Visão por Computador 2016-17, Guia Prático N.º 3

Rui Oliveira, Tomás Rodrigues
DETI, Universidade de Aveiro
Aveiro, Portugal
{ruipedrooliveira, tomasrodrigues}@ua.pt

Resumo –
Pretende-se através deste relatório expor sob forma escrita, o nosso desempenho e objetivos alcançados na aula prática n.3 da unidade curricular de Visão por Computador do Mestrado Integrado de Engenharia de Computadores e Telemática.
Neste relatório pretenderemos explicar as soluções por nós encontradas para a resolução dos diferentes problemas propostos.

***Palavras chave* – visão, computador, imagem digital, opencv, c++,**

I. REPOSITÓRIO: CÓDIGO FONTE

Todas as soluções dos problemas propostos estão disponível através do seguinte repositório (gitHub) criado para o efeito.

http://github.com/toomyy94/CV1617-68779-68129

A resolução dos problemas do presente guia encontram-se na pasta aula3.

II. PROBLEMAS PROPOSTOS

A. *Problema #1*

A.1 Enunciado

Implement a program to capture images from your digital camera (VideoCapture class) and include the capability of showing the grayscale and the black and white version of the acquired image. Use all the functionalities of the OpenCv library, namely the use of the functions cvtColor and adaptiveThreshold.

A.2 Resolução e principais conclusões

Inicialmente começámos por converter para escala de cinzento (`CV_BGR2GRAY`) a imagem obtida através da class `VideoCapture`. Posteriormente utilizá-mos o método `adaptiveThreshold` com os argumentos escolhidos pelo utilizador. Este método apenas pode ser aplicado a imagens com um único canal de 8 bits, daí a conversão para escala de cinza.
Mais especificações sobre a função `adaptiveThreshold` e os seus argumentos podem ser consultados no manual online do OpenCV.

A imagem seguinte demonstra a interação com o utilizador que é permitida.

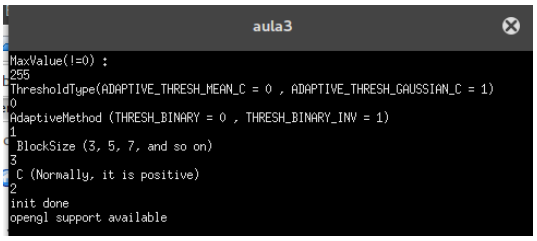

Figura 1: Interação com o utilizador

À esquerda encontra-se o resultado proveniente da conversão para escala de cinzento (`CV_BGR2GRAY`) e à direita a aplicação da função `adaptiveThreshold` com os parametros introduzidos pelo utilizador.

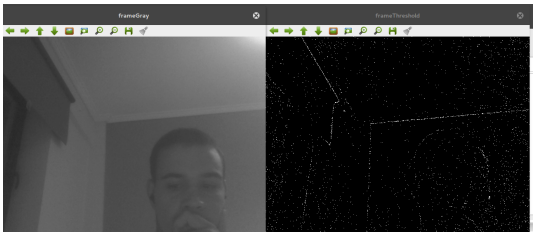

Figura 2: Resultado obtido

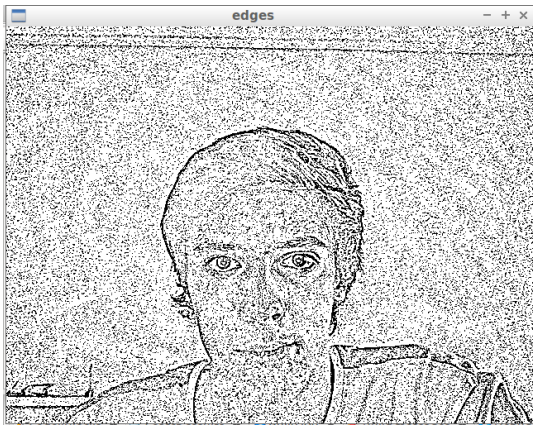

Figura 3: Resultado após execução do problema 1.

B. Problema #2

B.1 Enunciado

Include an option in the previous program to explore the use of the threshold function to obtain also a binary image.

B.2 Resolução e principais conclusões

Durante a recolha de imagem é possivel aplicar os seguintes thresholding, através a introdução dos seguintes caracteres pelo teclado.

- 0: Binary
- 1: Binary Inverted
- 2: Threshold Truncated
- 3: Threshold to Zero
- 4: Threshold to Zero Inverted
- q: Exit

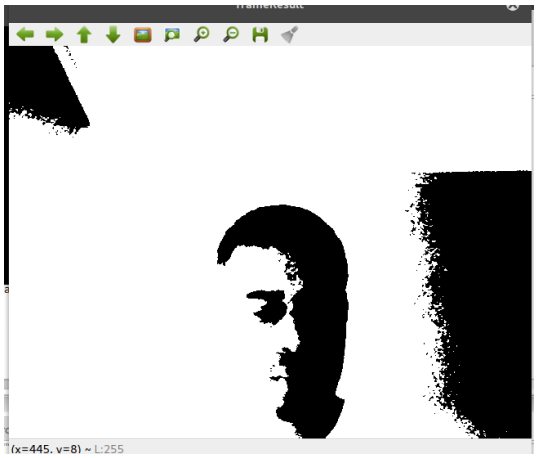

Figura 4: Binary

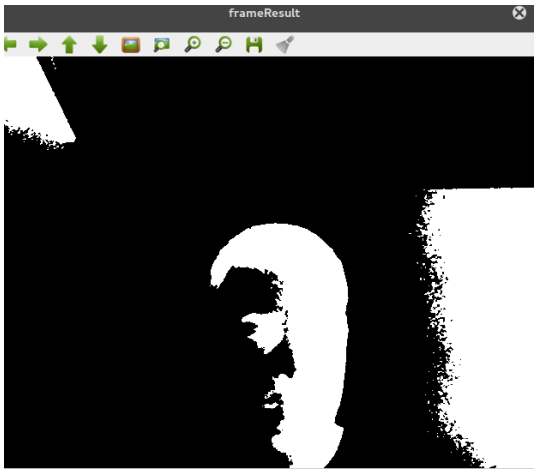

Figura 5: Binary Inverted

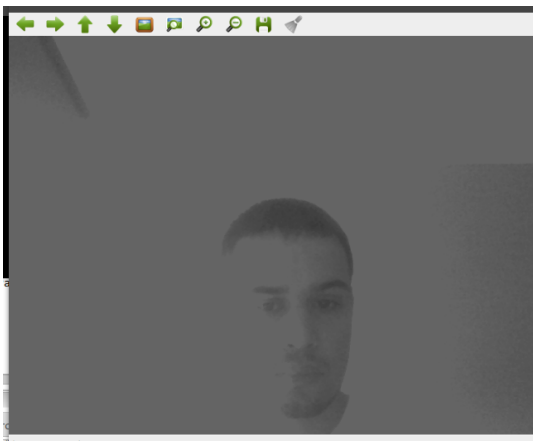

Figura 6: Threshold Truncated

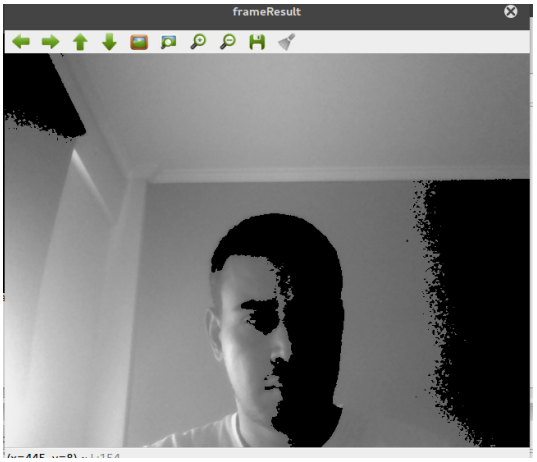

Figura 7: Threshold to Zero

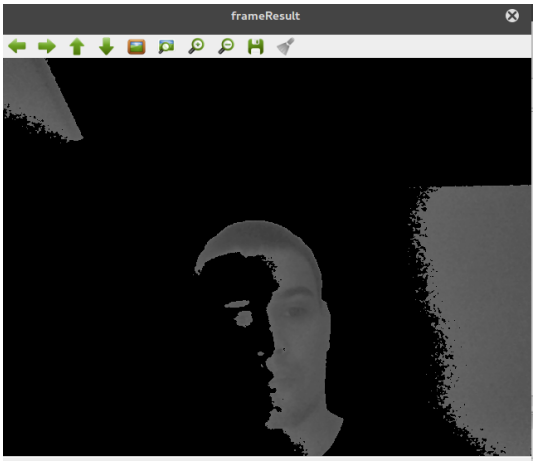

Figura 8: Threshold to Zero Inverted

C. Problema #3

C.1 Enunciado

Implement a program to perform a simple skin color detector based on chromaticity or other color properties.

C.2 Resolução e principais conclusões

Neste exercício tentámos obter um realce da pele humana atravéz essencialmente do contorno. Este abordagem não nos deu idealmente o que pretendíamos, falhando a detenção da pele em alguns aspetos como se pode ver na imagem abaixo.

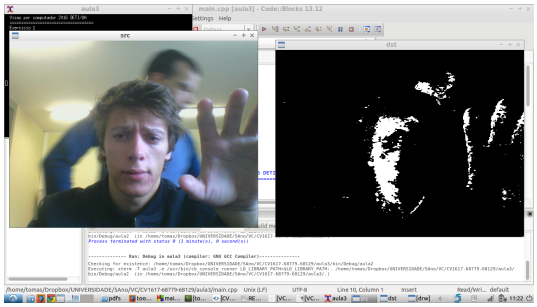

Figura 9: Resultado após execução do problema 3.

D. Problema #4

D.1 Enunciado

Implement a program to capture images from your digital camera and explore the use of filters to modify the images acquired.

D.2 Resolução e principais conclusões

Para este problema usámos duas funções de média que o opencv disponibiliza, a medianBlur e a gaussianBlur. Como era esperado verificámos que há medida que aumentávamos o tamanho do ksize a imagem ia ficando mais desfocada/smoth. Estas funções podem ser usadas para atenuar possível ruído em imagens!

```
medianBlur ( frame, frameResult, 15 );
```

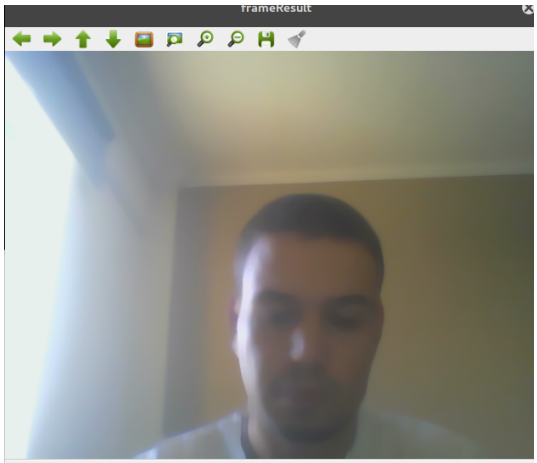

Figura 10: medianBlur

```
blur( frame, frameResult, Size( 10, 10
), Point(-1,-1));
```

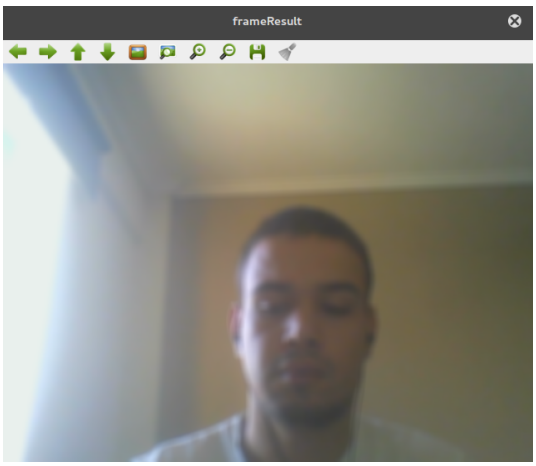

Figura 11: blur

E. Problema #5

E.1 Enunciado

Implement a program to capture images from your digital camera and calculate the histogram of each one of the chan

E.2 Resolução e principais conclusões

Depois de uma pequena pesquisa encontrámos uma solução ao problema que nos separa então a imagem em 3 componentes(R,G,B). Estabelecemos o número de barras do histograma em 256 e computámos e desenhámos o histograma nos 3 canais respetivos.

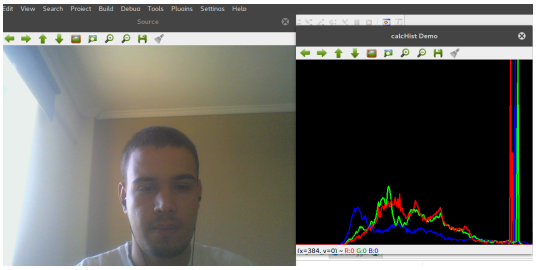

Figura 12: Histograma da imagem obtida pela camara. 3 Canais.

F. Problema #6

F.1 Enunciado

Include in the previous program the capability of apply histogram normalization. Visualize also the new histogram(s) obtained and comment the results.

F.2 Resolução e principais conclusões

Este problema foi resolvido com recurso á função do opencv já existente *equalizeHist*. Resultados obtidos podem ser encontrados abaixo.

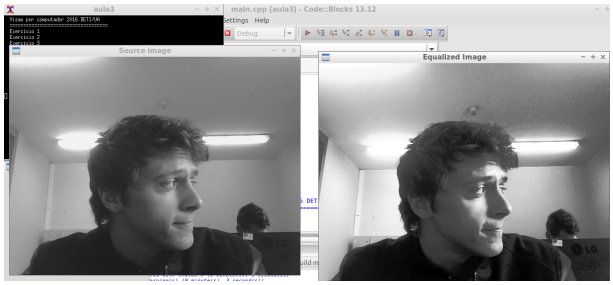

Figura 13: Resultado após execução do problema 6.

G. Problema #7

G.1 Enunciado

Implement a program that loads two images and compare the histograms of both.

G.2 Resolução e principais conclusões

Concluímos que os histogramas obtidos para as duas imagens – imagem original e a sua inversa – são exatamente iguais, uma vez que a distribuição de cores presentes são exatamente as mesmas.

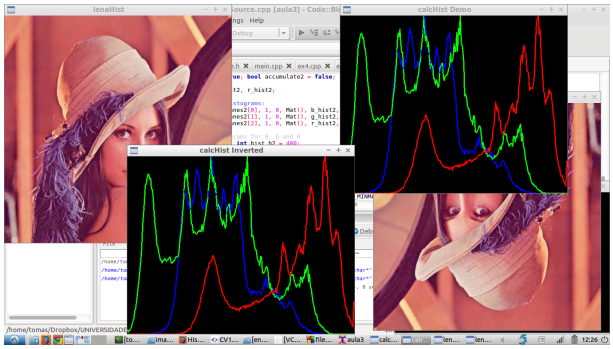

Figura 14: Resultado após execução do problema 7.

REFERÊNCIAS

[1] Neves, A. J. R.; Dias, P. Slides teóricos Visão por Computador - Aula 2 (2016)

[2] OpenCV. Opencv Documentation. Web. 24 Outubro 2016.