Explorando o OpenCV, reconhecendo cores

Víctor Rodrigues Pacheco
Matrícula: 17/0063879
victorrpacheco98@gmail.com

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

Abstract

Esse trbalho exalta o uso biblioteca multiplataforma OpenCV como uma ferramenta para a visão computacional quando combinada à linguagem de programação Python por meio do desenvolvimento de códigos para a idenficiaçãode posições de pixels e cores e destaque de pixels específicos visando auxiliar a entrada de entusiastas no universo da visão computacional de forma didática e open source.

1 Introduction

A Biblioteca multiplataforma OpenCV [1] que teve seu desenvolvimento formentado pela intel na virada do milênio auxilia a entrada de leigos no universo da visão computacional e traz avanço e unificação para os mais experientes. O processo de tratamento de imagem com base em suas cores é o primeiro prjeto de muitos que buscam mais conhecimento na área, se assimilando ao "Hello World" e ao "Piscar LED". Usuários de OpenCV podem utilizar desse artigo para a introdução ao uso da linguagem com a biblioteca. É importante ressaltar o caráter Open Source da linguagem e sua grande comunidade apta para auxiliar. Neste trabalho foram desenvolvidos para análise de imagens pixel a pixel e tratamento de imagem para destaque com base em cor tanto para arquivos JPG quanto para arquivos AVI. Isso visa capacitar de forma técnica para propiciar o desenvolvimento de projetos mais complexos futuramente.

1.1 Repositório

Os código desenvolvidos nesse projeto estão disponíveis de forma Open Source na platafora GitHub no repositório do autor disponível em https://github.com/VictorRPacheco/ExplorandoOpenCV. Para saber mais sobre o uso da ferramenta GitHub é recomendado o tutorial de Roger Dudler https://rogerdudler.github.io/git-guide/index.pt_BR.html que introduz o básico da ferramenta

2 Desenvolvimento

2.1 Sistema

Todos os códigos foram desenvolvidos e testados em uma máquina virtual Oracle VirtualBox [4] com as seguintes características:

- Versão VM 6.0.4
- Versão OS: Ubuntu 16.04 LTS 64-bits
- Versão Python: 2.7
- Versão OpenCV: 3.2.0

A máquina virtual utilizada está disponível em https://mega.nz/#F!MEgzWSiS!8HuAvnv2mFd0pyBnl2VVjg de autoria de Mateus Berardo de Souza Terra.

2.2 Requisitos

O sistema atende quatro requisitos, especificados a seguir:

1. Elabore uma aplicação utilizando OpenCV que abra um arquivo de imagem (do tipo JPG) e que permita ao usuário clicar com o botão esquerdo do mouse sobre um ponto na área da imagem e mostre no terminal a coordenada do ponto (row,column) na imagem, informando os valores do pixel RGB, quando a imagem for colorida ou o valor da intensidade do pixel quando a imagem for em nível de cinza (greyscale).

2. Repita o procedimento desenvolvido no Requisito 1 e crie uma rotina de seleção de pixels baseado na cor de onde for clicado. Seu programa deve comparar o valor da cor (ou tom de cinza) de todos os pixels da imagem com o valor da cor (ou tom de cinza) de onde foi clicado. Se a diferença entre esses valores for menor que 13, marque o pixel com a cor vermelha e exiba o resultado na tela.

3. Repita o procedimento desenvolvido no Requisito 2, em que ao invés de abrir uma imagem, abra um arquivo de vídeo (padrão avi ou x264) e realize os mesmos procedimentos do Requisito 2 durante toda a execução do vídeo. Cada vez que o usuário clica na imagem, o valor de referência deve ser atualizado.

4. Repita o procedimento desenvolvido no Requisito 3, em que ao invés de abrir um arquivo de vídeo, a aplicação abra o streaming de vídeo de uma webcam ou câmera USB conectada ao computador e realize todos os procedimentos solicitados no requisito 3.

2.3 Funcionamento

Para o funcionamento do código é necessário possuir OpenCV versão 3 compativel instalado em sua máquina, além de Python 2. Para executar o código basta entrar na parta /src e executar o comando:

```
python Trabalho1.py
```

Logo o programa indaga o usuário sobre qual função ele deve executar conforme os requisitos descritos. Dependendo da opção escolhida ele pede uma entrada de imagem ou vídeo para a correta execução ou que opte por um dos arquivos padrão de teste. Com o arquivo aberto ele chamada a função Cor_e_Posicao resposável por identificar a posição e cor do pixel selecionado pelo usuário utilizando o cursor do mouse e logo em seguinda mostra o resultado no terminal.

```
def Cor\_e\_Posicao(event, x, y, flags, param):\n
  global clickPoint
  global color

  if event == cv2.EVENT_LBUTTONDOWN:
    clickPoint = [x, y]
    color = image[y,x]
    print("----------------------")
    #Identidica se a imagem eh em tons de cinza
    if color[0] == color[1] == color[2]:
      print"Posicao [X, Y]: ", (x, y), "Intensidade: ",
color[0]
    else:
      print "Posicao [X, Y]: ", (x, y), "Cor: ", str(color)
```

Para o destaque em arquivos Preto e branco foi utilizado como critério a intensidade do pixel, variando de 0 a 255 destacando os que satisfazem a equação 1:

$$|p1 - p0| < 13 \tag{1}$$

Onde $(p0)$ é o valor de um pixel indicado pelo usuário com o auxilio do cursor do mouse e $(p1)$ representa o valor do pixel analisado para destaque.

Para arquivos coloridos que apresentam cores em 3 camadas é utilizada a diferença euclidiana [2] para selecionar os pixels que devem ser destacados satisfazendo a equação 2 e 3:

$$\Delta euclidiano = \sqrt{|R1 - R0|^2 + |G1 - G0|^2 + |B1 - B0|^2} \tag{2}$$

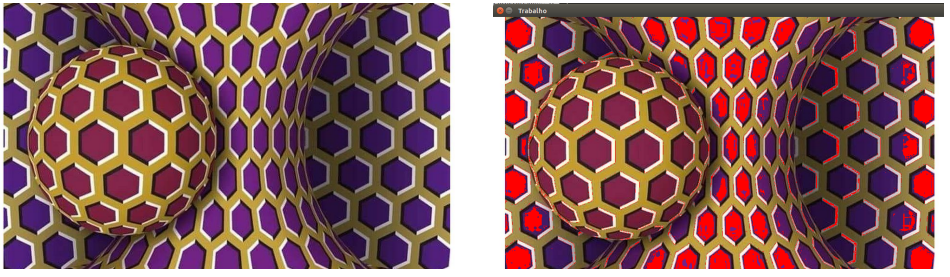
$$\Delta euclidiano < 13 \tag{3}$$

Onde o valor de $(\Delta euclidiano)$ da 2 é calculado com base nos valores $(R1)$, $(G1)$ e $(B1)$ que representam a cor do pixel que está sendo analizado e $(R0)$,$(G0)$ e $(B0)$ provenientes de um pixel indicado pelo usuário. Logo em seguida esse valor é utilizado em uma comparação com segundo a equação 3 para gerar um valor booleano indicando True para pixels que devem ser destacados. O destaque ocorre utilizando a biblioteca NumPy [1] para otimizar o uso das matrizes.

3 Resultados

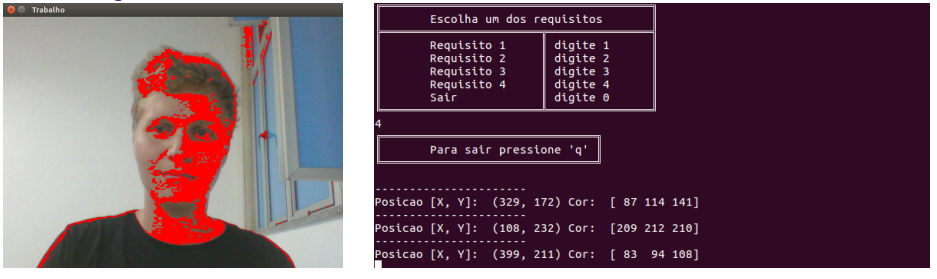
Utilizando como base o tutorial [6] para criação da função de identificação de cor e posição dos pixels selecionados pelo usuário permetiu a imprimir no terminal os dados do pixel selecionado. No momento de criação da matriz contendo os pixels que devem ser destacados

seguindo as equações 2 e 3 se obteve uma boa perfomace para imagens, porém ao se utilizar uma máquina virtual com recursos limitados há quedas de frames para vídeos de resoluções elevadas no momento de destaque de cores similares. O uso da camera trouxe bons resultados, sendo utilizado o tutorial [4], funcionou quase que em tempo real com quantidade constante de fraimes.



(a) (b)

Figure 1: (a) Imagem de entrada do requisito2; (b) Saída após o usuário ter clicado em um ponto no requisito2.



(c) (d)

Figure 2: (c) Saída do requisito 4 após clique do usuário; (d) Saída no terminal com as coordenadas e cor do ponto do requisito 4.

4 Conclusão

Pode-se concluir que o trabalho foi bem sucedido pois foi possível cumprir todos os requisitos de maneira eficiente utilizando a biblioteca OpenCV alinhada com a linguagem Python. Outro ponto a ser ressaltado é o desenvolvimento técnico que foi alcançado permitindo o desenvolvimento de futuros projetos envolvendo a biblioteca OpenCV para visão computacional de forma mais intuitiva.

References

[1] NumPy developers. Numpy, apr . URL http://www.numpy.org/.

[2] Kizar. Distância eucliadiana. URL https://pt.wikipedia.org/wiki/Dist%C3%A2ncia_euclidiana.

[3] Justin Mitchel. Web camera quick test. URL https://www.codingforentrepreneurs.com/blog/opencv-python-web-camera-quick-test/.

[4] Oracle. Oracle vm virtualbox. URL https://www.virtualbox.org/.

[5] OpenCV org. Opencv library. URL https://opencv.org.

[6] PyImageSearch. Capturing mouse click events with python and opencv. URL https://www.pyimagesearch.com/2015/03/09/capturing-mouse-click-events-with-python-and-opencv/.