Projeto Demonstrativo 1 - Explorando OpenCV

Natalia Oliveira Borges - 16/0015863
natioliveira97@hotmail.com

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

Abstract

Para esse projeto foram propostos quatro exercícios que têm como objetivo introduzir as principais funcionalidades da ferramenta OpenCV. Para realiza-los foram utilizadas funcões para leitura e escrita de imagens, além de algumas funcionalidades de inteface com usuário próprias do OpenCV.

1 Introdução

OpenCV é uma biblioteca open source de visão computacional livre para uso acadêmico e comercial. Nela exitem várias funções para tratamento de imagem, que foram otimizadas especificamente para essa tarefa. Além de funcões para processamento de imagem, como filtros e reconhecimento de features, o OpenCV também possui funções básicas de interface com usuário, como criação de janelas e reconhecimento de entrada de mouse e teclado. Esse projeto tem o objetivo de explorar essas funções.

O projeto possui quatro requisitos. O primeiro consiste em abrir uma imagem e ao clicar em um pixel, mostrar a intensidade (RGB ou grayscale). O segundo contempla o primeiro, porém, ao clicar no pixel, os pixels de cor até 13 tons próximos do pixel escolhido devem ser pintados de vermelho. O terceiro contempla o segundo, porém no lugar da imagem, deve-se usar um vídeo .avi ou x264. E o quarto também contempla o segundo, porém, em vez de imagem, deve-se abrir um streaming de video na webcam.

2 Metodologia

Para realizar o projeto foi criada uma classe imageClass que possui objetos e métodos para armazenar informações sobre imagem e requisito escolhido pelo usuário.

Em todos os requisitos é necessário receber a informação do mouse que indica qual pixel foi clicado pelo usuário. Para realizar essa tarefa foi utilizada a funcão cv::setMouseCallback (const string& winname, MouseCallback onMouse, void* userdata=0) [1]. Os argumentos dessa função são:

- 1. string& winname: o nome da janela, que foi criada usando cv::namedwindow().

- 2. MouseCallback onMouse: a função que indica o que deve ser feito quando o botão do mouse é pressinado. As funções do tipo MouseCallback tem o formato void onMouse(int event, int x, int y, int, void*), em que event indica qual botão do mouse foi apertado, x e y a coluna e linha da imagem, respectivamente, em que o click ocorreu, void * recebe o ponteiro para um tipo void qualquer que pode ser necessário nessa função.

- 3. void* userdata: userdata é o ponteiro void que pode apontar para um tipo qualquer, nesse projeto foi passado um ponteiro para classe imageclass.

As funções que realizam os requisitos estão em um arquivo functions.cpp e foram projetadas para serem reutilizadas por requisitos diferentes.

A funcão image() é utilizada pelos requisitos 1 e 2. Ela é responsável por abrir imagens, usando a função cv::imread(), criar uma janela, usando a função cv::namedwindow(), e mostrar a imagem na janela, usando a função cv::imshow(). É possível abrir tanto imagens RGB quanto grayscale, para saber qual das duas está sendo aberta ela checa o número de canais da imagem, se houver 1 a imagem é grayscale, porém é convertida para RGB usando a função cvtColor() para ficar com 3 canais. Se houver 3 canais a imagem ainda pode ser grayscale caso todos tenham o mesmo valor, então é chamada a função isRGB() que verifica essa condição. A função image(), por fim, chama a função setMouseCallback(), que recebe entrada do mouse.

A função video() é utilizada pelos requisitos 3 e 4 e é responsável por abrir tanto um arquivo video quando streaming usando a classe videoCapture do openCV. Um video não é mais que várias imagens sendo passadas rapidamente, então, após abrir o video, cada frame é processado separadamente como se fosse uma imagem, dessa forma as funções utilizadas para imagem puderam ser reutilizadas.

A função isRGB() verifica se os canais RGB de todos os pixel da imagem são iguais ou diferentes e retorna verdadeiro se a imagem for colorida e falso se for grayscale.

A função mouseClick() é do tipo MouseCallback e recebe as coordenadas do pixel clicado e em seguida escreve na tela a linha e coluna desse pixel e chama a função getPixelValues(). Ela também recebe como parametro um ponteiro para imageClass que indica o que deve ser feito dependendo do requisito selecionado pelo usuário.

A função getPixelValues() recebe como parametro as coordenadas x e y do pixel clicado e a imagem. Ela acessa o pixel e imprime na tela os valores RGB ou grayscale desse pixel, além de atribuir esses valores para a variável pixel da classe imageClass. A imagem recebida por essa função está sempre no formato RGB, já que as imagens grayscale abertas são convertidas para RGB. Para saber a verdadeira origem origem da imagem, essa função acessa a variável isRGB da classe imageClass e verifica se é falso ou verdadeiro.

A função drawRedPixels() recebe um ponteiro para a classe imageClass, que possui as informações que essa função precisa. Ela percorre cada pixel da imagem verifica a diferença de ton entre o pixel e o pixel clicado. Para isso é calculada a distância euclidiana do espaço tridimensional de cores usando a equação:

$$(R-R_0)^2+(G-G_0)^2+(B-B_0)^2<169 \quad (1)$$

Se a diferença de tons for menor que 13 tons, o pixel em análise é pintado de vermelho.

3 Resultados

3.1 Requisito 1

O requisito 1 funcionou como esperado. Ao selecioná-lo pergunta-se o nome do arquivo de imagem que deseja-se abrir. A imagem é aberta e ao clicar em pixels na tela o programa mostra as coordenadas do pixel escolhido e a intensidade da cor RGB ou grayscale.

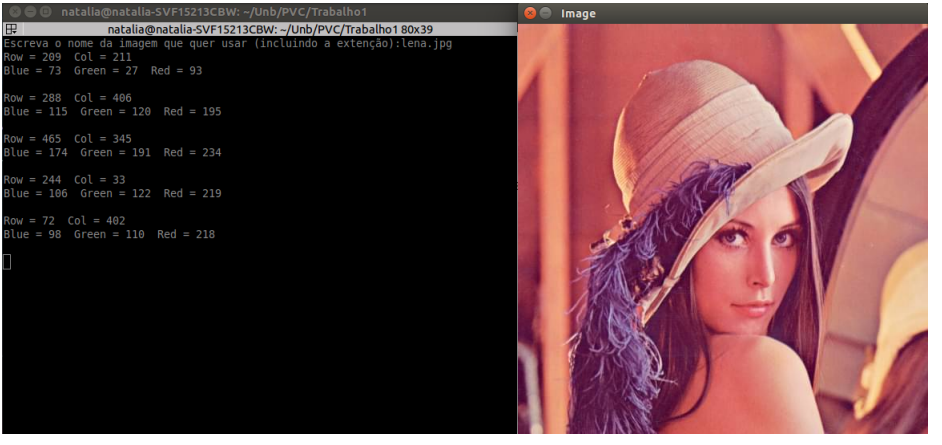

Figure 1: Requisito 1 - Exemplo de Imagem Colorida

Figure 2: Requisito 1 - Exemplo de Imagem Grayscale

3.2 Requisito 2

O requisito 2 é muito semelhante ao 1. Ao abrir a imagem e clicar em um pixel na tela, os pixel até 13 tons proximos dele são pintados de vermelho. O algoritmo utilizado compara todos os pixels para fazer essa tarefa.

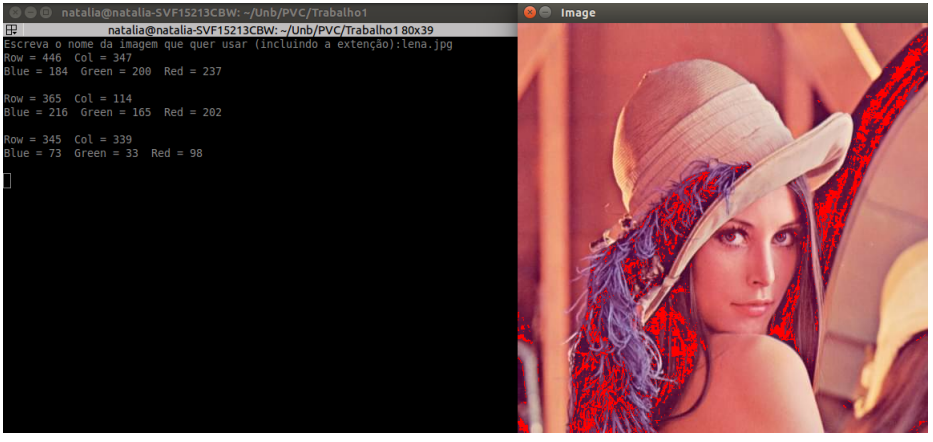

Figure 3: Requisito 2 - Exemplo de Imagem Colorida

3.3 Requisito 3

O requisito 3 faz o mesmo que o 2, porém usando videos. Os frames do vídeo são tratados como imagens e é realizado o mesmo processo de comparação entre o pixel escolhido e todos os outros pixel. Por causa disso, é possivel perceber que ao escolher um pixel o video fica mais lento, já que cada frame deve ser porcessado.

A abertura de videos, em princípio, não estava funcionando. Foi utilizado um conversor de videos online https://www.onlinevideoconverter.com/pt/contact para baixar um video do YouTube, e varios warnings apareciam na execução do programa. Ao utilizar um video .avi original o programa funcionou corretamente.

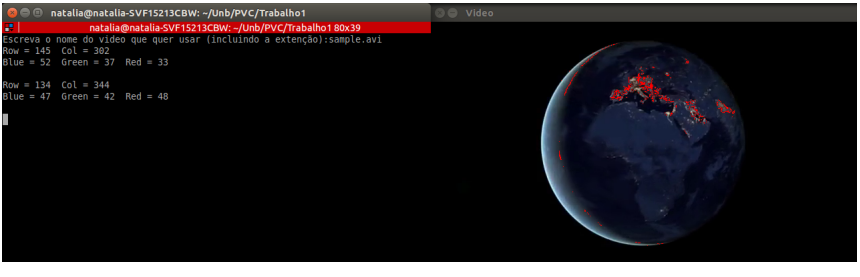

Figure 4: Requisito 3 - Videos

3.4 Requisito 4

O requisito 4 é praticamente identico ao 3, mas em vez de abrir um video, é aberto o streaming da câmera.

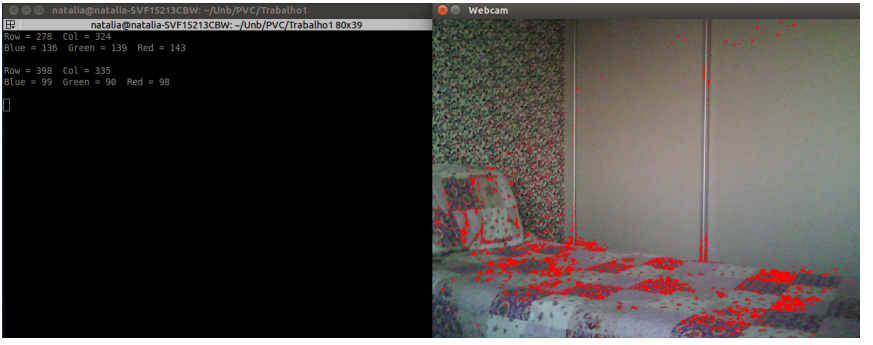

Figure 5: Requisito 4 - Webcam

4 Discussão e Conclusões

Esse projeto introdutório foi importante para explorar os conceitos basicos do OpenCV, como leitura e escrita em imagens e videos e recebimento de entrada de mouse e teclado usando algumas funções de interface com usuário.

Além disso verificou-se algumas propriedades importantes de imagens, como elas são armazenadas e como acessar um pixel nela. Usando a função cvtColor() também foi possível converter imagens de um espaço de cor para outro.

References

[1] OpenCV 2.4.13.7 documentation. User interface. URL https://docs.opencv.org/2.4/modules/highgui/doc/user_interface.html.

[2] Wikipédia. Opencv — wikipédia, a enciclopédia livre, 2018. URL https://pt.wikipedia.org/w/index.php?title=OpenCV&oldid=52528171.