

INSTITUO ATLÂNTICO AVANTI
CURSO BÁSICO DE MACHINE LEARNING

**RESUMO DO MÓDULO 3 SOBRE AS FASES DA VISÃO
COMPUTACIONAL**

LÓREN DUTRA SILVA

SÃO LUÍS - MA
2023

LÓREN DUTRA SILVA

**RESUMO DO MÓDULO 3 SOBRE AS FASES DA VISÃO
COMPUTACIONAL**

Resumo do módulo 3 declarado como
critério para a obtenção da nota final do
curso.

Professor: Renê Calixto

Tutor(a): Angélica Viana

SÃO LUÍS – MA

2023

LISTA DE ILUSTRAÇÕES

Figura 1 – Exemplo de código para o pré-processamento de uma imagem	5
Figura 2 – Exemplo de código para a segmentação de uma imagem	6
Figura 3 – Exemplo de código para a detecção/classificação de uma imagem	7

SUMÁRIO

1	INTRODUÇÃO	4
2	PRÉ-PROCESSAMENTO DE IMAGENS	5
2.1	INTRODUÇÃO.....	5
2.2	EXEMPLOS DE BIBLIOTECAS	5
2.3	EXEMPLO PRÁTICO E EXPLICAÇÃO	5
3	SEGMENTAÇÃO DE IMAGENS	6
3.1	INTRODUÇÃO.....	6
3.2	EXEMPLOS DE BIBLIOTECAS	6
3.3	EXEMPLO PRÁTICO E EXPLICAÇÃO.....	6
4	DETECÇÃO/CLASSIFICAÇÃO DE IMAGENS	7
4.1	INTRODUÇÃO.....	7
4.2	EXEMPLOS DE BIBLIOTECAS	7
4.3	EXEMPLO PRÁTICO E EXPLICAÇÃO	7
5	CONSIDERAÇÕES FINAIS.....	8
	REFERÊNCIAS.....	9

1 INTRODUÇÃO

Este trabalho explora três elementos fundamentais da visão computacional: pré-processamento de imagens, segmentação de imagens e detecção/classificação de objetos. Cada seção terá sua própria introdução para destacar a relevância dessas etapas.

Ao longo do texto, serão apresentadas as bibliotecas comumente utilizadas para implementar essas técnicas no ambiente Python. No final de cada parte, será apresentado um exemplo prático acompanhado por uma explicação correspondente ao código. O objetivo é fornecer uma compreensão abrangente de cada fase, desde a teoria até a aplicação prática. Este trabalho serve como um guia resumido do módulo 3 sobre as fases da visão computacional, enfatizando a importância e aplicação dessas etapas na análise de dados visuais.

2 PRÉ-PROCESSAMENTO DE IMAGENS

2.1 INTRODUÇÃO

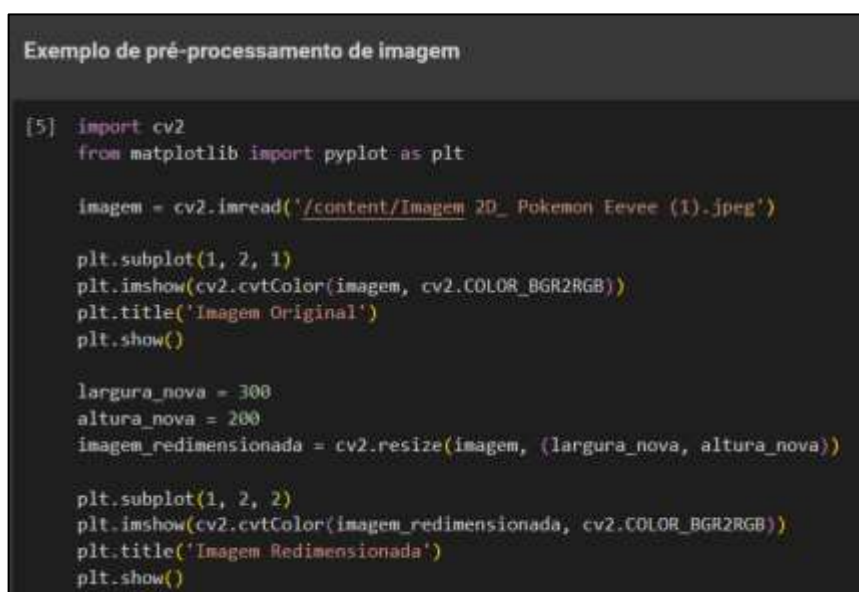
O pré-processamento de imagens é uma etapa crucial na visão computacional, onde aprimoramos a qualidade das informações antes de serem processadas. Envolve técnicas como normalização, redimensionamento e remoção de ruídos para otimizar algoritmos de reconhecimento e segmentação. Essencial para garantir resultados precisos em tarefas relacionadas à análise de imagens.

2.2 EXEMPLOS DE BIBLIOTECAS

O OpenCV e o Pillow, duas bibliotecas Python, são poderosas para manipulação e pré-processamento de imagens. O OpenCV é especializado em visão computacional, proporcionando diversas funcionalidades para detecção de objetos e reconhecimento de padrões. Enquanto isso, o Pillow é conhecido por sua facilidade de uso para operações básicas como redimensionamento e conversão de formatos.

2.3 EXEMPLO PRÁTICO E EXPLICAÇÃO

Figura 1 – Exemplo de código para o pré-processamento de uma imagem

A imagem mostra um editor de código com o título "Exemplo de pré-processamento de imagem". O código Python dentro do editor realiza as seguintes ações: importa as bibliotecas cv2 e pyplot; lê uma imagem de um caminho específico; cria um subplot para exibir a imagem original convertida para o formato BGR2RGB; define novas dimensões (300x200); redimensiona a imagem usando cv2.resize; cria outro subplot para exibir a imagem redimensionada convertida para BGR2RGB. O código é o seguinte:

```
[5] import cv2
from matplotlib import pyplot as plt

imagem = cv2.imread('/content/Imagem 20_Pokemon Eevee (1).jpeg')

plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB))
plt.title('Imagem Original')
plt.show()

largura_nova = 300
altura_nova = 200
imagem_redimensionada = cv2.resize(imagem, (largura_nova, altura_nova))

plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(imagem_redimensionada, cv2.COLOR_BGR2RGB))
plt.title('Imagem Redimensionada')
plt.show()
```

Fonte: Lóren Dutra Silva (2023)

Neste exemplo, a imagem é carregada usando o OpenCV, e em seguida, redimensionada para uma nova largura e altura. A imagem original e a redimensionada serão exibidas para visualização. Você pode ajustar os valores de **largura_nova** e **altura_nova** de acordo com as necessidades.

3 SEGMENTAÇÃO DE IMAGENS

3.1 INTRODUÇÃO

A segmentação de imagens é uma área crucial na visão computacional que visa dividir uma imagem em regiões significativas ou objetos. O objetivo é identificar áreas de interesse, sendo essencial em aplicações como reconhecimento de objetos, diagnóstico médico e veículos autônomos. Utilizando métodos que vão desde abordagens simples, como limiarização.

3.2 EXEMPLOS DE BIBLIOTECAS

Para segmentação de imagens, duas bibliotecas amplamente utilizadas são o Scikit-image e o OpenCV. O scikit-image oferece uma variedade de técnicas, desde segmentação baseada em região até métodos de contornos. Por outro lado, o OpenCV, fornece ferramentas robustas específicas para segmentação de imagens, sendo uma escolha popular em diversas aplicações.

3.3 EXEMPLO PRÁTICO E EXPLICAÇÃO

Figura 2 – Exemplo de código para a segmentação de uma imagem

A imagem mostra um editor de código com o título "Exemplo de segmentação de uma imagem". O código Python utiliza as bibliotecas cv2 e matplotlib.pyplot para carregar uma imagem de ressonância dos rins, convertê-la para escala de cinza e binarizá-la. A binarização transforma pixels abaixo de 127 para preto e acima de 127 para branco. A imagem original colorida e a imagem binarizada em escala de cinza serão exibidas.

```
import cv2
import matplotlib.pyplot as plt

imagem = cv2.imread('/content/Ressonancia dos rins.jpg', cv2.IMREAD_GRAYSCALE)
_, imagem_binarizada = cv2.threshold(imagem, 127, 255, cv2.THRESH_BINARY)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB))
plt.title('Imagem Original')

plt.subplot(1, 2, 2)
plt.imshow(imagem_binarizada, cmap='gray')
plt.title('Imagem Binarizada')

plt.show()
```

Fonte: Lóren Dutra Silva (2023)

Este código Python utiliza as bibliotecas OpenCV e Matplotlib para carregar uma imagem de ressonância dos rins, convertê-la para escala de cinza e binarizá-la. A binarização transforma pixels abaixo de 127 para preto e acima de 127 para branco. A imagem original colorida e a imagem binarizada em escala de cinza serão exibidas.

4 DETECÇÃO/CLASSIFICAÇÃO DE IMAGENS

4.1 INTRODUÇÃO

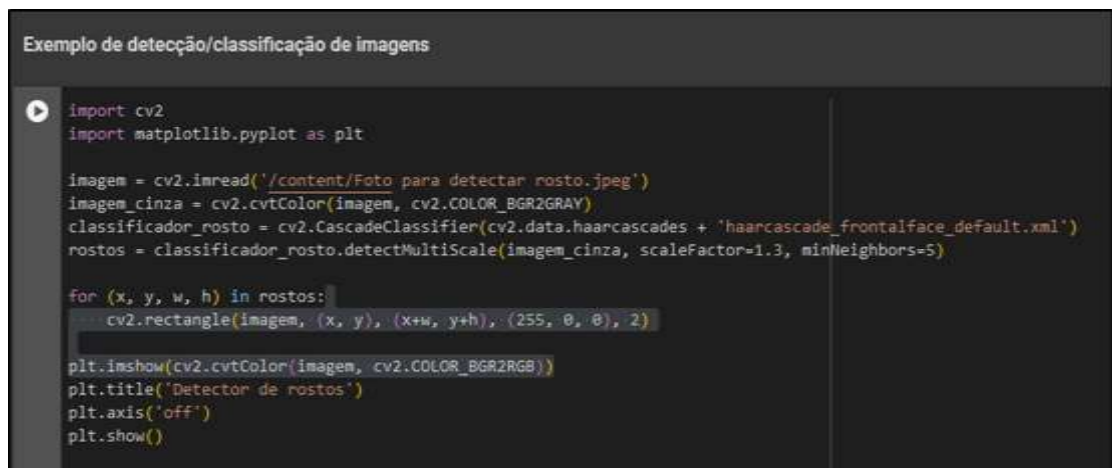
Detecção/classificação de imagens é crucial na visão computacional, identificando e categorizando objetos em imagens. Na detecção, localizamos objetos e delimitamos, enquanto na classificação, atribuímos rótulos. Essas tarefas são essenciais em áreas como reconhecimento facial e veículos autônomos. Algoritmos avançados, como redes neurais convolucionais, impulsionam a precisão, sendo vitais para a inteligência artificial, automação e diversas indústrias.

4.2 EXEMPLOS DE BIBLIOTECAS

Existem várias bibliotecas poderosas para detecção e classificação de imagens. Duas opções populares são TensorFlow e PyTorch. Ambas oferecem frameworks robustos para desenvolver modelos de aprendizado profundo.

4.3 EXEMPLO PRÁTICO E EXPLICAÇÃO

Figura 3 – Exemplo de código para a detecção/classificação de uma imagem

A screenshot of a code editor with a dark background. The title bar of the editor window reads "Exemplo de detecção/classificação de imagens". The code is written in Python and uses the OpenCV and Matplotlib libraries. It imports cv2 and matplotlib.pyplot as plt. The code reads an image from a file path, converts it to grayscale, and then uses a Haar Cascade classifier to detect faces. It iterates over the detected faces, drawing red rectangles around them. Finally, it displays the original image with the detected faces highlighted using plt.imshow and plt.show().

```
import cv2
import matplotlib.pyplot as plt

imagem = cv2.imread('/content/Foto para detectar rosto.jpeg')
imagem_cinza = cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)
classificador_rosto = cv2.CascadeClassifier(cv2.data.harcascades + 'haarcascade_frontalface_default.xml')
rostos = classificador_rosto.detectMultiScale(imagem_cinza, scaleFactor=1.3, minNeighbors=5)

for (x, y, w, h) in rostos:
    cv2.rectangle(imagem, (x, y), (x+w, y+h), (255, 0, 0), 2)

plt.imshow(cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB))
plt.title('Detector de rostos')
plt.axis('off')
plt.show()
```

Fonte: Lóren Dutra Silva (2023)

O código em Python utiliza as bibliotecas OpenCV e Matplotlib para detectar rostos em uma imagem. Ele lê uma imagem, a converte para tons de cinza e utiliza um classificador Haar Cascade pré-treinado para identificar rostos. Os rostos são destacados na imagem original com retângulos, e o resultado é exibido usando o Matplotlib. Em resumo, o código realiza a detecção e visualização de rostos em uma imagem.

5 CONSIDERAÇÕES FINAIS

Em um panorama abrangente, este trabalho explorou as etapas cruciais da visão computacional, percorrendo desde o pré-processamento de imagens até a segmentação e detecção/classificação de objetos. Ao mergulharmos nas bibliotecas, como o OpenCV, desbravamos ferramentas poderosas para manipulação e análise visual em Python.

Este trabalho, resumindo o módulo 3 do curso básico de Machine Learning, oferece uma perspectiva compreensiva sobre como essas fases são interligadas e essenciais para desvendar o mundo que nos cerca, desempenhando um papel crucial na convergência entre o digital e o visual, moldando o presente e apontando para um futuro repleto de possibilidades inexploradas.

REFERÊNCIAS BIBLIOGRÁFICAS

BONATO, Cassiana da Silva; NETO, Roberto Mendes Finzi. **Etapas de pré-processamento de imagens nas técnicas de reconhecimento biométricas por digitais**. Disponível em:< file:///C:/Users/user/Downloads/etapas-de-pr%C3%A9-processamento-de-imagens-nas-t%C3%A9cnicas-de-reconhecimento-biom%C3%A9tricas-por-digitais-2011.pdf>. Acesso em 07 de dezembro de 2023.

Didática Tech. **Bibliotecas Python para Processamento de Imagens**. Disponível em:< <https://didatica.tech/bibliotecas-para-processamento-de-imagens/>>. Acesso em 07 de dezembro de 2023.

Didática Tech. **O que é Detecção de Objetos?**. Disponível em:< <https://didatica.tech/deteccao-de-objetos/>>. Acesso em 08 de dezembro de 2023.

LIMA, Acervo. **DIFERENÇA ENTRE PYTORCH E TENSORFLOW**. Disponível em:< <https://acervolima.com/diferenca-entre-pytorch-e-tensorflow/>>. Acesso em 09 de dezembro de 2023.

LIMA, Acervo. **PROCESSAMENTO DE IMAGENS COM SCIKIT-IMAGE EM PYTHON**. Disponível em:<<https://acervolima.com/processamento-de-imagens-com-scikit-image-em-python/>>. Acesso em 08 de dezembro de 2023.

Piemontez. **Identificação, Detecção, Reconhecimento e Segmentação de Imagem e Objetos**. Disponível em:< <https://visaocomputacional.com.br/identificacao-deteccao-reconhecimento-e-segmentacao-de-imagem-e-objetos/>>. Acesso em 07 de dezembro de 2023.