Explorando OpenCV

Bruno Fernandes Carvalho - 15/0007159
Dep. Ciência da Computação - Universidade de Brasília (UnB)
Princípios de Visão Computacional - Turma A
Data de realização: 31/03/2017
brunofcarvalho1996@gmail.com

Abstract

The main idea of this project is explore and experience OpenCV environment to understand how to use this powerful computer vision tool. It was possible to make basic image/video manipulations, like interfacing it with the user, and to understand the architecture of the library, as the basic image container. After finished this project, it can be affirmed that it gave enough groundwork to apply this tool in various applications.

1. Objetivos

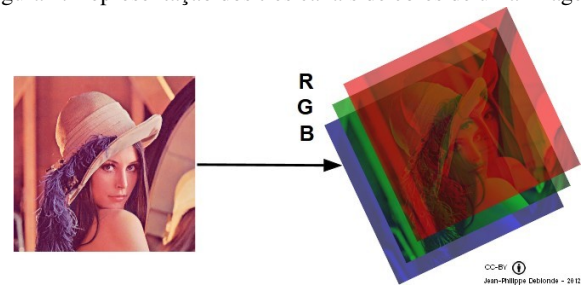
Essa atividade tem como base entender e experienciar a biblioteca OpenCV, que dá suporte e ferramentas suficientes para manipular, processar e mostrar ao usuário imagens e vídeos. Para isso, devem ser desenvolvidos algoritmos que realizam operações básicas em imagens e vídeos.

2. Introdução

Para manipular corretamente uma imagem, deve ser entendido como ela é armazenada e quais são seus atributos. Quando capturada e transformada em dispositivos digitais, a imagem passa a ser uma representação numérica de vários pontos específicos. Esses pontos são as menores entidades que formam uma imagem e são chamados de pixel. Tendo essa definição como base, é possível extrair diversos atributos, como sua altura e largura em quantidade de pixels. Isso caracteriza a formação de matrizes para a representação de imagens.

Além disso, outro conceito essencial é de canal da imagem. Se cada pixel fosse representado por apenas um valor numérico, seria muito difícil abranger todas as combinações possíveis de cores, sendo necessário utilizar números muito grandes para conter todas as opções. Para resolver esse problema, uma imagem colorida é composta de três canais, cada um representando a intensidade de uma das cores primárias: vermelho, azul e verde. Assim, na representação final da imagem, esses três canais são combinados e formam diversas cores existentes. Na figura 1, é mostrado um exemplo de imagem com três canais sobrepostos.

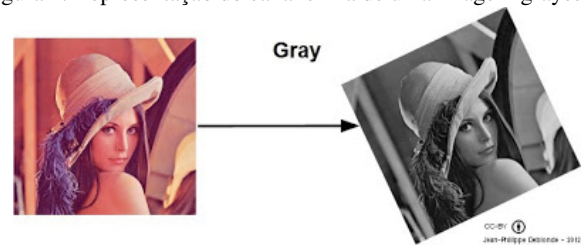
Figura 1. Representação dos três canais de cores de uma imagem



Fonte: Podeplace Blogspot[1]

Quando a imagem não é colorida, ela recebe o nome de ”grayscale”, que significa ”escala cinza”. Isso quer dizer que as intensidades de cada pixel da imagem representam apenas tonalidade de cinza, sendo necessário apenas um canal para sua representação. Um exemplo dessa imagem é mostrado na figura 2.

Figura 2. Representação do canal cinza de uma imagem grayscale



Fonte: Podeplace Blogspot [1]

Sabendo as principais características de uma imagem, pode-se entender o conceito de vídeo, que é um reprodução contínua de imagens que dão ideia de movimento. Assim, um vídeo é composto por ”frames”(imagens) que são repro-

[1]Disponível em ¡http://podeplace.blogspot.com.br/2012/12/pseudocolorisation-with-javascript.html¿ Acesso em abril.2017

duzidos à uma determinada taxa que pode variar.

3. Materiais e Metodologia

O desafio desse projeto é fazer uma aplicação que mostre uma imagem de entrada ao usuário e permita que ele clique com botão esquerdo sobre ela. Ao fazer isso, mostra-se no terminal os valores de intensidade do pixel clicado: se é uma imagem colorida, é apresentado os valores dos três canais RGB, e se a imagem for grayscale, mostra-se a intensidade do pixel que tem apenas um canal. Além disso, todos os pixels que tiverem menos que 5% de variação de intensidade em relação ao pixel clicado deve ser trocado para a cor vermelha.

Assim como o desafio é realizado com uma imagem, ele também deve ser feito quando coloca-se um vídeo como parâmetro de entrada do programa ou quando é determinado a abertura de streaming de vídeo da própria câmera do computador ou de algum dispositivo USB externo.

Para a realização dessa atividade, foram utilizados vídeos e imagens diversas, tanto coloridas quanto grayscale, para verificar a manipulação correta dos pixels clicados pelo usuário. Quando o programa é executado sem parâmetro de entrada, é aberto o streaming de vídeo com a câmera do computador. Não foi possível realizar o procedimento com uma câmera USB externa devido à falta de equipamento, porém os testes realizados em tempo real com a câmera interna garantem o funcionamento correto do programa com qualquer dispositivo USB.

A metodologia aplicada para lidar com o pixel clicado pelo usuário foi usar uma função que é chamada quando o botão esquerdo do mouse é apertado em cima da imagem. Ela recebe como parâmetros as posições x e y do pixel na matriz da imagem, possibilitando mostrar no terminal os valores de cada canal RGB. No caso de uma imagem em escala de cinza, é verificado se todos os canais de cada pixel obtém o mesmo valor e, se isso for verdade, mostra-se no terminal apenas o valor de intensidade.

Já para verificar quais pixels estão dentro dos valores exigidos para serem trocados para vermelho, deve ser percorrida toda a imagem checando essa condição. Após o clique do usuário em um frame do vídeo, será mantido o critério de mudança para o vermelho até que o usuário clique em outra posição da imagem. Dessa forma, antes de mostrar cada frame é aplicado o processamento de mudança de cor descrito acima.

Vale deixar claro que, para o desenvolvimento dos algoritmos, foi utilizado a linguagem de programação C++, assim como técnicas de orientação a objetos. Na hora de execução do código no terminal, pode ter como parâmetro de entrada uma imagem ou vídeo, e se nenhum parâmetro for inserido, será considerado a abertura de um streaming de vídeo local.

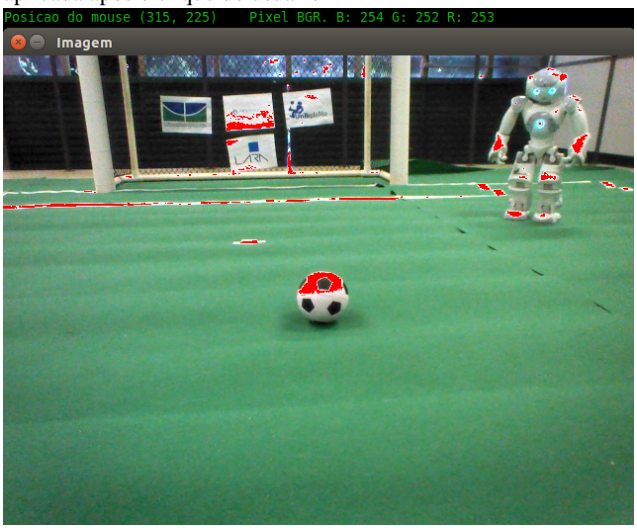
4. Resultados

Figura 3. Figura colorida

Fonte: Próprio autor

Para uma imagem colorida como parâmetro de entrada, mostrada na figura 3, foi gerada uma máscara a partir do clique do usuário que, aplicada sobre a imagem original, resultou na seguinte figura.

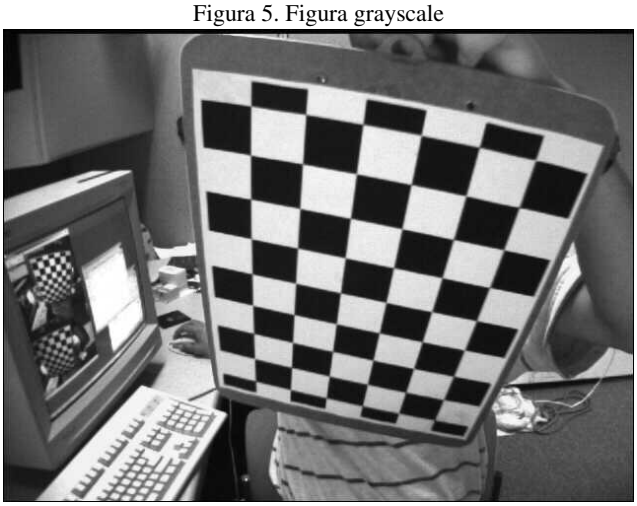
Figura 4. Representação dos canais RGB e da máscara vermelha aplicada após o clique do usuário



Fonte: Próprio autor

Como visto na figura 4, todos os pixels que tiverem 5% de variação em relação ao valor do pixel mostrado na posição (315,225) foram mudados para vermelho. Além disso, no topo da figura, observa-se a intensidade de cada canal da posição escolhida pelo usuário.

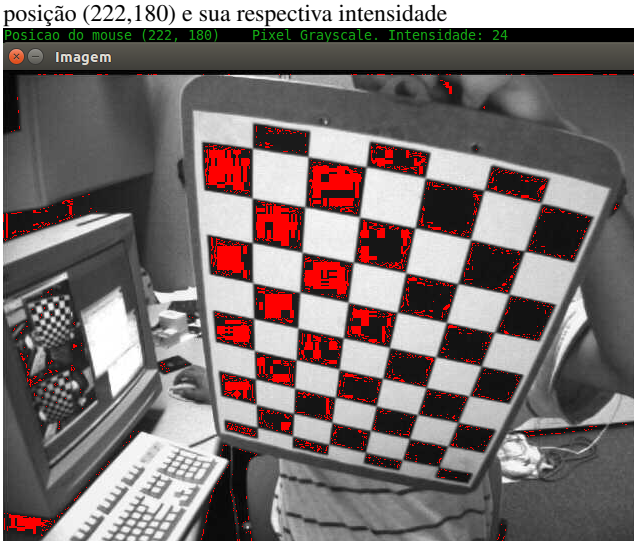
Figura 5. Figura grayscale



Fonte: Próprio autor

Para verificar o funcionamento correto do algoritmo para imagens em escala de cinza, mudou-se a entrada do programa para a figura 5 e obteve-se os seguintes resultados.

Figura 6. Transformação da imagem após o clique do usuário na posição (222,180) e sua respectiva intensidade



Fonte: Próprio autor

Conforme visto na figura 6, observa-se que o algoritmo soube diferenciar a entrada de imagens grayscale e mostrou no terminal (parte superior da figura) apenas a intensidade do pixel clicado. Além disso, a máscara vermelha foi aplicada corretamente com base na posição escolhida pelo usuário.

Fazendo os mesmos testes para vídeos e streaming local, percebeu-se que os resultados foram os mesmos mostrados nas avaliações feitas acima. Além disso, em cada frame posterior ao clique do mouse, é aplicada a máscara vermelha para os valores determinados. Apesar que cada frame deve ser percorrido pixel a pixel, nota-se que o algoritmo consegue processar em tempo real as mudanças necessárias.

5. Conclusão

Ficou claro o funcionamento dos principais métodos, ferramentas e estruturas da biblioteca OpenCV, dando uma boa base para qualquer aplicação futura. Além disso, verificou-se na prática os principais conceitos de uma imagem, como o pixel e o canal de cor. Conclui-se também que o projeto foi realizado com sucesso, visto que atendeu as demandas exigidas mesmo com aplicações em tempo real.

Referências

[1] G. Bradski and A. Kaehler, *Learning OpenCV*, 1st ed. O'Reilly.

[2] Forsyth and Ponce, *Computer Vision - A Modern Approach*, 1st ed. Pearson.

[3] OpenCV API Reference. (2017, 31 Março). [Online]. Available: http://docs.opencv.org/2.4/modules/refman.html

[1] [2] [3]