Projeto Demonstrativo 1 - Explorando OpenCV (25/03/2018)

Tiago de Carvalho Gallo Pereira (14/0164138)
Universidade de Brasília - UnB
Princípios da visão computacional - Turma A
tiagodecarvalhopereira@gmail.com

Abstract

Projeto demonstrativo para aprender a instalar e utilizar o OpenCV. Aprendendo a utilizar funções básicas como ler e mostrar uma imagem ou um vídeo, iterar nos pixels da imagem, analisar eles e tomar decisões em cima da análise.

1. Introdução

O OpenCV é uma das maiores ferramentas *open source* de visão computacional disponível. Aprender a usar essa ferramenta e os princípios básicos de como uma imagem é vista pelo computador e como trabalhar com ela computacionalmente abre muitas possibilidades de aplicação. Para desenvolvimento desse primeiro projeto demonstrativo precisamos entender os seguintes pontos.

1.1. Representaç^{a}o computacional da imagem

O computador "enxerga"uma imagem como uma matriz tridimensional, onde as duas primeiras dimensões mapeiam os pixels da imagem, e cada pixel da imagem contém 3 canais (valores). Os 3 canais de um pixel são os canais "R"(*red*), "G"(*green*) e "B"(*blue*), onde cada um deles representa a intensidade dessa cor no pixel. O valor dessa intensidade de cor costuma ser um número inteiro não sinalizado variando de 0 a 255, onde 0 significa ausência da cor e 255 significa intensidade máxima dessa cor. A figura 1 mostra como o conjunto desses 3 canais permite a representação de todas as cores conhecidas.

As imagens em preto e branco costumam ser representada com apenas 1 canal com o valor variando de 0 (preto) a 255 (branco), onde os valores intermediários representam a intensidade do tom de cinza. No entanto, esse tipo de imagem pode também ser representada com 3 canais igual as imagens coloridas, nesse caso pixel da imagem terá o mesmo valor de intensidade de cor em cada um dos seus canais.

A partir dessas informações, podemos também definir como funciona a transformação da imagem colorida para a imagem em preto e branco. Para isso basta calcular a média aritmética dos valores dos 3 canais e armazenar o resultado em 1 canal que representa a intensidade do tom de cinza.

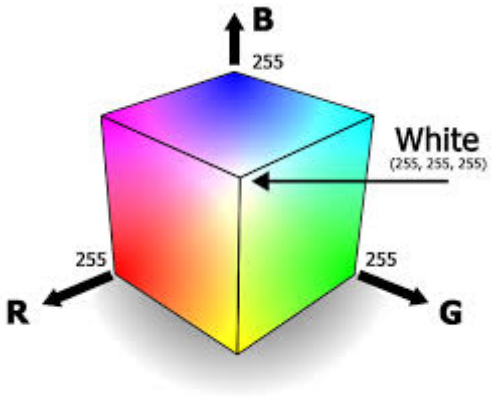

Figura 1. Cubo de cores RGB

1.2. Distância Eucliadiana

Existem alguns métodos para quantizar o quão próxima é a cor de dois pixels, nesse projeto demonstrativo iremos utilizar o conceito da distância euclidiana que é calculada segundo a equação 1. Se a distância euclidiana entre as cores de dois pixels for menor que 13, vamos considerar que os dois tem uma cor próxima. Já no caso das imagens preto e branco vamos ver se diferença entre a intensidade de cinza é menor que 13.

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

2. Materiais e métodos

Para a realização desse experimento foi utilizado um ambiente instalado com Python 3.5.2, OpenCV 3.2.0 e NumPy 1.14.2. Para cada um dos requisitos foi criado um arquivo com o código fonte em python, em cada arquivo foi criada uma classe que cumpria os requisitos desejados.

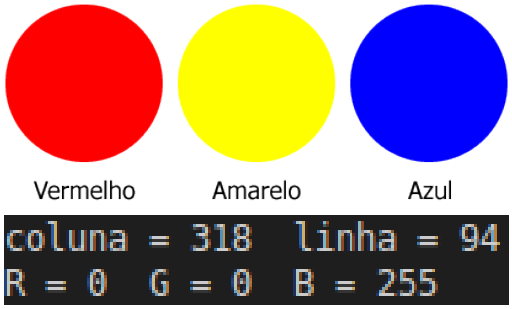

Figura 2. Resultado Requisito 1

3. Resultados

3.1. Requisito 1

O requisito 1 pedia para elaborar uma aplicação que abrisse uma imagem no formato JPG e permitisse que o usuário clicasse em algum ponto da imagem, quando o usuário clicasse em algum ponto da imagem, a aplicação deveria retornar no terminal a linha e coluna correspondentes a imagem e o valor armazenado naquela posição do pixel da imagem. A figura 2 mostra, na parte de cima, uma imagem utilizada para teste e, na parte de baixo, a saída da aplicação quando o usuário clicou em um ponto no círculo azul.

3.2. Requisito 2

O requisito 2 pedia para elaborar uma aplicação que fizesse as mesmas coisas que o requisito 1 e ainda colorisse na imagem todos os pixels que tivessem uma cor parecida com a cor do pixel clicado. Para definir as cores parecidas foi feito o cálculo da distância euclidiana como descrito na equação 1, se o resultado da equação fosse menor que 13, consideramos que as cores são parecidas e pintamos todos os pixels de cores parecidas com a cor vermelha pura. A figura 3 mostra a imagem de saída da aplicação quando a imagem de entrada foi a mesma que a imagem superior da figura 2 e o usuário clicou em algum ponto do círculo azul.

3.3. Requisito 3 e 4

Os requisitos 3 e 4 pediam que fossem elaboradas aplicações que cumprissem os mesmos critérios do requisito 2,

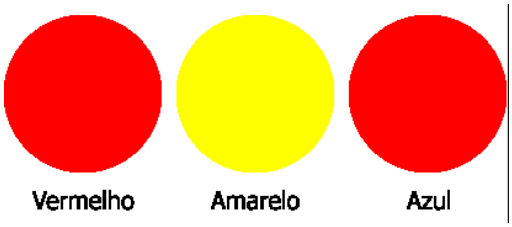

Figura 3. Resultado Requisito 2

Figura 4. Resultado Requisito 3

mas agora ao invés de termos uma imagem estáticas, o trabalho será feito com vídeo para o requisito 3 e com o uso da webcam para o requisito 4. As figuras 4 e 5 mostram possíveis saídas dessas aplicações.

4. Discussão e Conclusões

Nos resultados dos requisitos 3 e 4, podemos analisar uma situação que pode parecer estranha, tal situação é que a aplicação deve pintar de vermelho puro todos os pixels de cores parecidas. No entanto, no resultado do requisito 4 que pode ser conferido na imagem 5 podemos ver que o usuário clicou em um ponto da camisa que é lisa e com apenas uma cor, mas a aplicação não pintou toda a camisa. Isso ocorre porque quando nós olhamos para algo com nossos olhos conseguimos ignorar um pouco as diferenças que a quantidade de luz pode causar na cor e aí temos a impressão de que aquele objeto está todo da mesma cor. Entretanto, o computador analisa apenas uma imagem capturada e com isso, a diferença de iluminação em certos pontos do objeto fazem com que ele tenha uma gama maior de cores e por isso a aplicação não pintou a camisa toda, pois analisando a imagem capturada naquele momento a camisa certamente não era da mesma cor em sua totalidade.

Essa situação descrita tem de ser levada em consideração em várias aplicações de visão computacional, pois há muitas aplicações onde queremos encontrar algo em uma

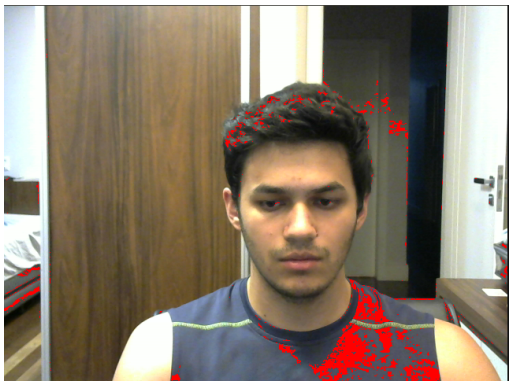

Figura 5. Resultado Requisito 4

imagem e optamos por colocar uma etiqueta ou marcador naquele ponto para identificarmos ele mais facilmente. É natural pensarmos em colocarmos uma cor marcante nessa etiqueta, no entanto dependendo da iluminação do ambiente, o computador pode não encontrar essa etiqueta e com isso toda a sua aplicação seria perdida, logo devemos sempre buscar mais de um tipo de validação e evitar usar fatores que envolvam apenas a cor do objeto sozinho.

Vários métodos do OpenCV utilizados nesse projeto são simples de serem implementados a mão, no entanto conforme os projetos vão crescendo, tentar implementar todos os métodos a mão pode nos levar a uma incrível perda de tempo e eficiência. Logo, é muito importante se familiarizar com bibliotecas de apoio como o OpenCV para facilitar o desenvolvimento de diversas aplicações de visão computacional.

Referências

[1] OpenCV. http://www.opencv.org. Acessado em 25 de março de 2018.

[2] OpenCV. *Mouse as a Paint-Brush*. OpenCV Example link. Acessado em 18 de março de 2018.