UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

Processamento Digital de Imagens

Trabalho 1

LIMIARIZAÇÃO E ALTERAÇÃO LOCAL DE BRILHO

Relatório

Renan Augusto Leonel ra: 115138
Pedro Henrique de Melo Costa ra: 112653

1. Introdução

O processo de tresholding consiste em escolhermos um valor de brilho na imagem, onde a partir deste valor realizaremos uma alteração em todos os pixels de valores maiores ou menores do que ele. Neste trabalho, para decidirmos como será realizada esta operação, também será utilizada uma variável *acima*, onde *acima = True* implica em alterarmos o brilho para todos os pixels com valores **maiores** do que o limiar, e *acima = False* implica em alterarmos para todos os pixels com valores **menores** do que o limiar. Em caso de *acima = True* onde o valor de brilho exceda 255, deve ser atribuído apenas o valor 255. Analogamente para *acima = False,* para valores que sejam menores do que 0, será atribuído apenas 0. Ao realizarmos estas operações, garantimos que a imagem terá tons de cinza entre branco e preto.

Para o desenvolvimento, foi utilizado a linguagem Python, com o auxílio das bibliotecas OpenCV e numpy.

2. Desenvolvimento

O desenvolvimento consiste em converter todos os pixels da imagem para tipo inteiro, para que seja possível a manipulação, onde a imagem é do tipo numpy.ndarray para que seja possível utilizar fatiamento e indexação proporcionados pela biblioteca A partir disso, é criada uma matriz composta de valores booleanos, correspondentes aos pixels que devem ser alterados, matriz esta que dependerá do valor booleano “acima” coletado por input. Portanto, será inicializada uma matriz com valores True para os pixels que devem ter o brilho alterado.
Após isso, é feita uma verificação para que o valor de cada pixel acrescido do brilho não ultrapasse 255, ou seja inferior a 0. Para isso, foram utilizadas as funções da biblioteca numpy “maximum” e “minimum”, onde cada pixel que ultrapasse 255 terá seu valor fixo em 255, e cada pixel que seja inferior a 0 terá seu valor fixado em 0.
Por fim, é feita uma conversão de volta para o tipo uint8 da imagem original, juntamente com a chamada das funções da biblioteca OpenCV para imprimir a imagem em tela.

Para a execução do código, foram utilizadas duas imagens em tons de cinza, onde a partir delas, foram geradas 4 imagens adicionais para cada, utilizando em todas um valor de limiar = 127, juntamente com:

a) valor booleano acima = True e brilho = 127
b) valor booleano acima = True e brilho = -127
c) valor booleano acima = False e brilho = 127
d) valor booleano acima = False e brilho = -127

3. Resultados obtidos

Para cada item descrito na seção 2, foi gerada uma imagem de saída, a partir da imagem original abaixo:

imagem 1

Os resultados estão apresentados na ordem descrita anteriormente na seção 2.

2.a)

1. positive_true

2.b)

1. negative_true

2.c)

1. positive_false

2.d)

1. negative_false

imagem 2

2.a)

2. positive_true

2.b)

2. negative_true

2.c)

2. positive_false

2.d)

2.negative_false

4. Implementação

```
import numpy as np
import cv2 as cv

def limiarizacao(img, t, acima, brilho):
    if acima:
        x = img >= t
    else:
        x = img < t

    img = img.astype(int)

    if brilho > 0:
      img[x] = np.minimum(img[x] + brilho, 255)
    else:
```

```
      img[x] = np.maximum(img[x] + brilho, 0)

    img = img.astype(np.uint8)
    cv.imshow('image',img)
    cv.waitKey(0)

def main():
    img  = cv.imread('image2.png', 0)
    print(type(img))
    t = int(input("T = "))
    acima = bool(input("Acima (True | False) = "))
    brilho = int(input("Brilho = "))
    limiarizacao(img, t, acima, brilho)

if __name__ ==  '__main__':
    main()
```

5. Bibliografia

https://docs.opencv.org/4.x/index.html
https://numpy.org/doc/stable/