Desenvolvimento de um programa para reconhecimento facial de emoções

Professor – Adrian Dediu

Mario Felix 30011520 – João Reis 30010484

Índice

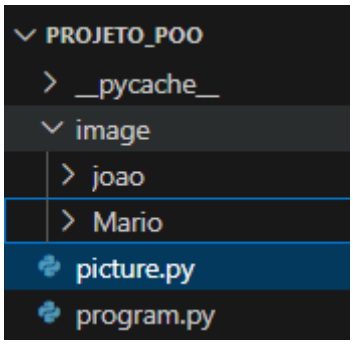
1.Introdução

O objetivo deste projeto consistiu em desenvolver um sistema de reconhecimento facial e emoções

Para a sua implementação foi utilizada a ferramenta Visual Studio Code com a linguagem de programação Python.

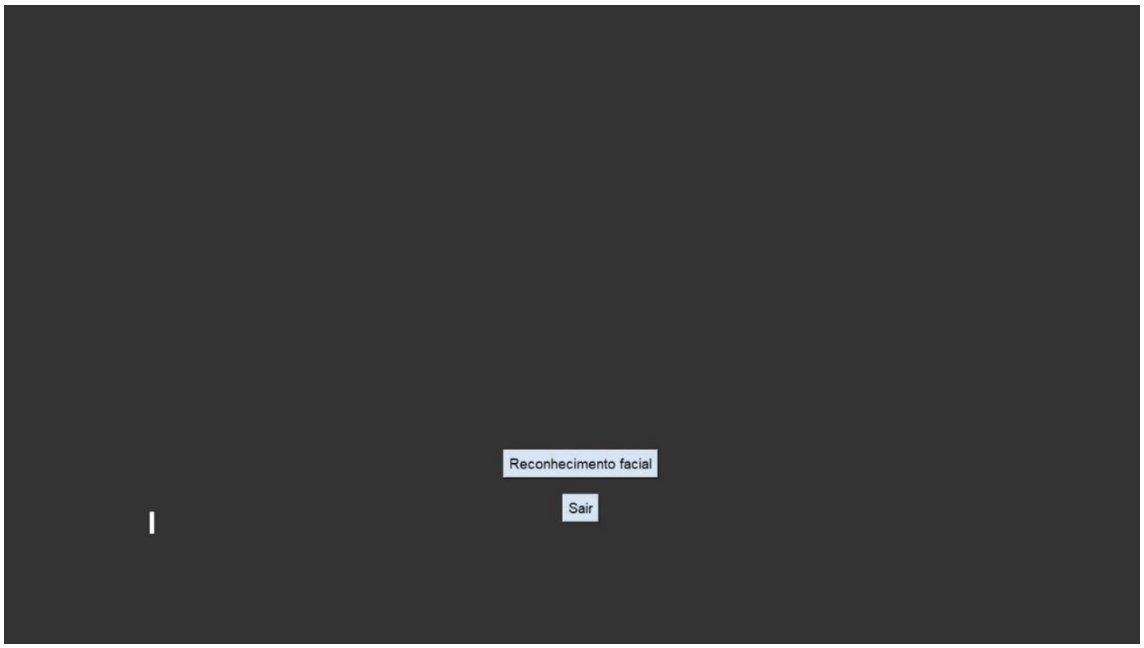
O projeto foi escrito em inglês para manter consistência.

Segue um print screen dos objetos criados neste projeto:



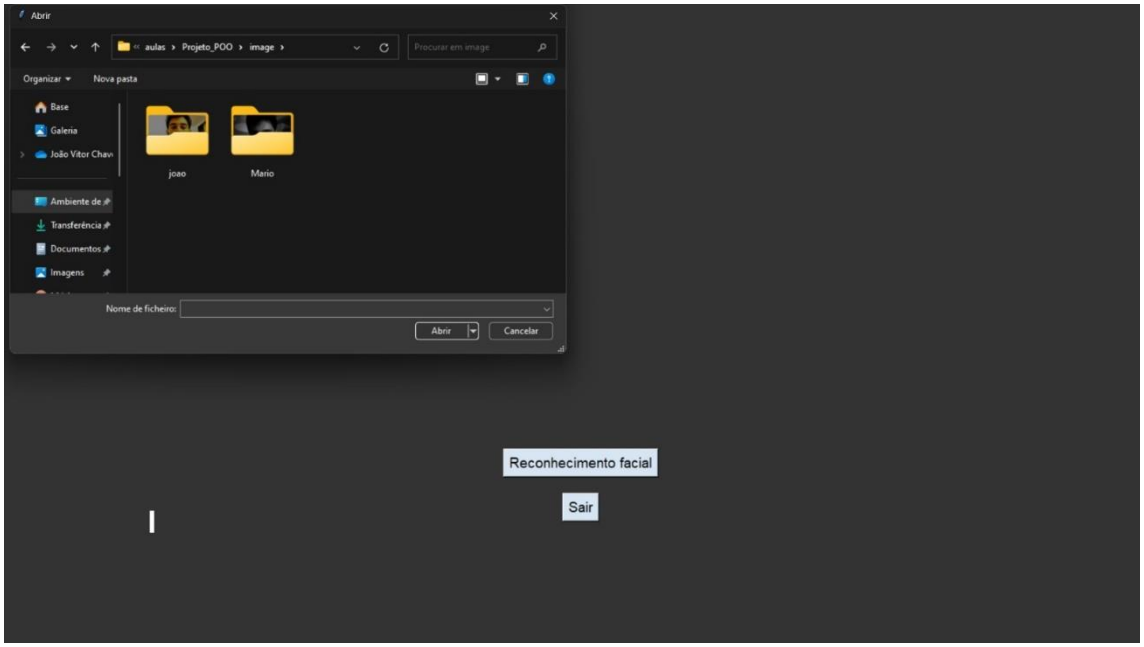
2.Funcionalidades (Passo a passo)

1. **O programa é iniciado e a janela principal da interface é exibida.**

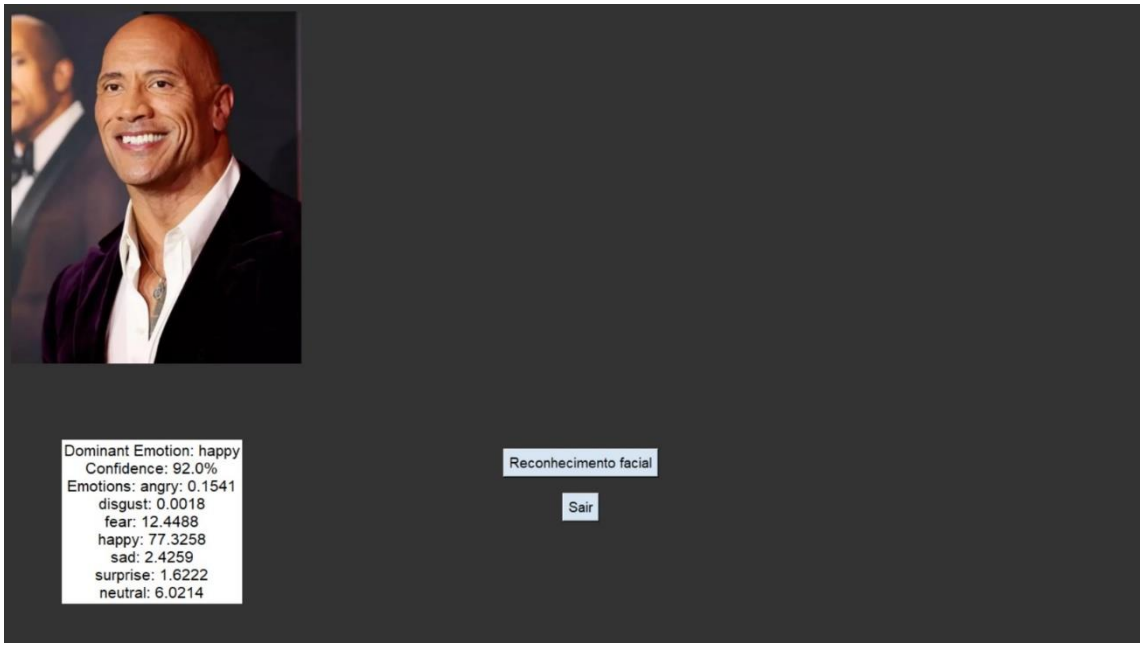


2. **O usuário clica no botão "Reconhecimento facial":**

- **Abre o seletor de arquivos para o usuário carregar uma imagem.**

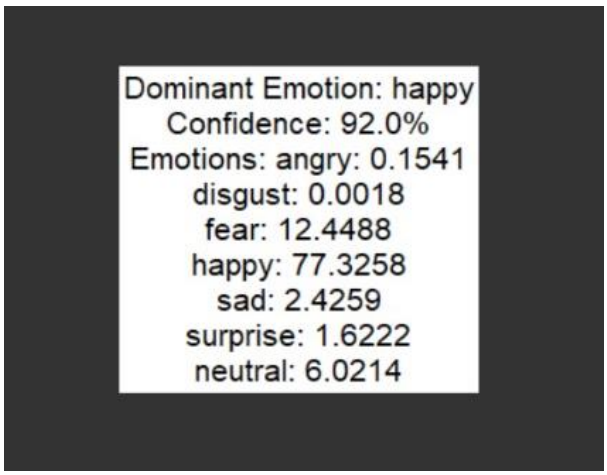


- **A imagem é exibida no canvas secundário.**



3.A análise de emoções é executada:

- **O DeepFace detecta as emoções e a emoção dominante.**
- **Os resultados são exibidos na interface.**



4. O usuário pode repetir o processo ou clicar em "Sair" para fechar o programa.

3.Desenvolvimento

Cada elemento do grupo fez a sua parte para criar este sistema. Começamos por discutir os requisitos para puder planear a estrutura do trabalho. No início tivemos dificuldades em ter uma ideia clara sobre como encontrar formas de implementar os diferentes requisitos do código.

Começamos por criar as interfaces e butões necessários sendo estes o “Facial recognition” e o “Exit” onde posteriormente o conteúdo inserido pelo usuário vai ser utilizado, como também as relações necessárias entre elas.

```
def __init__(self, master):
    self.master = master
    self.frame()
    self.image = None  # variável para manter a referência para imagem

    #Label para demonstrar emoções
    self.emotion_label = tk.Label(self.canvas, text="", font=('Arial', 16), bg="#FFFFFF", fg='#000000')
    self.canvas.create_window(200, 700, anchor='center', window=self.emotion_label)

def frame(self):
    # Criação da frame
    self.master.attributes('-fullscreen', True)
    self.master.title('Facial Recognition')

    # Background
    self.canvas = tk.Canvas(self.master, width=1920, height=1080, highlightbackground='#333333', bg="#333333")
    self.canvas.place(x=0, y=0)

    # Cria um background para onde a imagem vai ser colocada
    self.canvas2 = tk.Canvas(self.master, width=400, height=550, highlightbackground='#333333', bg="#333333")
    self.canvas2.place(x=0, y=0)

    # Botão para o reconhecimento facial
    self.recon_button = tk.Button(self.canvas, text="Reconhecimento facial", command=self.read_emotion, font=('Arial', 14), bg='#d8e6f4')
    self.canvas.create_window(780, 620, anchor='center', window=self.recon_button)

    # Botão para fecho do programa
    self.shutdown_button = tk.Button(self.canvas, text="Sair", font=('Arial', 14), command=self.master.destroy, bg='#d8e6f4')
    self.canvas.create_window(780, 680, anchor='center', window=self.shutdown_button)
```

De seguida, criamos as funções necessárias.

Método upload_img

Permite ao usuário selecionar uma imagem e exibi-la no canvas secundário:

1. **Seleção da Imagem:**
    - Abre o explorador de arquivos para o usuário escolher a imagem.
    - Cancela se o usuário não selecionar nada.
2. **Redimensionamento:**
    - Ajusta a imagem para caber no canvas secundário (400x550).
    - Mantém as proporções ao redimensionar.

3. **Exibição:**
    - Converte a imagem para um formato compatível com o Tkinter (ImageTk.PhotoImage).
    - Exibe no self.canvas2.
4. **Retorno do Caminho:**
    - Retorna o caminho da imagem carregada para uso em análises posteriores.

Método read_emotion

Analisa as emoções da imagem carregada utilizando o DeepFace:

1. **Carregamento da Imagem:**
    - Usa upload_img para selecionar a imagem.
    - Se nenhuma imagem for selecionada, a função é encerrada.
2. **Análise de Emoções:**
    - Lê a imagem com o OpenCV (cv2.imread).
    - Envia a imagem para o DeepFace, que retorna:
        - As emoções detectadas (com valores percentuais).
        - A emoção dominante.
        - A confiança de que o rosto foi identificado.
3. **Tratamento do Resultado:**
    - Se houver várias faces na imagem, utiliza apenas a primeira.
    - Armazena as emoções e arredonda os valores para 4 casas decimais.
4. **Exibição dos Resultados:**
    - Formata as informações:
        - Emoção dominante.
        - Confiança.
        - Todas as emoções e seus valores.

- Atualiza o texto do self.emotion_label para exibir os resultados na interface.

4.Contribuições

Todos deram o seu contributo no projeto. As tarefas foram trabalhadas maioritariamente em conjunto, porém, houve foco de ambos os integrantes em partes especificas do código das quais podemos descrever da seguinte forma:

- criação da frame → Mário
- Upload da imagem→ Mário
- Leitura das emoções → João

5.Conclusão

Consideramos que, apesar de não termos conseguido implementar o reconhecimento dos integrantes da equipa pelo programa, este projeto foi bem-sucedido pois conseguimos implementar a maioria dos requisitos solicitados.

Tivemos algumas dificuldades ao trabalhar com a junção da implementação da seleção de imagem e reconhecimento das emoções, contudo essas dificuldades foram ultrapassadas com a contribuição de todos.

Para terminar, aprendemos bastante com este projeto, quer a nível de programação quer a nível de organização/gestão do projeto.