Projeto demonstrativo 1
Explorando o OpenCV

Bruno Takashi Tengan
bt.tengan@gmail.com
Matrícula
12/0167263

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

Abstract

Este projeto tem como objetivo explorar e introduzir as funcionalidades básicas da biblioteca OpenCV, no caso desse trabalho em específico, em Python. As aplicações desenvolvidas focam em obtenção de dados e realizar comparação de semelhança entre os pixeis de uma imagem.

No fim, é notável a necessidade de bibliotecas de manipulação de matrizes para facilitar o desenvolvimento dos programas e obter um melhor desempenho em sua execução.

1 Introdução

OpenCV é uma biblioteca de software open source de visão computacional e machine learning. A biblioteca tem o objetivo de fornecer uma infraestrutura de fácil acesso para realizar visão computacional, afim de acelerar o avanço do desenvolvimento comercial e de pesquisa na área da visão computacional com alta eficiência de processamento.

Para explorarmos o OpenCV, foi feito 3 aplicações onde um lida com imagens, outro com vídeo no formato .avi e o último com streaming de vídeo. Todos eles o objetivo é com o mouse apontar um local na imagem e informar os valores de cor, no caso de imagem colorida, ou valor de luminosidade, no caso de imagem em escala de cinza, para o usuário e demarca com cor vermelha os lugares da imagem com cores semelhantes à amostrada.

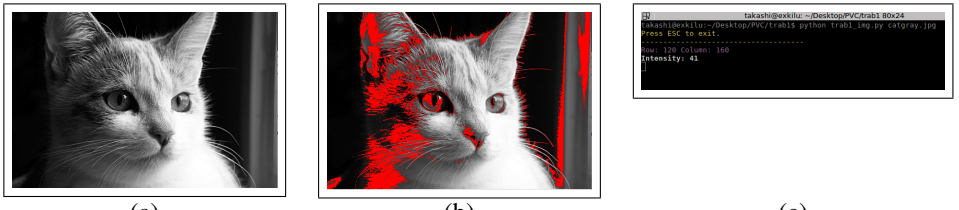
2 Metodologia

Os programas foram implementados em Python2.7 e foi utilizado principalmente as bibliotecas do OpenCV e do Numpy [1] para manipulação de matrizes multidimensionais. Os tutoriais fornecidos pelo próprio site da OpenCV [2] forneceram boas direções em como implementar os códigos.

Uma imagem ou frame de um vídeo é armazenado em forma de uma matriz no programa com seu tamanho definido pela largura e altura da imagem. Essas matrizes poderiam ser manipuladas fazendo loops no código para modificar elemento por elemento dessa matriz, mas com a biblioteca Numpy podemos fazer as operações matricialmente e isto fará uma diferença crucial no desempenho do programa, principalmente quando lidando com vídeos.

(a) (b) (c)

Figure 1: Exemplo de funcionamento da aplicação que analisa imagem com imagem colorida: (a) imagem original quando aberto o programa; (b) imagem após selecionado um ponto na imagem original com os pixeis de cor semelhantes realçados com a cor vermelha; (c) mensagem no terminal do programa informando informações sobre o pixel selecionado pelo usuário.



(a) (b) (c)

Figure 2: Exemplo de funcionamento da aplicação que analisa imagem com imagem em escala de cinza: (a) imagem original quando aberto o programa; (b) imagem após selecionado um ponto na imagem original com os pixeis de luminosidade semelhantes realçados com a cor vermelha; (c) mensagem no terminal do programa informando informações sobre o pixel selecionado pelo usuário.

Para avaliação da imagem e do vídeo como do tipo escala de cinza ou colorida é feito inicialmente uma avaliação da imagem e vídeo todo para ver se todos os seus pixeis possui um valor igual entre seus canais de cor. Saber se a imagem ou vídeo é em escala de cinza nos permite trabalhar apenas com um canal que é luminosidade e nos poupar operações no código.

3 Resultados

3.1 Imagem

O programa para análise de imagem funcionou como esperado. O que foi possível perceber foi a clara melhora no desempenho quando deixei de analisar os pixeis da matriz um por um e passei a utilizar os métodos da biblioteca do Numpy.

3.2 Vídeo

No programa responsável pelo vídeo se percebe uma grande diferença de desempenho quando comparado a execução de quando o vídeo é em escala de cinza e quando o vídeo é colorido. Se a restrição de quadros por segundo do vídeo for tirado, o vídeo em escala de cinza terminaria a execução do programa muito mais rápido que o colorido.

(a) (b)

Figure 3: Exemplo de funcionamento da aplicação que analisa streaming de vídeo: (a) vídeo da webcam antes de selecionar algum pixel; (b) vídeo da webcam após selecionado um pixel marcando os pixeis semelhantes de vermelho.

3.3 Streaming de vídeo

O programa de streaming de vídeo não teve resultados muito diferentes do que o programa de vídeos quando se executa com um vídeo colorido.

4 Conclusões

Os resultados durante o desenvolvimento do programa mostraram como é importante a otimização do código ao desenvolver programas para aplicações de visão computacional. Utilizar APIs ou bibliotecas já prontas e otimizadas como o OpenCV e o Numpy pode ser um caminho muito melhor e prático do que tentar recriar as operações que teria que usar de qualquer forma.

Se possível, utilizar imagem em escala de cinza é preferível para aplicações de visão computacional do que a imagem em RGB. O processo de conversão da imagem, já aliado com bibliotecas próprias para isso, mais que valem a pena o ganho de desempenho por não ter que analisar dados desnecessários de uma imagem.

References

[1] Numpy. Quickstart tutorials. https://www.numpy.org/devdocs/user/quickstart.html.

[2] OpenCV. Opencv-python tutorials. https://docs.opencv.org/3.1.0/d6/d00/tutorial_py_root.html.