Marcel Alessandro Zimmer

Grr 20221086

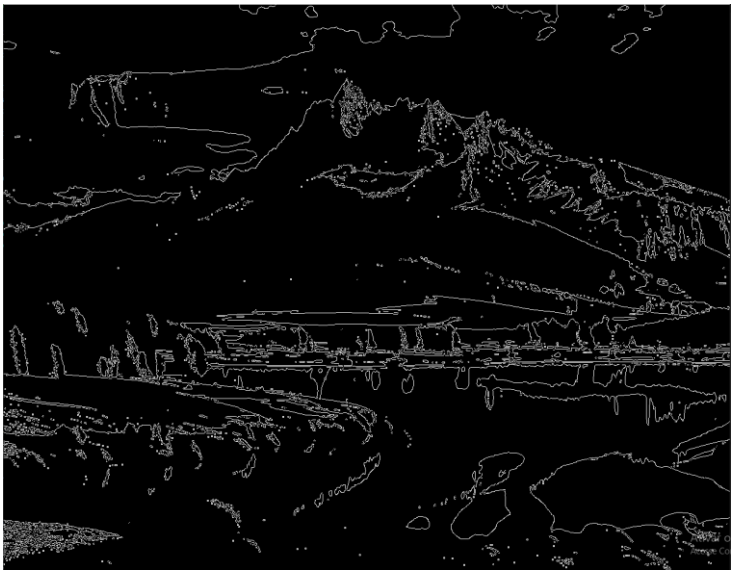
Atividade 7

Algoritmo de detecção de borda:

Foi usado a função cv2.Canny do open cv após converter a imagem em cinza e binárias

Imagem original:

Imagem depois do algoritimo :



Codigo:

atividade _7 > atv7.py > ...

```python
import cv2

imagem_colorida = cv2.imread(r"atividade _7\img\img.jpg")


largura = int(imagem_colorida.shape[1] * 0.3)
altura = int(imagem_colorida.shape[0] * 0.3)
imagem_redimensionada = cv2.resize(imagem_colorida, (largura, altura))

imagem_cinza = cv2.cvtColor(imagem_redimensionada, cv2.COLOR_BGR2GRAY)

_,imagem_binaria = cv2.threshold(imagem_cinza, 127, 255, cv2.THRESH_BINARY)

detBorda = cv2.Canny(imagem_binaria, 100, 200)



cv2.imshow("imagem",detBorda)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Análise do algoritmo:

O método Canny passa por várias etapas para garantir que as bordas detectadas sejam nítidas e bem definidas, entre elas estão:

- **Suavização**: O algoritmo aplica um filtro Gaussiano à imagem para reduzir o ruído, suavizando a imagem e prevenindo a detecção de bordas ruidosas.
- **Cálculo do Gradiente**: Calcula o gradiente de intensidade da imagem usando operadores como Sobel. Isso determina a magnitude e a direção das mudanças na intensidade dos pixels.
- **Supressão de Máximos**: Refinamento das bordas detectadas, mantendo apenas os pixels que são máximos locais na direção do gradiente. Isso ajuda a tornar as bordas mais finas.
- **Aplicação de Duplo Threshold**: Usa dois valores de limiar (superior e inferior) para classificar os pixels em:
    - **Bordas fortes**: Pixels com intensidade acima do limiar superior.
    - **Bordas fracas**: Pixels com intensidade entre os dois limiares.
    - Pixels abaixo do limiar inferior são descartados.
- **Conexão de Bordas**: Os pixels fracos são mantidos apenas se estiverem conectados a pixels fortes, resultando em uma imagem de bordas onde as bordas mais relevantes são preservadas.

Dessa forma após a aplicação do algoritmo podemos observar uma grande diferença nas imagens, onde na imagem binária se tem o contorno feito pelos bits 1 do branco gerando a imagem, após o processamento conseguimos verificar que a imagem agora possui um contorno onde os bits 1 geram o desenho de uma forma, transformando a imagem em um grande contorno da original.

Algoritmo de filtragem:

Foi usado a função cv2. GaussianBlur do open cv após converter a imagem em cinza e binárias como opção de filtragem que gera um blur na imagem

Imagem binaria:

Imagem depois do algoritmo :

Código:

atividade _7 > atv7_filtragem > ...

```python
import cv2

imagem_colorida = cv2.imread(r"atividade _7\img\img.jpg")


largura = int(imagem_colorida.shape[1] * 0.3)
altura = int(imagem_colorida.shape[0] * 0.3)
imagem_redimensionada = cv2.resize(imagem_colorida, (largura, altura))

imagem_cinza = cv2.cvtColor(imagem_redimensionada, cv2.COLOR_BGR2GRAY)

_,imagem_binaria = cv2.threshold(imagem_cinza, 127, 255, cv2.THRESH_BINARY)

filtragem = cv2.GaussianBlur(imagem_binaria, (5, 5), 0)



cv2.imshow("imagem",filtragem)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Análise do algoritmo de filtragem:

A função cv2.GaussianBlur() do OpenCV é usada para aplicar um filtro Gaussiano a uma imagem, suavizando-a e reduzindo o ruído, ele separa o processo em 3 etapas principais:

- **Filtro Gaussiano**: O filtro é baseado na função de densidade da distribuição normal (Gaussiana). Ele atribui pesos a pixels em uma vizinhança com base na distância em relação ao pixel central, com pixels mais próximos recebendo maior peso.
- **Parâmetros**:
- **Imagem**: A imagem de entrada que você deseja suavizar.
- **Tamanho do kernel**: Um tupla que define o tamanho da janela do filtro (exemplo: (5, 5)). O kernel deve ser de tamanhos ímpares para garantir que haja um pixel central.
- **Desvio padrão (sigma)**: Um valor que determina a quantidade de suavização. Se sigma for 0, o OpenCV calcula automaticamente o valor com base no tamanho do kernel.
- **Operação**: Para cada pixel da imagem, a função aplica o filtro Gaussiano, calculando uma média ponderada dos pixels vizinhos, onde a média é influenciada pelo desvio padrão. Isso resulta em um novo pixel suavizado que reduz os detalhes finos e o ruído

Dessa forma podemos verificar que após o processamento da imagem a mesma se encontra com menos ruídos, deixando mais claro o que é a imagem e o que faz parte do fundo