SISTEMA DE RESERVAS DE HOTEL



UNIVERSIDADE AUTONOMA DE LISBOA

PROGRAMAÇÃO ORIENTADA A OBJECTOS

TRABALHO REALIZADO POR:

GUILHERME RESTOLHO Nº30012684

GONÇALO MIRANDA Nº30012356

AFONSO SOEIRO Nº30012583

# Índice

[Índice 2](#_Toc187508201)

[1. Introdução 3](#_Toc187508202)

[2. Preparação dos Dados 3](#_Toc187508203)

[3. Reconhecimento facial com confiança 3](#_Toc187508204)

[4. Reconhecimento de emoções com confiança 4](#_Toc187508205)

[5. Alteração e reconhecimento de fotos modificadas 5](#_Toc187508206)

[6. Desenho orientado a objetos 6](#_Toc187508207)

[7. Testes e qualidade do código 7](#_Toc187508208)

[8. Contribuições individuais 8](#_Toc187508209)

[9. Conclusão 8](#_Toc187508210)

# 1. Introdução

Este relatório descreve o desenvolvimento de um sistema de Reconhecimento Facial e Emoções, implementado em Python utilizando bibliotecas como OpenCV e DeepFace. O objetivo do projeto foi criar um sistema capaz de reconhecer rostos e identificar emoções com confiança, além de avaliar a robustez do sistema em cenários com alterações nas imagens. Adicionalmente, foram aplicados conceitos de programação orientada a objetos para modularidade e reutilização, acompanhados de uma interface gráfica para interação com o usuário.

# 2. Preparação dos Dados

Os dados foram organizados para garantir variedade e realismo. Cada aluno contribuiu com um conjunto de cinco imagens faciais diferentes, registradas em diferentes condições (expressões, iluminação, e ângulos) para garantir a robustez do sistema. Além disso, alterações como pixelização, adição de ruído e mudanças de cores foram aplicadas a algumas imagens, permitindo avaliar o impacto dessas modificações no desempenho do reconhecimento facial e na detecção de emoções.

# 3. Reconhecimento facial com confiança

Nesta etapa do projeto, foi implementado um sistema de reconhecimento facial utilizando o modelo LBPH, disponível na biblioteca OpenCV. Este modelo foi escolhido pela sua simplicidade e eficiência em cenários de reconhecimento facial, especialmente para conjuntos de dados relativamente pequenos.

O sistema foi configurado para identificar rostos conhecidos, atribuindo um grau de confiança a cada previsão. Esta confiança reflete a similaridade entre a imagem analisada e as imagens do conjunto de treino. Quanto menor o valor da confiança retornada, maior a precisão da identificação.

**Pré-processamento das Imagens**

* As imagens foram organizadas numa pasta(imgs) para serem mais fáceis de aceder
* Foram aplicadas técnicas de pré-processamento, como redimensionamento e conversão para um tom de cinza, para facilitar o modelo.

**Treinamento do Modelo**

* O modelo LBPH foi inicializado com a função cv2.face.LBPHFaceRecognizer\_create() e treinado com as imagens de treino.
* As labels associadas a cada imagem foram representadas por números inteiros, mapeadas para os nomes correspondentes.

**Reconhecimento e Geração de Confiança**

* O reconhecimento foi realizado com o método predict(), que retorna a label prevista e a confiança associada à previsão.
* O sistema mostra a label do indivíduo reconhecido, bem como o grau de confiança, permitindo avaliar a qualidade do reconhecimento. No entanto, optamos por apenas mostrar o grau de confiança e não mostrar a label identificada na nossa aplicação.

**Resultados Obtidos:**

Nos testes realizados, o sistema demonstrou boa precisão para identificar emoções em imagens de rostos capturadas em condições normais. O grau de confiança fornecido pelo modelo foi útil para interpretar a confiabilidade das predições.

# 4. Reconhecimento de emoções com confiança

O sistema também foi projetado para reconhecer as emoções faciais dos indivíduos presentes nas imagens. Para isso, foi utilizado a biblioteca **DeepFace**, que permite identificar emoções como felicidade, tristeza, raiva, surpresa, entre outras, juntamente com o grau de confiança associado a cada emoção.

**Criação da Classe Emotions**

* A funcionalidade de reconhecimento de emoções foi encapsulada na classe Emotions, garantindo modularidade e facilidade de reutilização.
* O construtor da classe (\_\_init\_\_) recebe o path da imagem e o widget onde o resultado será exibido (como parte da interface gráfica).

**Análise de Emoções**

* O método DeepFace.analyze foi utilizado para analisar a imagem especificada e retornar um dicionário que contém as emoções detectadas e os seus respectivos graus de confiança.
* Caso a imagem contenha vários rostos, o sistema trata este cenário extraindo apenas as informações do primeiro rosto detectado.

**Resultados Obtidos:**

Nos testes realizados, o sistema foi capaz de identificar emoções predominantes com boa precisão. As emoções foram exibidas com suas respectivas porcentagens de confiança, permitindo avaliar a confiabilidade da análise.

# 5. Alteração e reconhecimento de fotos modificadas

Uma etapa importante e um pouco mais complicada do projeto foi a implementação de funcionalidades para modificar imagens e avaliar o impacto dessas alterações no reconhecimento facial e na detecção de emoções. Para isso, foi desenvolvida uma função que aplica pixelização e alteração de cores às imagens carregadas, permitindo simular cenários desafiadores para o sistema.

**Criação da Função pixelize\_image**

* A função pixelize\_image foi implementada para realizar duas alterações principais nas imagens:
  + **Pixelização:** Reduz a resolução da imagem e, em seguida, restaura-a ao tamanho original, criando um efeito de pixels.
  + **Alteração de Cor:** Aplica um tom de cor adicional à imagem, modificando os valores RGB de cada pixel.

**Detalhes da Alteração**

* **Pixelização:** A resolução da imagem foi reduzida proporcionalmente ao valor definido no parâmetro pixel\_size, criando uma aparência de blocos de cores uniformes.
* **Alteração de Cor:** A função adiciona valores constantes a cada canal RGB, alterando o tom geral da imagem. A função garante que os valores dos pixels permaneçam no intervalo permitido (0-255) para evitar erros.

**Exibição na Interface Gráfica**

* A imagem modificada foi redimensionada para um tamanho fixo (300x300 pixels) para exibição consistente na interface gráfica.
* A atualização foi integrada ao widget do Tkinter, permitindo que o usuário visualize imediatamente as alterações aplicadas.

**Resultados Obtidos**

Os testes realizados demonstraram que o sistema consegue aplicar pixelização e alterações de cores de forma eficiente. Após as alterações, a imagem modificada foi novamente analisada pelo sistema de reconhecimento facial e de emoções, permitindo avaliar o impacto das mudanças.

# 6. Desenho orientado a objetos

O sistema foi projetado com base nos conceitos de Programação Orientada a Objetos, visando modularidade, reutilização e clareza. A implementação utilizou três classes principais para dividir as responsabilidades do sistema:

**Classe Emotions**

* Modularidade: Encapsula toda a lógica relacionada à análise de emoções numa classe separada.
* Reutilização: Pode ser integrada facilmente a outros sistemas que necessitem de reconhecimento emocional.

**Classe ImageApp**

* Responsável pela interface gráfica, utilizando métodos para interagir com o usuário.
* Métodos como pixelize\_image, recognize\_image, e emotions\_image seguem o princípio de **encapsulamento**, pois cada funcionalidade está bem isolada dentro da classe.

**Classe Facial\_recognition**

* Projetada para lidar exclusivamente com o reconhecimento facial, promovendo a separação de responsabilidades.

**Conceitos Utilizados:**

* **Encapsulamento:** Cada classe contém métodos específicos para lidar com as suas respectivas funcionalidades.
* **Reutilização:** A modularidade do sistema permite que as classes sejam reutilizadas em diferentes partes do projeto ou em futuros desenvolvimentos.
* **Clareza:** A separação de responsabilidades entre interface gráfica, reconhecimento facial e análise de emoções torna o código mais compreensível e de fácil manutenção.

**Diagrama UML:**

A screenshot of a computer

Description automatically generated

# 7. Testes e qualidade do código

A qualidade do código e a garantia do funcionamento correto do sistema foram aspectos fundamentais durante o desenvolvimento deste projeto. Para isso, foram seguidas boas práticas de programação e realizados testes extensivos das funcionalidades principais.

**Clareza e Organização do Código:**

* O código foi estruturado em módulos, separando as funcionalidades da interface gráfica, reconhecimento facial e análise de emoções.
* Foram utilizadas classes para encapsular as responsabilidades de forma modular e reutilizável, seguindo os princípios da programação orientada a objetos.

**Documentação:**

* Todas as classes e métodos principais foram comentados, explicando as suas finalidades e parâmetros. Isto facilita a manutenção e o entendimento do código.

**Tratamento de Erros:**

* Mecanismos de exceções foram implementados em métodos sensíveis, como os que processam imagens e executam análises de emoções, garantindo que falhas sejam tratadas adequadamente sem interromper o funcionamento do sistema.

Para garantir a funcionalidade e precisão do sistema, foi elaborada uma série de testes com foco nos principais componentes:

**Reconhecimento Facial:**

* Testes realizados com imagens não utilizadas no treino, incluindo variações de iluminação, ângulos e expressões faciais.
* O desempenho foi avaliado com base no grau de confiança retornado pelo sistema.

**Reconhecimento de Emoções:**

* Foram utilizadas imagens com diferentes expressões faciais para validar a detecção de emoções e o grau de confiança associado.
* Casos de falha foram analisados para melhorar a robustez do sistema.

**Manipulação de Fotos:**

* Testes realizados com imagens pixelizadas e com alterações de cor, avaliando o impacto dessas modificações no reconhecimento facial e na detecção de emoções.

**Interface Gráfica:**

* A interface foi testada para verificar a funcionalidade dos botões, carregamento de imagens e exibição dos resultados.
* Foram realizados testes manuais em diferentes computadores para validar a usabilidade.

Concluindo, os testes demonstraram que o sistema atende aos requisitos propostos, apresentando resultados consistentes em reconhecimento facial e análise de emoções. As boas práticas de código e os esforços em garantir a qualidade do código tornam o sistema confiável e de fácil manutenção.

# 8. Contribuições individuais

Gonçalo Miranda: Reconhecimento facial e reconhecimento emoções

Guilherme Restolho: Distorção imagem

Afonso Soeiro: Interface gráfica

# 9. Conclusão

O desenvolvimento deste projeto de Reconhecimento Facial e de Emoções em Python foi uma oportunidade valiosa para aplicar conceitos de programação, inteligência artificial e design de sistemas. O sistema final atendeu aos requisitos propostos, implementando funcionalidades como reconhecimento facial, análise de emoções, manipulação de imagens e interação por meio de uma interface gráfica intuitiva.

Durante o processo, os desafios enfrentados, como a adaptação do modelo a diferentes condições de imagem e o impacto de alterações visuais nos resultados, foram abordados com soluções que demonstraram a eficácia dos métodos utilizados. A abordagem modular, baseada em conceitos de programação orientada a objetos, garantiu um código claro, organizado e facilmente extensível.

Os testes realizados confirmaram que o sistema é funcional e apresenta resultados satisfatórios em diferentes cenários, com graus de confiança que validam a confiabilidade das predições. Além disso, a documentação e a separação de responsabilidades tornaram o projeto bem estruturado e preparado para futuras expansões.

Em suma, o projeto não apenas atingiu os objetivos propostos, mas também proporcionou uma aprendizagem significativa no desenvolvimento de sistemas inteligentes e na aplicação de boas práticas de codificação.