**Nome: Johnatan Duarte Franco**

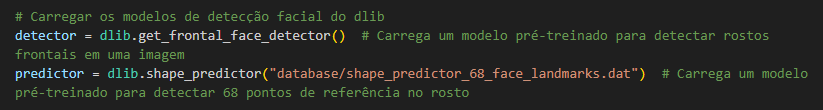
**Relatório: Substituição de Rostos entre Imagens**

**Introdução**

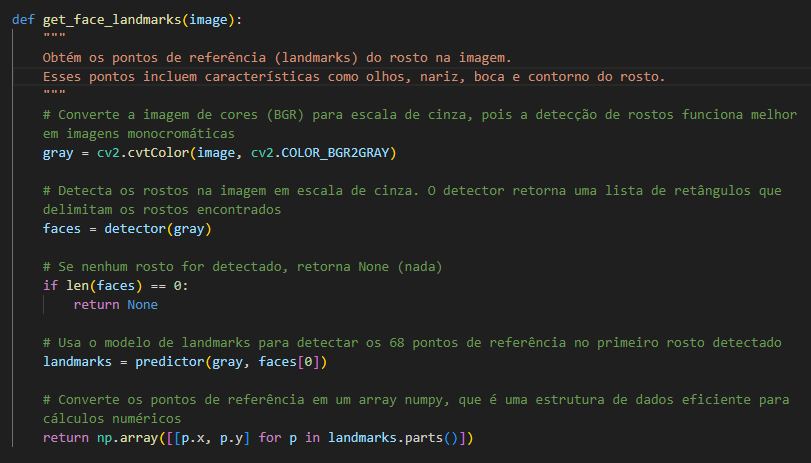
Este relatório descreve a implementação de uma aplicação de **substituição de rostos entre imagens**, utilizando técnicas de processamento de imagem e visão computacional. O objetivo é substituir o rosto de uma pessoa em uma imagem pelo rosto de outra pessoa, de forma que o resultado seja o mais natural possível. Para isso, foram utilizadas as bibliotecas **OpenCV** e **dlib**, que fornecem ferramentas para detecção facial, identificação de pontos de referência (landmarks) e transformação de imagens.

**Metodologia**

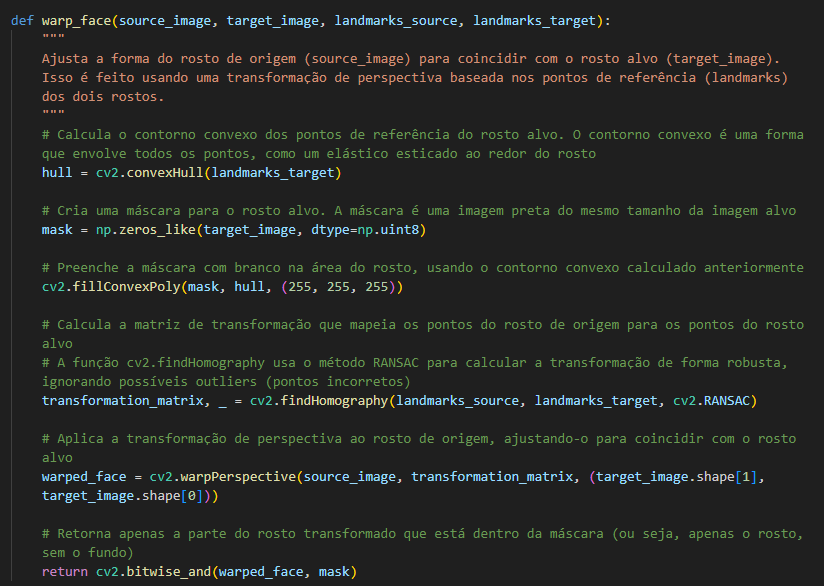
O processo foi dividido em etapas, conforme descrito abaixo:

**1. Detecção de Rostos e Pontos de Referência (Landmarks)  
  
**

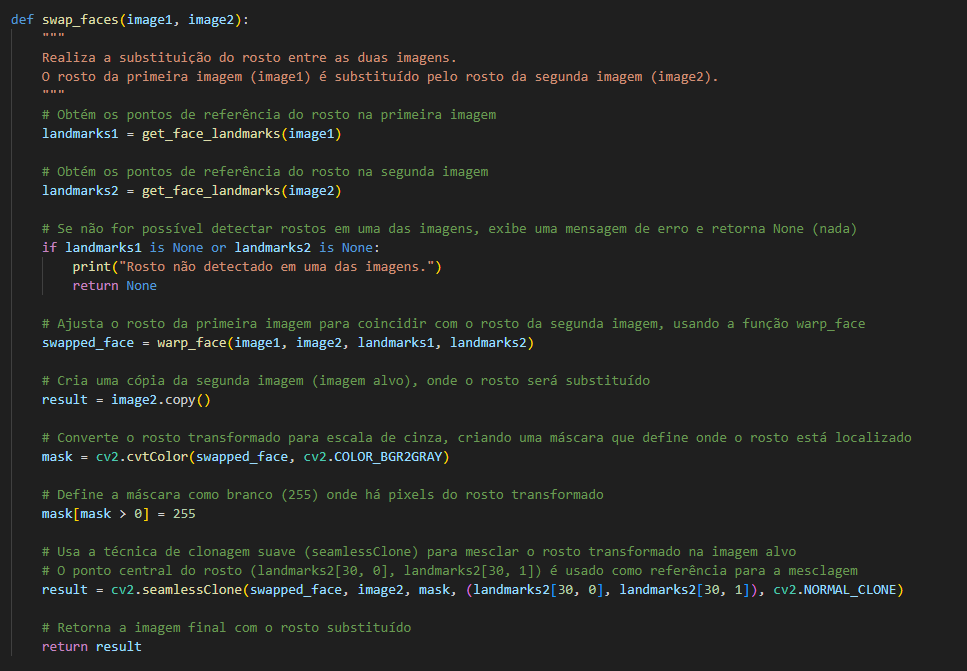
* **Função**: dlib.get\_frontal\_face\_detector()
  + **O que faz?**: Carrega um modelo pré-treinado para detectar rostos em uma imagem.
  + **Parâmetros**: Nenhum. O modelo já está pré-configurado para detectar rostos frontais.
  + **Explicação**: Este detector é usado para identificar a região da imagem onde há um rosto.
* **Função**: dlib.shape\_predictor()
  + **O que faz?**: Carrega um modelo pré-treinado para detectar 68 pontos de referência (landmarks) no rosto.
  + **Parâmetros**:
    - "database/shape\_predictor\_68\_face\_landmarks.dat": Caminho para o arquivo de modelo que contém os dados dos landmarks.
  + **Explicação**: Esses pontos de referência são usados para identificar características faciais, como olhos, nariz, boca e contorno do rosto.

**2. Obtenção dos Pontos de Referência (Landmarks)  
  
**

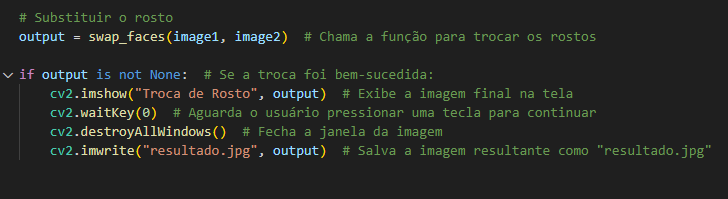
* **Função**: cv2.cvtColor()
  + **O que faz?**: Converte a imagem de cores (BGR) para escala de cinza.
  + **Parâmetros**:
    - image: Imagem de entrada.
    - cv2.COLOR\_BGR2GRAY: Modo de conversão para escala de cinza.
  + **Explicação**: A conversão para escala de cinza facilita a detecção de rostos e landmarks.
* **Função**: detector(gray)
  + **O que faz?**: Detecta rostos na imagem em escala de cinza.
  + **Parâmetros**:
    - gray: Imagem em escala de cinza.
  + **Explicação**: Retorna uma lista de rostos detectados. Se nenhum rosto for encontrado, a função retorna None.
* **Função**: predictor(gray, faces[0])
  + **O que faz?**: Detecta os 68 pontos de referência no rosto detectado.
  + **Parâmetros**:
    - gray: Imagem em escala de cinza.
    - faces[0]: O primeiro rosto detectado na imagem.
  + **Explicação**: Retorna os landmarks como uma lista de coordenadas (x, y).

**3. Transformação do Rosto  
  
**

* **Função**: cv2.convexHull()
  + **O que faz?**: Calcula o contorno convexo dos pontos de referência do rosto alvo.
  + **Parâmetros**:
    - landmarks\_target: Pontos de referência do rosto alvo.
  + **Explicação**: O contorno convexo é usado para criar uma máscara que define a região do rosto.
* **Função**: cv2.findHomography()
  + **O que faz?**: Calcula a matriz de transformação que mapeia os pontos do rosto de origem para o rosto alvo.
  + **Parâmetros**:
    - landmarks\_source: Pontos de referência do rosto de origem.
    - landmarks\_target: Pontos de referência do rosto alvo.
    - cv2.RANSAC: Método robusto para estimar a transformação.
  + **Explicação**: A matriz de transformação é usada para ajustar o rosto de origem ao rosto alvo.
* **Função**: cv2.warpPerspective()
  + **O que faz?**: Aplica a transformação de perspectiva ao rosto de origem.
  + **Parâmetros**:
    - source\_image: Imagem do rosto de origem.
    - transformation\_matrix: Matriz de transformação calculada.
    - (target\_image.shape[1], target\_image.shape[0]): Dimensões da imagem alvo.
  + **Explicação**: O rosto de origem é transformado para coincidir com o rosto alvo.

**4. Substituição do Rosto  
  
**

* **Função**: cv2.seamlessClone()
  + **O que faz?**: Mescla o rosto transformado na imagem alvo de forma suave.
  + **Parâmetros**:
    - swapped\_face: Rosto transformado.
    - image2: Imagem alvo.
    - mask: Máscara que define a região do rosto.
    - (landmarks2[30, 0], landmarks2[30, 1]): Ponto central do rosto alvo (nariz).
    - cv2.NORMAL\_CLONE: Método de clonagem que preserva a textura e a iluminação.
  + **Explicação**: A função mescla o rosto transformado na imagem alvo, garantindo que o resultado seja natural.

**5. Visualização e Salvamento do Resultado  
  
**

* **Função**: cv2.imshow()
  + **O que faz?**: Exibe a imagem resultante na tela.
  + **Parâmetros**:
    - "Troca de Rosto": Nome da janela.
    - output: Imagem resultante.
  + **Explicação**: Permite visualizar o resultado da substituição de rostos.
* **Função**: cv2.imwrite()
  + **O que faz?**: Salva a imagem resultante em um arquivo.
  + **Parâmetros**:
    - "resultado.jpg": Nome do arquivo de saída.
    - output: Imagem resultante.
  + **Explicação**: Salva a imagem final para uso posterior.

**Resultados**

O projeto foi capaz de substituir o rosto de uma pessoa em uma imagem pelo rosto de outra pessoa, gerando um resultado visualmente natural. A aplicação utiliza técnicas avançadas de processamento de imagem, como detecção de landmarks, transformação de perspectiva e clonagem suave, para garantir que a substituição seja realista.



**Conclusão**

A aplicação desenvolvida demonstra a eficácia das técnicas de processamento de imagem e visão computacional para a substituição de rostos entre imagens. O uso de bibliotecas como OpenCV e dlib facilitou a implementação de algoritmos complexos, como a detecção de landmarks e a transformação de perspectiva. O resultado final é uma ferramenta que pode ser usada para criar efeitos visuais divertidos e educativos, além de servir como base para projetos mais avançados em visão computacional.

**Referencias**

[Index of /files](https://dlib.net/files/)  
[Facial landmark Detection com OpenCV | by Jonys Arcanjo | Data Hackers | Medium](https://medium.com/data-hackers/facial-landmark-detection-com-opencv-744075300cfb)