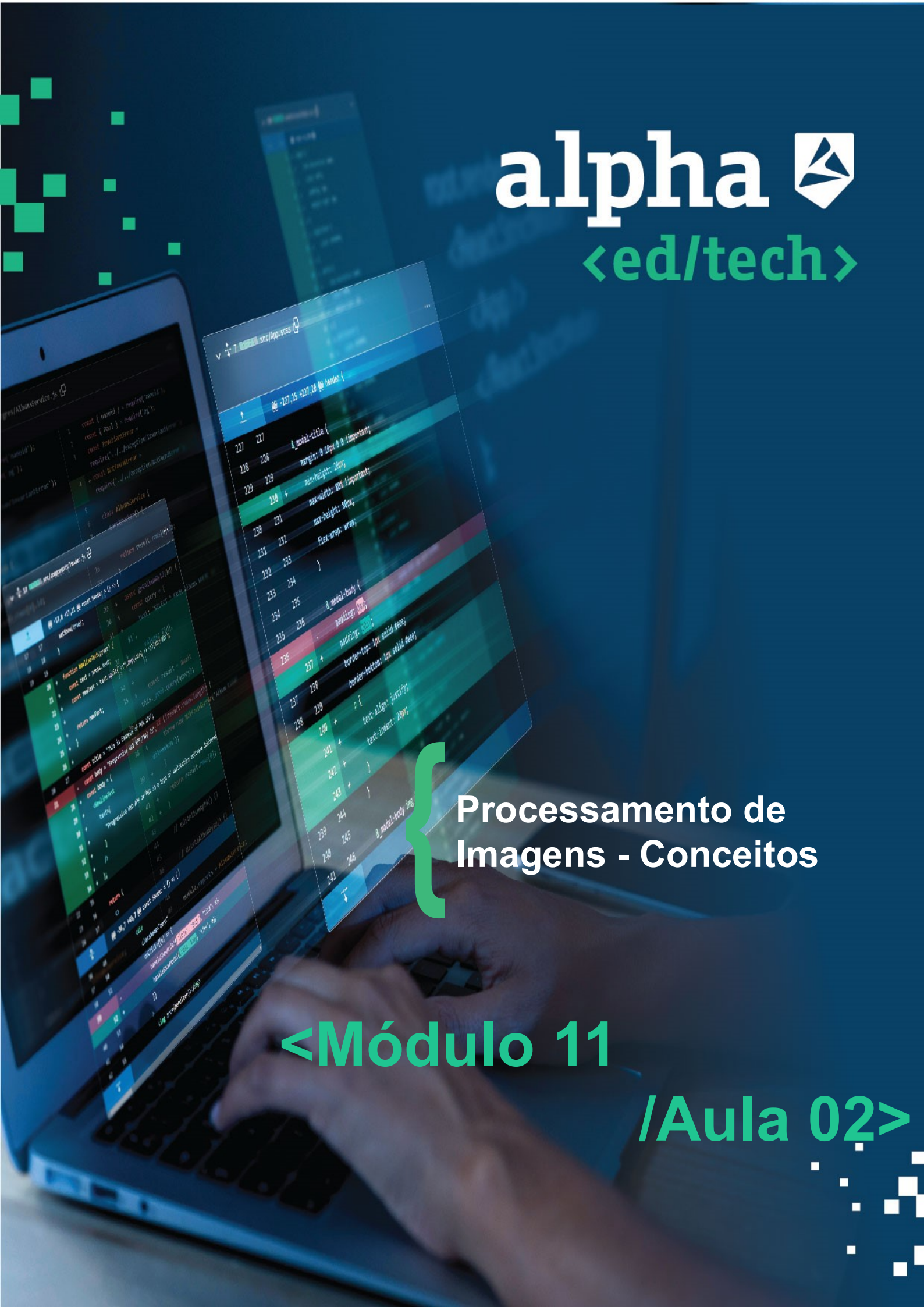
alpha
<ed/tech>
Processamento de Imagens - Conceitos
<Módulo 11
/Aula 02>

# Processamento de Imagens

## Conceitos

Conversão de padrões de cores em imagens usando bibliotecas em Python:

Para converter padrões de cores em imagens, podemos usar bibliotecas populares como PIL (Python Imaging Library) ou OpenCV.

Importar a biblioteca escolhida. Por exemplo, para PIL, você pode usar:

```
from PIL import Image
```

Crie uma nova imagem com as dimensões desejadas.
Isso pode ser feito usando a função `Image.new()` do PIL:

```
largura = 100
altura = 100
imagem = Image.new('RGB', (largura, altura), 'white')
```

Agora que temos uma imagem em branco, podemos manipular os pixels conforme o padrão de cores desejado. Por exemplo, para criar uma imagem com um padrão xadrez, podemos alternar entre duas cores:

```
for x in range(largura):
        for y in range(altura):
            if (x + y) % 2 == 0:
                imagem.putpixel((x, y), (0, 0, 0))  # Cor preta
            else:
                imagem.putpixel((x, y), (255, 255, 255))  # Cor branca
```

Salvar a imagem resultante usando o método `save()`:

```
imagem.save('padrao.png')
```

Isso cria uma imagem chamada "padrao.png" com o padrão de cores especificado. Você pode ajustar as cores e padrões conforme necessário, manipulando os valores dos pixels de acordo com suas preferências.

# Processamento de Imagens

A varredura tipo raster em uma imagem é um processo de percorrer cada pixel da imagem em uma sequência predefinida.

Importar bibliotecas, com a biblioteca PIL (Python Imaging Library), que permite trabalhar com imagens.

Carregue a imagem na qual você deseja realizar a varredura tipo raster.

Determine a largura e a altura da imagem para iterar sobre todos os pixels.

Para a "Varredura tipo raster" deve percorrer cada pixel da imagem em uma ordem predefinida, que geralmente é linha por linha ou coluna por coluna.

Dentro do loop, pode acessar e manipular as propriedades de cada pixel, como sua cor ou intensidade.

```
from PIL import Image

# Carregar a imagem
imagem = Image.open('imagem.jpg')

# Obter as dimensões da imagem
# (tupla com 2 valores que eh o retorno do metodo size)
largura, altura = imagem.size

# Varredura tipo raster
for y in range(altura):
    for x in range(largura):
        # Obter cor do pixel
        cor_pixel = imagem.getpixel((x, y))
        # Exemplo de manipulação de pixel: inverter cores
        nova_cor = (255 - cor_pixel[0], 255 - cor_pixel[1], 255 -
cor_pixel[2])
        # Definir nova cor do pixel
        imagem.putpixel((x, y), nova_cor)

# Salvar a imagem resultante
imagem.save('imagem_invertida.jpg')
```

Para transformar uma imagem colorida em negativo, pode inverter os valores de cada canal de cor (vermelho, verde e azul) para obter o efeito desejado.

Para cada pixel na imagem:
- Para o canal vermelho (R): novo_valor_R = 255 - valor_R
- Para o canal verde (G): novo_valor_G = 255 - valor_G
- Para o canal azul (B): novo_valor_B = 255 - valor_B

Essa fórmula simplesmente subtrai o valor atual de cada canal de cor de 255, o valor máximo possível para cada canal, resultando em uma inversão da cor. A inversão dos valores dos canais de cor cria o efeito de negativo, semelhante ao material fotográfico tradicional

# Processamento de Imagens

Na biblioteca OpenCV em Python, você pode realizar transformações automáticas de padrões, como converter de BGR para RGB ou de RGB para tons de cinza, usando funções específicas.

1. BGR para RGB: Para converter uma imagem de formato BGR para RGB, você pode usar a função `cv2.cvtColor()`, especificando o código de conversão `cv2.COLOR_BGR2RGB`.

```
import cv2

imagem_bgr = cv2.imread('imagem_bgr.jpg')
imagem_rgb = cv2.cvtColor(imagem_bgr, cv2.COLOR_BGR2RGB)
```

2. RGB para tons de cinza: Para converter uma imagem colorida em tons de cinza, você pode usar a função `cv2.cvtColor()`, especificando o código de conversão `cv2.COLOR_RGB2GRAY`.

```
import cv2

imagem_rgb = cv2.imread('imagem_rgb.jpg')
imagem_tons_de_cinza = cv2.cvtColor(imagem_rgb, cv2.COLOR_RGB2GRAY)
```

Essas são as transformações automáticas mais comuns, mas a biblioteca OpenCV oferece uma ampla gama de funções para manipulação de imagens, permitindo realizar uma variedade de outras transformações e operações de processamento de imagem.

Outro exemplo de conversão pode ser alterar a saturação das cores em uma imagem em Python, utilizando a biblioteca OpenCV. A saturação de uma cor determina a intensidade ou pureza dessa cor na imagem.

```
import cv2
import numpy as np

# Carregar a imagem
imagem = cv2.imread('imagem.jpg')

# Converter a imagem de BGR para HLS (Matiz (Hue), Luminância (Luminance),
Saturação (Saturation))
imagem_hls = cv2.cvtColor(imagem, cv2.COLOR_BGR2HLS)

# Ajustar a saturação
fator_saturacao = 1.5  # Ajuste o valor conforme necessário
imagem_hls[:, :, 2] = np.clip(imagem_hls[:, :, 2] * fator_saturacao, 0,
255)

# Converter a imagem de HLS de volta para BGR
nova_imagem = cv2.cvtColor(imagem_hls, cv2.COLOR_HLS2BGR)

# Exibir a imagem original e a imagem com saturação alterada
cv2.imshow('Imagem Original', imagem)
cv2.imshow('Imagem com Saturação Alterada', nova_imagem)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

# Processamento de Imagens

Neste exemplo, ajustamos a saturação multiplicando o canal de saturação (para o índice 2) pelos fatores de saturação desejados. Usamos `np.clip()` para garantir que os valores permaneçam dentro do intervalo válido (0 a 255).

Por fim, convertemos a imagem de volta para o formato BGR para visualização. Experimente diferentes valores para o parâmetro `fator_saturacao` para ver como eles afetam a saturação da imagem.

Outro padrão de cores YMC é um sistema de cores utilizado em alguns contextos de impressão, como na indústria gráfica. Ele é composto por três cores primárias: Y (amarelo), M (magenta) e C (ciano). Vamos entender cada uma dessas cores:

1. Ciano (C): O ciano é uma cor semelhante ao azul, mas mais clara e com uma tonalidade mais esverdeada. Na impressão, o ciano é usado para reproduzir tons de azul, verde e suas variações.

2. Magenta (M): O magenta é uma cor entre o vermelho e o roxo. Na impressão, o magenta é utilizado para reproduzir tons de vermelho, rosa e púrpura.

3. Amarelo (Y): O amarelo é uma cor primária que representa a luz combinada do vermelho e do verde. Na impressão, o amarelo é usado para reproduzir tons de amarelo, laranja e algumas variações de marrom.

Ao combinar essas três cores primárias em diferentes proporções, é possível criar uma ampla gama de cores. O padrão YMC é especialmente utilizado em impressoras e equipamentos de impressão digital, onde as cores são sobrepostas em camadas para criar uma variedade de tons e matizes. Por exemplo, para reproduzir uma cor como o azul, seria necessário usar uma combinação de ciano e magenta.

Em resumo, o padrão de cores YMC é fundamental para a reprodução precisa de cores na impressão, permitindo que uma ampla variedade de tons seja criada através da combinação de suas cores primárias: ciano, magenta e amarelo.