Seleção de Pixels Semelhantes

Rosana Ribeiro
rosanasrib@gmail.com

Departamento de Ciência da Comptutação
Universidade de Brasília
Campus Darcy Ribeiro, Asa Norte
Brasília-DF, CEP 70910-900, Brazil

Abstract

Este documento relata o processo de implementação de um algoritmo em *python* para identificação e marcação de *pixels* semelhantes a partir de arquivos de imagem, de vídeo e da leitura da câmera, utilizando as bibliotecas *OpenCV* e *NumPy*.

1 Introdução

Uma imagem digital é uma matriz de elementos denominados *pixels*. O mapeamento dos elementos se refere diretamente à posição de um ponto de cor específica na imagem.

A representação mais conhecida popularmente de um *pixel* é em 3 valores de 8 bits (de 0 a 255), no espaço de cores RGB. Essa é a representação utilizada neste projeto, que então pode definir a imagem como uma matriz de 3 eixos, que chamaremos de matriz**[linha, coluna, camada]**, onde a linha e a coluna definem o tamanho da imagem e a camada é fixa em 3, para os valores RGB de cada elemento.

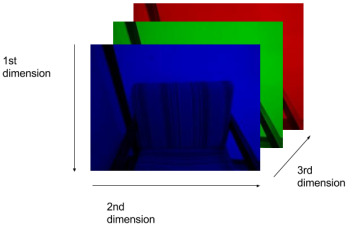

Figure 1: Representação da matriz com as camadas B, G e R

Serão exploradas duas formas de leitura de uma imagem:

- Imagem Colorida
- Imagem Preto e Branca ou em Tons de Cinza

Neste projeto, definiremos que *pixels* semelhantes são aqueles que possuírem uma diferença menor que 13. No caso de uma imagem colorida, será feita a diferença euclidiana e no caso de imagens em tons de cinza, será a diferença crua entre a cor em referência e o *pixel* em questão.

A seleção de *pixels* semelhantes é uma ferramenta muito comum em aplicativos e programas de *photoshop* e este projeto visa a marcação desses elementos, de acordo com uma referência escolhida pelo usuário, ao clicar com o botão esquerdo do mouse em alguma parte da imagem.

2 Metodologia

A utilização das bibliotecas *OpenCV*[1] e *NumPy* foram essenciais para a realização dessa atividade e a implementação deste experimento se deu em quatro etapas. Sendo elas:

1. Abertura de arquivo de imagem e identificação do *pixel* selecionado, assim como suas componentes R, G e B, caso for no modo Colorido, ou sua componente *grayscale*, caso for no modo Cinza.

2. Identificação e marcação dos *pixels* semelhantes ao escolhido em uma nova imagem.

3. Adaptação para os *frames* de um vídeo, em modo colorido e em tons de cinza.

4. Adaptação para os *frames* capturados a partir da câmera do computador.

2.1 Abertura de arquivo e identificação do *pixel* selecionado

A biblioteca *OpenCV* possui funções que possibilitam a abertura de um arquivo e que recebem os chamados eventos do *mouse*, como o clique do botão esquerdo. Foi criada uma rotina para quando esse evento ocorrer, que basicamente imprime na tela as coordenadas e os valores RGB do ponto selecionado.

```
def mouse_click(event, x, y, flags, params):
    global img, init, real_x, real_y, b, g, r
    if event == cv2.EVENT_LBUTTONDOWN:
        init = True
        real_x = x
        real_y = y
        print('\n\nPixel selecionado em x,y = (', x,',', y,').')
        if opt == '1' or opt == '3' or opt == '5':
            b = img[y,x,0]
            g = img[y,x,1]
            r = img[y,x,2]
            print('Valores RGB:\nR = ', r,'G = ', g, 'B = ', b)
            matrix = np.square(np.subtract(img.astype('int32'), np.array([b, g, r])))
            matrix = np.sqrt(np.sum(matrix, axis = 2))
            img2 = np.copy(img)
            img2[matrix<13] = np.array([0, 0, 255])
            cv2.imshow('1',img2)

        elif opt == '2' or opt == '4' or opt == '6' :
            print('Valor em GRAYSCALE:\nG = ',img[y,x],'.\n')
            g = img[y,x]
            matrix = np.abs(np.subtract(img.astype('int32'), g))
            img2 = np.reshape(img,(img.shape[0], img.shape[1], 1))
            img2 = np.repeat(img2, 3, axis = 2)
            img2[matrix<13] = np.array([0, 0, 255])
            cv2.imshow('1',img2)
```

Figure 2: Função chamada quando ocorre evento do mouse

```
def image(img) :
    print('Aperte \'Esc\' para sair.\n')
    cv2.imshow('Matrix', img)
    cv2.setMouseCallback('Matrix', mouse_click)

    while cv2.waitKey(1) != 27:
        pass
    cv2.destroyAllWindows()
```

Figure 3: Função para abertura de imagem

2.2 Identificação e marcação de *pixels* semelhantes

A biblioteca *NumPy* analisa e manipula os elementos das matrizes, de acordo com o objetivo descrito em 1.

Nesta etapa, o foco é a identificação de *pixels* semelhantes. Essa semelhança é definida por uma diferença menor que 13.

No caso de imagens coloridas, o método utilizado foi a distância euclidiana, que consiste em tirar a diferença das componentes R, G e B da cor de referência com todos os elementos da matriz original, elevá-los ao quadrado, somar e tirar a raiz.

Distância entre duas instâncias $\mathbf{p_i}$ e $\mathbf{p_j}$ definida como:

$$d = \sqrt{\sum_{k=1}^{n} (p_{ik} - p_{jk})^2}$$

$\mathbf{p_{ik}}$ e $\mathbf{p_{jk}}$ para k = 1, ..., **n** são os **n** atributos que descrevem as instâncias $\mathbf{p_i}$ e $\mathbf{p_j}$, respectivamente

Figure 4: Função para distância euclidiana

No caso de imagens em tons de cinza, a diferença foi crua. As funções de implementação são as mesmas das figuras 2 e 3 na seção 2.1.

2.3 Leitura para *frames* de um vídeo

Nessa etapa, os cuidados foram apenas em mudar a entrada da imagem para o vídeo escolhido e adicionar as funções referentes à vídeo da *OpenCV*.

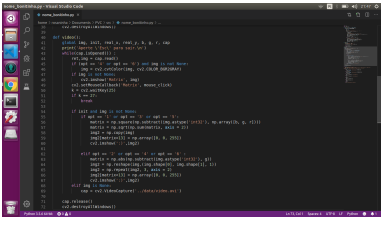

Figure 5: Função para abertura de imagem

2.4 Leitura para *frames* da câmera

Nesta etapa, a única diferença em relação à 2.3, foi a origem da entrada, selecionada como 0.

```
94      elif opt == '5':
95          cap = cv2.VideoCapture(0)
96          video()
97      elif opt == '6':
98          cap = cv2.VideoCapture(0)
99          video()
```

Figure 6: Função para abertura de imagem pela câmera

3 Resultados

Ao executar o programa, o usuário visualiza um menu com as opções desejadas. Os resultados obtidos estão a seguir.

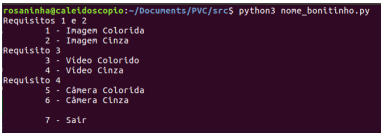

Figure 7: Menu que aparece no terminal ao executar o programa

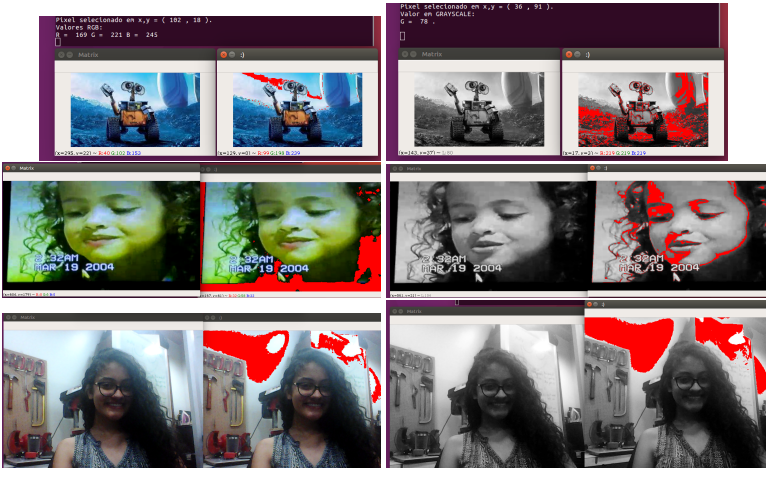

Figure 8: Resultado de cada opção

4 Discussão e Conclusões

Em se tratando das bibliotecas utilizadas, nota-se que a manipulação de matrizes se torna muito mais eficiente com as funções da *NumPy*. Operações que seriam realizadas em *loops* foram realizadas de forma mais ágil com funções como *np.subtract(...)*. A *OpenCV* possui funções específicas para manipulação de imagens, vídeos e câmera, além das funções que capturam eventos, que são essenciais para o funcionamento e a implementação do código.

Nota-se a necessidade de declarar variáveis globais nas definições das funções

A forma que o programa escolhe os arquivos a serem lidos está predeterminada no código, então o usuário não tem a opção de escolher uma outra imagem ou vídeo, a não ser alterando o código.

O experimento teve resultados satisfatórios e cumpriu com o objetivo.

References

[1] V.; BOUGUET J. BRADSKY, G. R; PISAREVSKY. *Learning OpenCV: Computer Vision with OpenCV Library*. Springer, 2006.