCORE:
{self.best_Score:.2f}\nEXEC TIME: {exec_time:.4f} seconds")
        else:
            self.result_text.set("No match found.")
            print("No match found after processing all files.")

    def create_result_image(self):
        """ Draw matches between the sample and the best matching fingerprint. """
        result = cv2.drawMatches(self.sample, self.kp1, self.best_image, self.kp2,
self.mp, None,

flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
        result = cv2.resize(result, None, fx=2, fy=2)
        return Image.fromarray(cv2.cvtColor(result, cv2.COLOR_BGR2RGB))

    def show_result_image(self, result_image):
        """ Display the result image on the GUI. """
        result_img_tk = ImageTk.PhotoImage(result_image)
        self.result_label.config(image=result_img_tk)
        self.result_label.image = result_img_tk  # Keep reference to prevent
garbage collection

if __name__ == "__main__":
    root = tk.Tk()
    app = FingerprintMatcher(root)
    root.mainloop()
```

BIBLIOGRAFIAS

GOGONI, R. O que é biometria? Os 4 tipos mais usados na tecnologia. Disponível em: <https://tecnoblog.net/responde/o-que-e-biometria-tecnologia/>. Acesso em: 26 out. 2024.

O que é biometria e como é utilizada na segurança? Disponível em: <https://www.kaspersky.com.br/resource-center/definitions/biometrics>. Acesso em: 2 nov. 2024.

TOTVS, E. O que é biometria: como ela funciona e principais tipos. Disponível em: <https://www.totvs.com/blog/gestao-para-assinatura-de-documentos/o-que-e-biometria/>. Acesso em: 29 out. 2024.

Tipos de biometria: conheça os principais. Disponível em: <https://www.vsoft.com.br/post/principais-tipos-de-biometria>. Acesso em: 5 nov. 2024.