UNIVERSIDADE ESTADUAL DE MARINGÁ

TRABALHO DE PROCESSAMENTO DIGITAL DE IMAGENS

MATHEUS AUGUSTO SCHIAVON PARISE

RA: 107115

MARINGÁ, 2022

1. INTRODUÇÃO

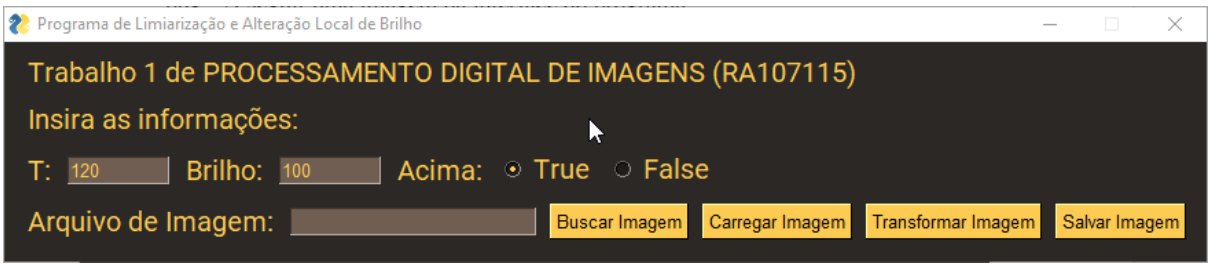
O intuito desse trabalho é desenvolver um programa em python 3.8 que seja capaz de receber uma imagem ‘a’, um inteiro ‘t’, um inteiro ‘brilho’ e um valor de verdadeiro ou falso para variável ‘acima’ e a partir disso somar o ‘brilho’ na imagem ‘a’ para os pixels de tons de cinza maiores ou iguais a ‘t’ caso ‘acima’ seja verdadeiro ou para os valores menores que ‘t’ caso ‘acima’ seja falso, os valores não podem exceder a faixa de valor branco que neste trabalho tem valor 255 e nem ser menores que preto que neste trabalho tem valor 0.

2. FUNCIONAMENTO DO PROGRAMA

O programa criado para esse trabalho é uma pequena interface que possui as entradas para informações solicitadas. Primeiramente, as entradas T e Brilho, **elas devem receber números inteiros**, caso um número float ou alguma entrada inválida seja digitado o programa irá gerar um erro. Em segundo lugar, Acima pode ter somente valor True ou False de acordo com a alternativa escolhida. Se a altura máxima da imagem original exceder 512px o programa irá alterar o tamanho da imagem exibida para ter no máximo 512px para facilitar a visualização, o programa **NÃO ALTERA** a imagem original apenas faz com que ela seja exibida de forma reduzida, ele ainda irá fazer as transformações e salvar a imagem com tamanho original, o mesmo vale para a largura mas esta foi limitada à 864 px .

Em seguida a parte dos arquivos da imagem, não há necessidade de digitar o caminho manualmente basta clicar no botão Buscar e selecionar a imagem (ela não precisa estar em tons de cinza o programa faz isso para você), se ela não aparecer basta mudar a extensão dos arquivos na busca de .jpg e jpeg para *(todos). Após selecionar a imagem clique no botão Carregar Imagem, ele irá fazer exatamente isso, após isso o botão Transformar imagem serve para aplicar as transformações considerando as variáveis T, Brilho e Acima na imagem carregada (ou seja, **é obrigatório carregar a imagem, é válido ressaltar que o arquivo precisa existir, estar com o nome correto e ser de fato uma imagem caso contrário o programa irá levantar erro**). Por fim o botão de salvar, ao ter uma imagem carregada e transformada se for de interesse, é possível salvar a imagem ao clicar no botão salvar, o programa irá salvá-la automaticamente na mesma pasta da imagem original com o formato .png e o nome tendo a seguinte formatação: Nome original da imagem +

"_Brilho" +Valor do Brilho + "_T" + Valor de T + "_Acima" + Valor de Acima + ".png". A seguir uma imagem da interface do programa.



3. CÓDIGO

A seguir estão os prints relacionados ao código do programa descrito acima.

```python
import numpy as np
import PySimpleGUI as sg
import os, io
from PIL import Image

file_types = [("PNG (*.png)", "*.png"),
              ("JPEG (*.jpg)", "*.jpg"),
              ("All files (*.*)", "*.*")]

def tela():
    sg.ChangeLookAndFeel("DarkAmber")
    # Layout
    layout = [  [sg.Text("Trabalho 1 de PROCESSAMENTO DIGITAL DE IMAGENS (RA107115)", font="Roboto")],
                [sg.Text("Insira as informações:", font="Roboto")],
                [
                    sg.Text("T:", font="Roboto"), sg.Input(default_text="120", size=(10, 0), key="txtT"),
                    sg.Text("Brilho:", font="Roboto"), sg.Input(default_text="100", size=(10, 0), key="txtBrilho"),
                    sg.Text("Acima:", font="Roboto"),
                    sg.Radio("True", "group1", key="radioBtnTrue", font="Roboto", default=True),
                    sg.Radio("False", "group1", key="radioBtnFalse", font="Roboto")
                ],
                [
                    sg.Text("Arquivo de Imagem:",  font="Roboto"),
                    sg.Input(size=(25, 1), key="-FILE-"),
                    sg.FileBrowse(button_text="Buscar Imagem",file_types=file_types),
                    sg.Button("Carregar Imagem"),
                    sg.Button("Transformar Imagem"),
                    sg.Button("Salvar Imagem")
                ],
                [sg.Image(key="-IMAGE-")]
              ]
    # Janela
    janela = sg.Window('Programa de Limiarização e Alteração Local de Brilho', layout, icon=r'./ic.ico')
    imagem = None
```

```python
while True:
    # Extrair os dados da tela
    event, values = janela.Read()

    # Função para sair do programa
    if event == "Exit" or event == sg.WIN_CLOSED:
        break

    if event == "Carregar Imagem":
        filename = values["-FILE-"]
        if os.path.exists(filename):
            imagem = Image.open(values["-FILE-"]) # Objeto PIL Image
            width, height = imagem.size

            # Limitando altura maxima
            width = (864 if width > 864 else width)
            height = (512 if height > 512 else height)
            imagem2 = imagem.resize([width,height])

            # Salvando a imagem na memoria em binario
            bio = io.BytesIO()
            imagem2.save(bio, format="PNG")

            # Mostrando a imagem na janela
            janela["-IMAGE-"].update(data=bio.getvalue())
```

```python
    if event == "Transformar Imagem":
        if imagem is not None:
            matriz = np.array(imagem.convert('L')) # Transforma a imagem para tons de cinza
            matriz = matriz.astype(int) # int32

            Acima = values["radioBtnTrue"]
            t = int(values["txtT"])
            brilho = int(values["txtBrilho"])
            if Acima:
                # Soma o brilho no intervalo dos valores maiores ou iguais a t
                matriz[matriz >= t] += brilho
            else:
                # Soma o brilho no intervalo dos valores menores que t
                matriz[matriz < t] += brilho

            matriz = np.minimum(np.maximum(matriz, 0), 255) # Limita o valor maximo para 255 e o minimo para 0
            matriz = matriz.astype(np.uint8) # Volta a matriz para o tipo uint8 do numpy
            pil_image= Image.fromarray(matriz) # Transforma de volta pra imagem PIL

            # Limitando altura maxima
            width, height = imagem.size
            width = (864 if width > 864 else width)
            height = (512 if height > 512 else height)
            pil_image2 = pil_image.resize([width,height])

            # Salvando a imagem na memoria em binario
            bio = io.BytesIO()
            pil_image2.save(bio, format="PNG")

            # Mostrando a imagem na janela
            janela["-IMAGE-"].update(data=bio.getvalue())
```

```
        if event == "Salvar Imagem":
            if imagem is not None:
                # Salvando a imagem no local da pasta da imagem original
                tam_formato = len(imagem.format)
                tam_nome = len(imagem.filename)
                tam_f = tam_nome - tam_formato
                nome_final = imagem.filename[:tam_f] + "_Brilho" + str(brilho) + "_T" + str(t) + "_Acima" + str(Acima) + ".png"
                pil_image.save(nome_final)

    janela.close()

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

4. IMAGENS

Foram selecionadas três imagens diferentes de preto, branco e transparente em consideração a sanidade mental do professor. As seguintes variações de configurações para cada uma delas foram definidas:

- T = 120 para primeira, T = 60 para segunda, T = 200 para terceira
- Acima valendo True ou False
- Brilho Valendo 100 ou -100

Isso gera 4 configurações para 3 imagens dando 12 imagens, como irei mostrar as versões originais isso totalizará 15 imagens.

Imagem 1 - Coelho Com Cenoura Original

Imagem 2 - Coelho Com Cenoura T = 120, Acima True, Brilho 100

Imagem 3 - Coelho Com Cenoura T = 120, Acima False, Brilho 100

Imagem 4 - Coelho Com Cenoura T = 120, Acima True, Brilho -100

Imagem 5 - Coelho Com Cenoura T = 120, Acima False, Brilho -100

Imagem 6 - Gato Francês Original

Imagem 7 - Gato Francês T = 60, Acima True, Brilho 100

Imagem 8 - Gato Francês T = 60, Acima True, Brilho -100

Imagem 9 - Gato Francês T = 60, Acima False, Brilho 100

Imagem 10 - Gato Francês T = 60, Acima False, Brilho -100

Imagem 11 - Espaço Sideral James Webb Original

Imagem 12 - Espaço Sideral James Webb T = 200, Acima True, Brilho 100

Imagem 13 - Espaço Sideral James Webb T = 200, Acima True, Brilho -100

Imagem 14 - Espaço Sideral James Webb T = 200, Acima False, Brilho 100

Imagem 15 - Espaço Sideral James Webb T = 200, Acima False, Brilho -100

5. BIBLIOGRAFIA

Para o desenvolvimento deste trabalho foi necessário Python 3.8, com as bibliotecas *NUMPY* para manipular as matrizes das imagens, *PySimpleGUI* para criar a interface de usuário, *PILLOW* para abrir, manipular e salvar as imagens, *OS* e *IO* para interagir com os arquivos e seus formatos e por fim *PYINSTALLER* para gerar o arquivo executável. As documentações do python 3.8 e dessas bibliotecas também foram utilizadas.