Identificação: Emanuelle Bavaresco Teodoro

Data: 17/05/2024

1. Introdução

O desafio a seguir foi proposto pela empresa Tarken e envolve a análise de uma imagem. O objetivo era contar o número de estrelas e meteoros, quantos meteoros caíram na água e encontrar a fase escondida. Para isso, um dos aspectos fundamentais na análise de imagens é a compreensão dos pixels que compõem a imagem e como eles podem ser analisados para extrair informações úteis.

2. Ambiente de desenvolvimento

O desenvolvimento e a execução do código foram realizados no Google Colab, utilizando Python na versão 3.10.12. E a biblioteca utilizada para a manipulação de imagens foi a Pillow (PIL), instalada e importada no ambiente Colab.

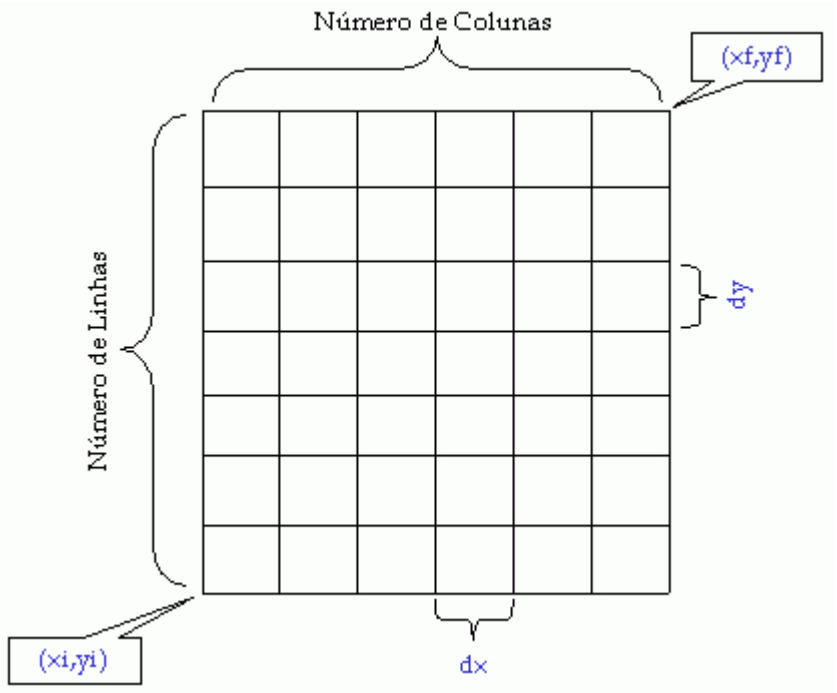
3. Pixels: A Unidade Básica da Imagem

Os pixels, abreviação de "picture elements", são os elementos básicos que compõem uma imagem digital. Cada pixel representa uma pequena área da imagem e contém informações sobre a cor e a intensidade luminosa dessa área, sendo definidos como um conjunto de elementos discretos e de tamanhos finitos. Em imagens coloridas, um pixel geralmente é composto por três canais de cor: vermelho, verde e azul (RGB). Em algumas imagens, um quarto canal alfa pode ser adicionado para representar transparência.

Sabe-se que uma imagem digital é representada por uma matriz como uma grade de pixels. Sendo que cada pixel tem uma posição única identificada por suas coordenadas (x, y):

- x representa a posição horizontal (a coluna) do pixel.
- y representa a posição vertical (a linha) do pixel.

Por exemplo, as coordenadas (0, 0) correspondem ao pixel no canto superior esquerdo da imagem. À medida que você move para a direita, o valor de x aumenta, e à medida que você move para baixo, o valor de y aumenta.Ao atribuir x, y = 10, 10, estamos simplesmente escolhendo um pixel localizado na coluna 10 e linha 10 da imagem.

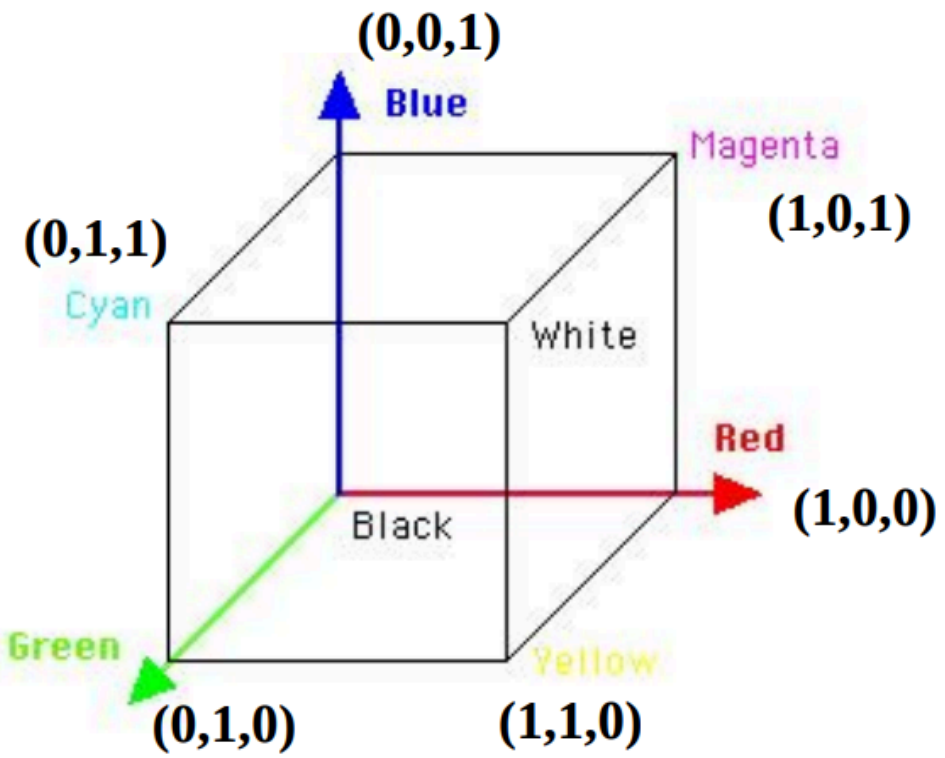


Neste contexto, a área de processamento de imagens se dedica à manipulação desses pixels com os seguintes objetivos: aprimoramento da imagem, remoção de elementos indesejados e extração de informações.

4. Representação

Internamente, os pixels são representados por valores numéricos que indicam a intensidade de cada canal de cor. Por exemplo, um pixel branco pode ser representado por (255, 255, 255), indicando intensidades máximas de vermelho, verde e azul. E um pixel preto seria representado por (0, 0, 0), indicando

intensidades mínimas em todos os canais. A imagem a seguir representa o modelo RGB:



5. Pontos conectados

Esta abordagem foi desenvolvida para verificar se uma estrela ou meteoro é composto por mais de um pixel e contabilizar esse agrupamento como uma entidade única, mesmo que sejam formados por vários pixels adjacentes.

Considerar a vizinhança em uma imagem digital refere-se à maneira como os pixels são considerados adjacentes uns aos outros. Sendo assim, o algoritmo examinará todos os pixels de uma região conectada antes de passar para as regiões vizinhas. Essa abordagem assegura que cada estrela ou meteoro, mesmo que formado por vários pixels, seja contado como um único elemento.

6. Contagem de Estrelas e meteoros

Para contar as estrelas e meteoros na imagem, o algoritmo percorre cada pixel da imagem. Sempre que encontrar um pixel branco (representando uma estrela) ou vermelho(representando uma meteoro), utiliza um algoritmo de região conectada para identificar todos os pixels adjacentes a ele para serem contabilizados como pertencentes a um elemento. Essa técnica garante que estrelas e meteoros compostos por múltiplos pixels sejam contadas como uma única entidade.

7. Contagem de Meteoros na Água

Para detectar uma possível queda de meteoros na água, o código verifica se há pixels de cor azul nos pixels da coluna que estão abaixo da trajetória do meteoro. Isso é feito percorrendo os pixels abaixo da posição do meteoro e verificando se algum deles corresponde à cor da água.

Se for encontrado um pixel azul, significa que o meteoro está caindo na água, e a função retorna de verdade. Caso contrário, retorna falso.

8. Metodologia e Aplicação

O código foi planejado em etapas, cada uma com uma função específica:

1. Carregar a imagem: A função “carregar_imagem” carrega a imagem especificada pelo caminho fornecido e retorna os pixels da imagem, bem como sua largura e altura.

2. Contagem: As funções “contar_estrelas” e “contar_meteoros” percorrem cada pixel da imagem e verifica se a cor do pixel é branca (255,255,255) ou vermelha(255,0,0). Se for, ela chama a função “regiao_conectada” para verificar se o pixel faz parte de uma região conectada de pixels brancos. Se for, a contagem de estrelas é incrementada.

3. Verificar se o meteoro está na água: A função “meteoro_na_agua” verifica se um meteoro está caindo na água. Ela percorre os pixels abaixo do meteoro até encontrar um pixel preto (0, 0, 0) ou um pixel azul (0, 0, 255). Se encontrar um pixel azul, significa que o meteoro está caindo na água.

4. Verificar regiões conectadas: A função “regiao_conectada” verifica se um pixel faz parte de uma região conectada de pixels com a mesma cor.

5. Função principal: A função “contar_estrelas_e_meteoros” chama as funções anteriores para carregar a imagem, contar as estrelas e meteoros, e retornar os resultados.

```
from PIL import Image

def carregar_imagem(caminho_da_imagem):
    imagem = Image.open(caminho_da_imagem)
    pixels = imagem.load()
    largura, altura = imagem.size
    return pixels, largura, altura

def contar_estrelas(pixels, largura, altura):
    contagem_estrelas = 0
    for y in range(altura):
        for x in range(largura):
            pixel = pixels[x, y]
            if len(pixel) == 4:
                r, g, b, a = pixel
            else:
                r, g, b = pixel
            if (r, g, b) == (255, 255, 255):
                if regiao_conectada(pixels, x, y, largura, altura, (255,
255, 255)):
                    contagem_estrelas += 1
    return contagem_estrelas

def contar_meteoros(pixels, largura, altura):
    contagem_meteoros = 0
    meteoros_na_agua = 0
```

```
    for y in range(altura):
        for x in range(largura):
            pixel = pixels[x, y]
            if len(pixel) == 4:
                r, g, b, a = pixel
            else:
                r, g, b = pixel
            if (r, g, b) == (255, 0, 0):
                if regiao_conectada(pixels, x, y, largura, altura, (255,
0, 0)):
                    contagem_meteoros += 1
                    if meteoro_na_agua(pixels, x, y, largura, altura):
                        meteoros_na_agua += 1
    return contagem_meteoros, meteoros_na_agua

def meteoro_na_agua(pixels, x, y, largura, altura):
    for check_y in range(y + 1, altura):
        pixel_abaixo = pixels[x, check_y]
        if len(pixel_abaixo) == 4:
            r_abaixo, g_abaixo, b_abaixo, a_abaixo = pixel_abaixo
        else:
            r_abaixo, g_abaixo, b_abaixo = pixel_abaixo
        if (r_abaixo, g_abaixo, b_abaixo) == (0, 0, 0):
            break
        if (r_abaixo, g_abaixo, b_abaixo) == (0, 0, 255):
            return True
    return False

def regiao_conectada(pixels, x, y, largura, altura, cor):
    fila = [(x, y)]
    while fila:
        x_atual, y_atual = fila.pop(0)
        for dx, dy in [(0, 1), (0, -1), (1, 0), (-1, 0)]:
            novo_x, novo_y = x_atual + dx, y_atual + dy
            if 0 <= novo_x < largura and 0 <= novo_y < altura:
                novo_pixel = pixels[novo_x, novo_y]
                if len(novo_pixel) == 4:
                    r, g, b, a = novo_pixel
                else:
                    r, g, b = novo_pixel
                if (r, g, b) == cor:
                    fila.append((novo_x, novo_y))
                    pixels[novo_x, novo_y] = (0, 0, 0)
```

```
    return True

def contar_estrelas_e_meteoros(caminho_da_imagem):
    pixels, largura, altura = carregar_imagem(caminho_da_imagem)
    contagem_estrelas = contar_estrelas(pixels, largura, altura)
    contagem_meteoros, meteoros_na_agua = contar_meteoros(pixels,
largura, altura)
    return contagem_estrelas, contagem_meteoros, meteoros_na_agua

caminho_da_imagem = 'meteor_challenge_01.png'

contagem_estrelas, contagem_meteoros, meteoros_na_agua =
contar_estrelas_e_meteoros(caminho_da_imagem)

print("Número de Estrelas:", contagem_estrelas)
print("Número de Meteoros:", contagem_meteoros)
print("Meteoros caindo na Água:", meteoros_na_agua)
```

9. RESULTADOS

Número de Estrelas	314
Número de Meteoros	322
Meteoros caindo na Água	105

10. REFERÊNCIAS

Felgueiras, C. A. (s.d.). Processamento Digital de Imagens: Fundamentos Teóricos Iniciais. Recuperado de https://www.dpi.inpe.br/~carlos/Academicos/Cursos/Pdi/pdi_teorias.html

Martins, A. C. G. (s.d.). Introdução à Análise de Imagens Digitais [Documento PDF]. Computação Instrumental – Engenharia Ambiental – UNESP/Sorocaba. Recuperado de https://www.sorocaba.unesp.br/Home/Graduacao/EngenhariaAmbiental/antonio/imagens.pdf

Nedevschi, S., Marița, T., Dănescu, R., Oniga, F., Brehar, R., Giosan, I., Vancea, C., & Varga, R. (2023). Image Processing Laboratory Works (2nd ed.). UTPRESS.

USP. Processamento de Imagens [Slides]. Instituto de Ciências Matemáticas e de Computação,SCC0251. Recuperado de https://edisciplinas.usp.br/pluginfile.php/657744/mod_resource/content/0/Intro2016.pdf