Projeto Demonstrativo 1

Pedro Henrique Luz de Araujo
pedrohluzaraujo@gmail.com

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

Abstract

O presente Projeto Demonstrativo consiste em um programa capaz de: informar posição e valor de um pixel escolhido tanto em imagens coloridas quanto em escala de cinza; e colorir todos os pixeis de uma imagem ou vídeo, capturado em tempo real ou não, cujo valor seja próximo ao de um pixel selecionado. O projeto divide-se em quatro requisitos: o primeiro trata da obtenção das coordenadas e valores de pixeis; o segundo, da colorização de pixeis semelhantes de uma imagem; o terceiro é análogo ao segundo, mas em vídeo; por fim, o quarto é equivalente ao terceiro aplicado a uma captura de vídeo em tempo real.

1 Introdução

O Projeto Demonstrativo 1 tem como objetivo estabelecer o primeiro contato com técnicas de processamento de imagem e vídeo que serão usadas em aplicações de Visão Computacional. Tais aplicações buscam reconstruir propriedades visuais do mundo (forma, iluminação, distribuição de cores) a partir de uma ou mais imagens [1]. Como ponto de partida, foram explorados o carregamento de imagens e vídeos e a representação computacional das cores.

As imagens podem ser representadas digitalmente como matrizes tridimensionais, em que duas das dimensões representam a posição dos pixeis, enquanto a outra representa a cor de cada pixel. Um vídeo por sua vez, pode ser considerado uma sequência de imagens, onde cada uma representa um *frame* do vídeo.

Já as cores são comumente representadas por três *bytes*, no caso de imagens coloridas, ou apenas um *byte* no caso de imagens em preto e branco. No primeiro caso, cada um dos *bytes* corresponde a intensidade de uma cor primária. No espaço de cores RGB, por exemplo, cada *byte* representa a intensidade de vermelho, verde ou azul. No segundo caso, 0 representa preto, 255, branco, e os valores intermediários são tons de cinza. Dessa forma, é possível estabelecer uma medida de semelhança entre cores, por exemplo, por meio da distância euclidiana entre elas no espaço de cores

O presente trabalho está organizado da seguinte maneira: a Seção 2 apresenta a metodologia utilizada para a construção do programa; a Seção 3 apresenta os resultados obtidos; e a Seção 4 conclui o trabalho.

2 Metodologia

2.1 Ferramentas

Usamos a biblioteca OpenCV [1] para o carregamento e exibição de imagens e vídeos. Para realizar operações sobre as matrizes de imagens, utilizamos a biblioteca NumPy. Usamos ainda, como linguagem, Python 3.5.2, e o gerenciador de bibliotecas Anaconda 3.

2.2 Implementação e Uso

Para o carregamento e exibição de imagens e vídeos, bastou a utilização de funções da biblioteca OpenCV. O usuário tem como opção usar imagens padrão guardadas na pasta *data* ou utilizar imagens próprias, conforme instruções presentes no arquivo *read-me.txt*.

O requisito 1 trata do carregamento e apresentação de uma imagem na tela. O usuário pode clicar na imagem e obter no terminal as coordenadas e o valor do pixel clicado. Em caso de imagens coloridas, tal valor é uma tripla de inteiros que representam a intensidade de vermelho, verde e azul; em imagens em tons de cinza, trata-se apenas de um inteiro. Ressaltamos que a biblioteca OpenCV representa as cores em BGR, o inverso de RGB, de modo que mostrou-se necessário inverter a tripla ao apresentá-la ao usuário.

O requisito 2 permite que o usuário clique em um pixel de modo que todos os outros pixeis com cor "semelhante" à dele são coloridos de vermelho. Foi estabelecida como medida de semelhança a distância euclidiana no espaço de cores. Definiu-se que pixeis semelhantes apresentam distância menor que 13. A distância euclidiana foi calculada utilizando a biblioteca de álgebra linear do NumPy, em razão de sua implementação otimizada.

O requisito 3 aplica a técnica do requisito 2 a vídeos. A diferença principal é a necessidade de colorir cada um dos *frames*. Por fim, o requisito 4 é análogo ao requisito 3, mas aplicado a vídeos obtidos em tempo real.

3 Resultados

3.1 Requisito 1

A figura 1 apresenta a imagem carregada e analisada e o retorno ao usuário após clicar, respectivamente, nos cantos superior esquerdo, superior direito, inferior esquerdo e inferior direito da imagem, e nas partes com cor vermelha, verde e azul. Tais experimentos indicam o funcionamento correto do procedimento

3.2 Requisito 2

As figuras 2 e 3 comparam a imagem original antes e depois de clicar nas partes pretas e azul. Ao clicar no preto, tudo que é preto tornou-se vermelho; ao clicar no azul, o azul tornou-se vermelho.

As figuras 4 e 5, e 6 e 7 exibem o funcionamento do requisito 2 em uma imagem em preto e branco e em uma imagem com intensa nuance de cores, respectivamente.

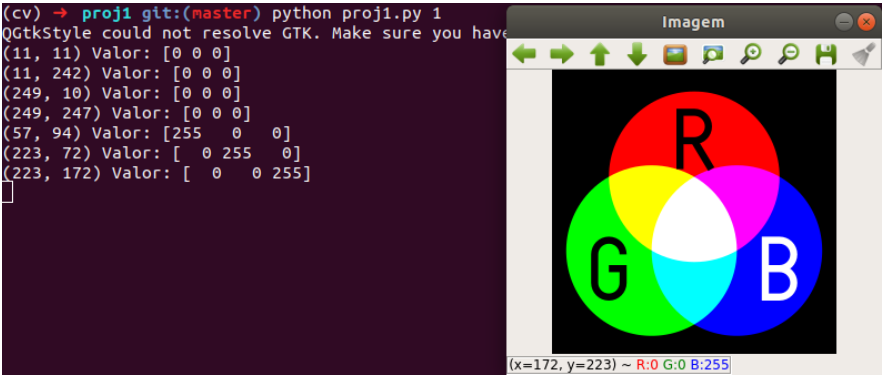

Figure 1: Imagem [3] analisada e exibição de coordenadas e valores dos pixeis clicados.

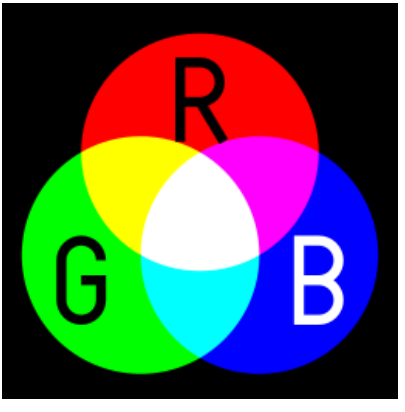

Figure 2: Figura original.

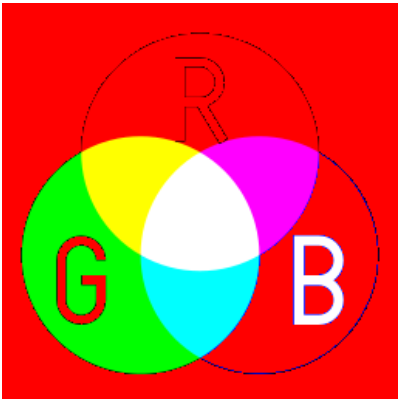

Figure 3: Figura após edições.

3.3 Requisito 3

A figura 8 exibe um *frame* do vídeo padrão do requisito 3 [1], que consiste em uma gota caindo em água, após um clique. Na execução do requisito, os *frames* são colorizados e atualizados em tempo real à medida em que o vídeo progride.

3.4 Requisito 4

A figura 9 consise em um *frame* de uma captura por *webcam* após o clique em um pixel do vídeo. Assim como no caso do requisito 3, os *frames* são colorizados em tempo real durante a captura do vídeo.

4 Discussão e Conclusões

O presente trabalho demonstrou o processo de carregamento e exibição de imagens e vídeos usando a biblioteca OpenCV, além da manipulação individual dos pixeis de uma imagem ou

[1] Retirado do site http://www.engr.colostate.edu/me/facil/dynamics/avis.htm, acesso em 26 de agosto de 2018

Figure 4: Figura original.

Figure 5: Após clicar nas penas.

Figure 6: Figura original [].

Figure 7: Após clicar no pescoço.

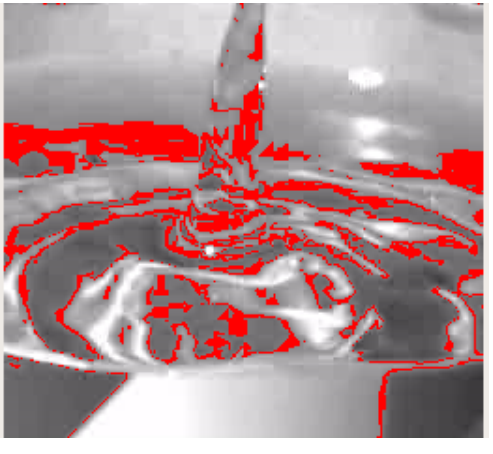

Figure 8: *Frame* após colorização.

frame por meio da biblioteca NumPy.

Primeiramente, ficou clara durante a implementação da aplicação a necessidade de vetorização do procedimento de colorização das imagens. Embora um *loop* por todos os pixeis seja viável na colorização das imagens, o mesmo não se aplica ao caso dos vídeos, que apresentam vários *frames* a serem processados por segundo. Daí a importância do uso das

Figure 9: *Frame* após colorização.

operações sobre matrizes implementadas pelo NumPy, que são otimizadas.

Além disso, ao se comparar a colorização da imagem 6 com as demais, nota-se que poucos pixeis foram colorizados. Tal fato se deve à grande variedade de cores da imagem. Para conseguir um bom resultado na colorização dela seria necessário aumentar a distância máxima para se considerar duas cores semelhantes.

O código-fonte do projeto está disponível em https://github.com/peluz/cv-foundations-1.

References

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

[2] Wikimedia Commons. File:peacock plumage.jpg — wikimedia commons, the free media repository, 2017. URL https://commons.wikimedia.org/w/index.php?title=File:Peacock_Plumage.jpg&oldid=240821319. [Online; versão de 19:08, 26 de junho 2016].

[3] Wikimedia Commons. File:additivecolor.svg — wikimedia commons, the free media repository, 2018. URL https://commons.wikimedia.org/w/index.php?title=File:AdditiveColor.svg&oldid=312348511. [Online; versão de 00:37, 25 de julho 2018].

[4] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010. ISBN 1848829345, 9781848829343.