Funções básicas do OpenCV

Mateus de Moura Ramos Bittencourt
bittenmat@gmail.com

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

Abstract

This document demonstrates some of the simplest functions to image processing and computer vision on OpenCV. It contains examples of its usage and the result as well. The main library used on these programs was OpenCV, followed by numpy. Every non-intuitive function or logic implementation will be explained here.

1 Introdução

Visão computacional e processamento de imagens são duas áreas muito próximas, ambas contendo várias características atrelaçadas. Neste trabalho, foi utilizado técnicas de processamento de imagem de forma a integrar a imagem ou o video com o mouse. Para isso, utilizou-se duas bibliotecas principais, OpenCV e Numpy.

1.1 OpenCV

OpenCV, ou Open Computer Vision, é uma biblioteca de visão computacional fundada pela Intel em 2000, que passou por várias empresas, até retornar a intel e tornar-se open source, que permite a qualquer um editar. Atualmente, na versão 4.1, essa biblioteca pode ser usada para várias linguagens e sistemas operacionais. Pode-se dizer que grande parte do avanço na área de visão computacional deve-se à popularização dessa biblioteca.

[1]

1.2 Numpy

O trabalho aqui feito, foi realizado em python e uma de suas bibliotecas mais conhecidas, a numpy. Numpy é uma biblioteca de uso científico, que contém funções pre implementadas para facilitar o uso de matriz, vetor, e algebra como um todo. Criada por Travis Oliphant em 2005, foi amplamente usada ao longo dos anos e atualmente também é uma biblioteca aberta a todos.

2 Problema Proposto

2.1 Integração mouse-imagem

O primeiro problema proposto consiste em abrir uma imagem e ao clicar em algum ponto da imagem, retornar os valores de RGB se esta for colorida, ou intesidade de cinza se esta for em grayscale, assim como as coordenadas do ponto.

[5] [3]

2.2 Destacar uma relação entre pixels em uma imagem

Nesta seção, após selecionar um pixel na imagem, pixels com uma distancia euclidiana de cor menor que 13 deviam ser destacados, caso a imagem seja colorida, e uma diferença de intensidade menor que 13 no caso grayscale.

[4]

2.3 Destacar uma relação entre pixels em um vídeo

O desafio dessa parte se mantém, porém agora com um vídeo. Isso é, ao clicar o valor do pixel é armazenado e comparado à todos os pixels do frame. Ao mudar de frame, o pixel clicado se mantem e compara-se aos novos pixels da imagem.

2.4 Destacar uma relação entre pixels na webcam

O problema final é repetir o problema de achar a relação entre pixels, porém agora em um vídeo gerado na webcam em tempo real.

3 Metodologia

Para destacar uma área considerada "próxima" ao pixel clicado em uma imagem, uma nova imagem era criada em uma nova janela, contendo os mesmo pixels da imagem original, porém com os pixels analisados como "próximos" pintados de uma cor só.

Para obter uma relação de pixels como foi feito no trabalho, foi necessário fazer a distância euclidiana para imagens coloridas. Para isso, clicou-se num pixel da imagem e se obteve os valores de azul (B_1), verde (G_1) e vermelho (R_2). A comparação com outro pixel qualquer (B_2, G_2, R_2) se da pela fórmula:

$$Dist = \sqrt{((B_1 - B_2)^2 + (G_1 - G_2)^2 + (R_1 - R_2)^2)} \quad (1)$$

Caso a imagem esteja em grayscale, basta subtrair o valor do pixel clicado com o pixel analisado. Caso seja menor que 13 ele deveria receber o destaque.

Vale ressaltar que o processamento em python de uma matriz, passando pixel a pixel resultaria em um atraso muito grande, pois seria um loop dentro de um loop, ordem (n^2), portanto para reduzir esse tempo de processamento uma lógica nova foi implementada. A partir do pixel clicado uma imagem nova é gerada contendo somente esse pixel, e te tamanho igual ao tamanho da imagem original. A partir disso, subtrai-se essa imagem da original e

guarda o resultado numa imagem de distâncias. A partir dela, operações lógicas são aplicadas de forma a pintar de vermelho todos os pixels que possuem distância menor que 13. Dessa forma, não é necessário nenhum loop para analisar todos os pixels.

[6] [7] [2]

3.1 Vídeos e webcam

Como dito anteriormente, para analisar um vídeo basta analisar várias imagens em sequência. Para isso criou-se um loop de visualização de frames com a condição de parada de que o próximo frame seja igual ao número total de frames do vídeo. Caso isso ocorra o vídeo acaba e a janela se encerra. A análise da webcam se dá da mesma forma, analisando frame a frame, porém não há contagem de frames, portanto para se encerrar o usuário deve clicar a tecla esc ou cancelar o código através do terminal.

4 Resultados

Aqui estão alguns dos resultados obtidos com os problemas propostos:

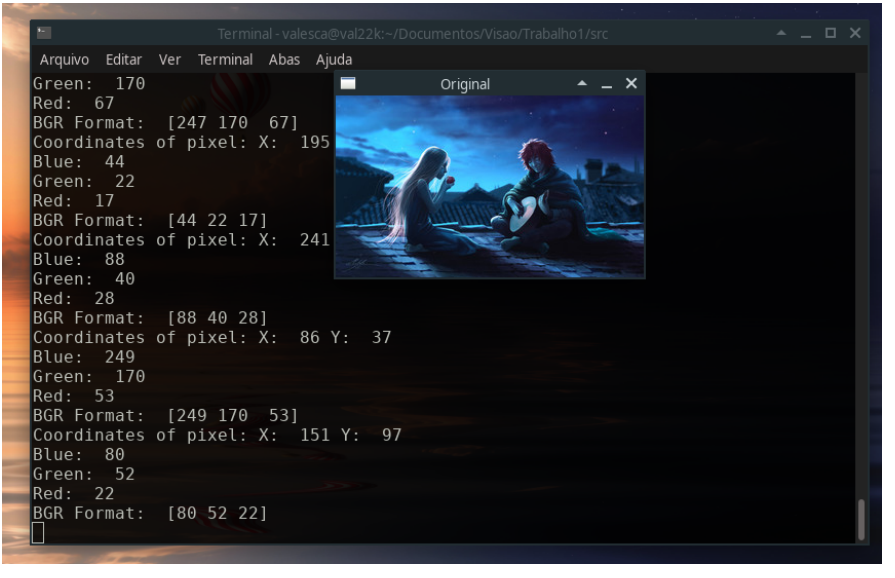

Figure 1: Resultado do primeiro problema proposto

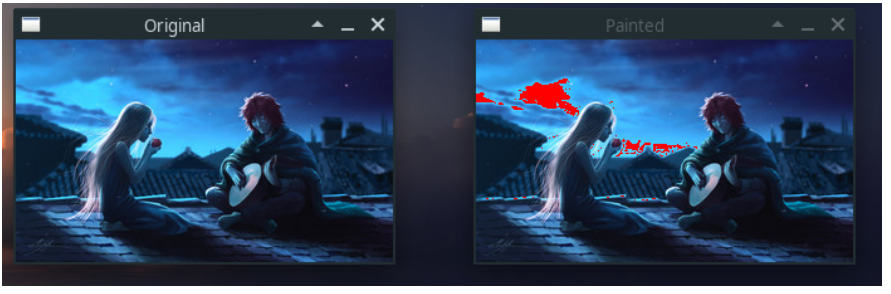

Figure 2: Resultado do segundo problema proposto

Figure 3: Resultado do terceiro problema proposto

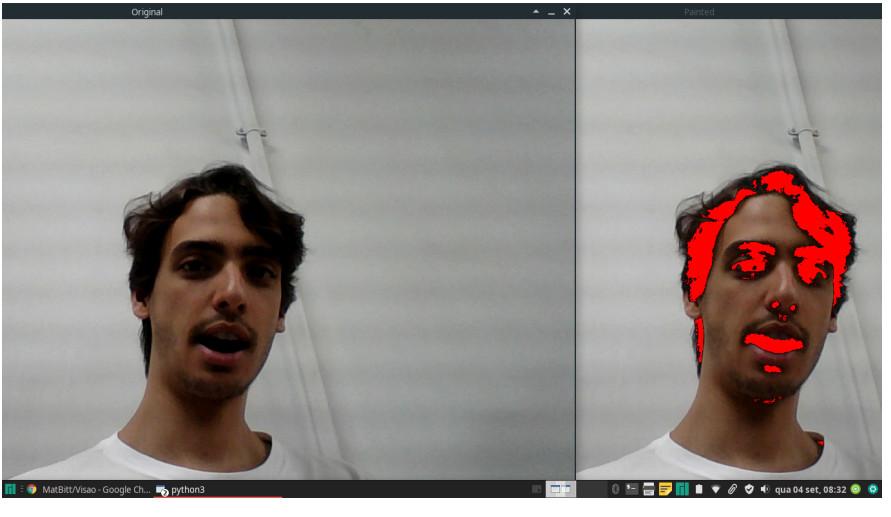

Figure 4: Resultado do quarto problema proposto

5 Discussão e Conclusão

Os resultados obtidos aqui foram satisfatórios e consistentes. Percebe-se nas imagens presentes na aba de resultados que os problemas propostos foram solucionados.

Python possui uma vasta gama de funções pré implementadas que facilitam o manuseio de matrizes e vetores, com a biblioteca numpy, mas isso acarreta numa demora maior para o processamento das mesmas. Portanto loops para essa linguagem são altamente não recomendáveis. Nesse trabalho, esse problema foi contornado com sucesso, através de operações lógicas e transposição de matrizes.

Por fim, não houve nenhum atraso aparente na webcam, apresentando resultados muito bons também.

References

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[2] *The OpenCV Reference Manual*. OpenCV Community, 3.4.1 edition, August 2019. URL https://docs.opencv.org/3.4.3/d6/d6d/tutorial_mat_the_basic_image_container.html.

[3] *The OpenCV Reference Manual*. OpenCV Community, 3.4.1 edition, August 2019. URL https://docs.opencv.org/3.0-beta/modules/highgui/doc/user_interface.html.

[4] *The OpenCV Reference Manual*. OpenCV Community, 3.4.1 edition, August 2019. URL https://docs.opencv.org/3.4.2/df/d9d/tutorial_py_colorspaces.html.

[5] *The OpenCV Reference Manual*. OpenCV Community, 3.4.1 edition, August 2019. URL https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_core/py_basic_ops/py_basic_ops.html.

[6] *The OpenCV Reference Manual*. OpenCV Community, 3.4.1 edition, August 2019. URL https://docs.opencv.org/3.4/d7/d1b/group__imgproc__misc.html#gae8a4a146d1ca78c626a53577199e9c57.

[7] Brad Solomon-(https://realpython.com/team/bsolomon/). Look ma, no for-loops: Array programming with numpy. Real Python. URL https://realpython.com/numpy-array-programming/.