Visão por Computador 2016-17, Guia Prático N.º 4

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.4 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 aula4.

II. PROBLEMAS PROPOSTOS

A. Problema #1

A.1 Enunciado

Implement a program to capture images from your digital camera and explore the use of the func- tion Sobel to calculate the image of gradients. Calculate the gradient of the first order. Explore the parameters of the function and comment the results. Explore also the Scharr's algorithm.

A.2 Resolução e principais conclusões

Neste exercício foi usado o algoritmo de `Sobel` para calcular a imagem de gradiente. Para além deste algoritmo, também foi explorado o algoritmo de `Scharr's`.
Para este exercício aplicamos as seguintes operações:

1. `GaussianBlur`
2. `cvtColor` para `CV_BGR2GRAY`
3. `Scharr` ou `Sobel` para o gradiente X
4. `convertScaleAbs` para o grandiente X
5. `Scharr` ou `Sobel` para o gradiente Y
6. `convertScaleAbs` para o grandiente Y
7. Adição dos dois grandientes através do método `addWeighted`

O resultado obtido com o algoritmo de `Sobel` é o seguinte:

Figura 1: Resultado obtido após exercício 1

O resultado obtido com o algoritmo de `Scharr` é o seguinte:

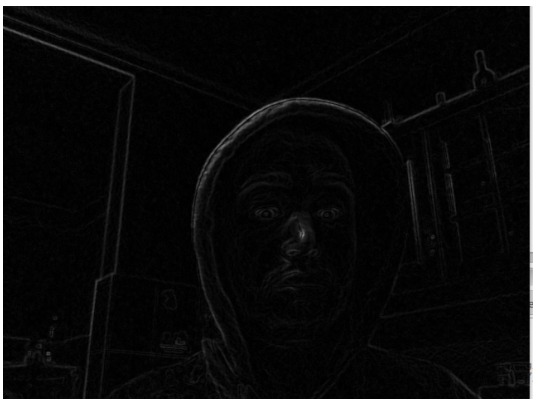

Figura 2: Resultado obtido após exercício 1

B. Problema #2

B.1 Enunciado

Based on the previous exercise, calculate also the Laplacian of an image. Explore the parameters of the function and comment the results.

B.2 Resolução e principais conclusões

Para a resolução deste exercício foram aplicadas as seguintes operações:

1. Aplicação do `GaussianBlur`

2. Conversão da imagem para grayscale através do método `cvtColor`
3. Aplicação do método `Laplacian`
4. Por fim é usado o método `convertScaleAbs`

Figura 3: Resultado obtido após exercício 2

C. Problema #3

C.1 Enunciado

Implement a program to capture images from your digital camera and perform edge detection with Canny's algorithm. Explore the effects of changing the parameters and comment the obtained results.

C.2 Resolução e principais conclusões

No problema 3 aplicámos o algoritmo de Canny diretamente e tal como nos foi dito na aula teórica, depois de ajustados os parâmetros conseguimos delinear as arestas mais importantes da imagem. Por exemplo, a imagem abaixo observámos facilmente que deteta com precisão muito mais arestas que a de cima, apenas alterando dois parametros `ratio` e o `threshold`.

Figura 4: Resultado após execução do problema 3.

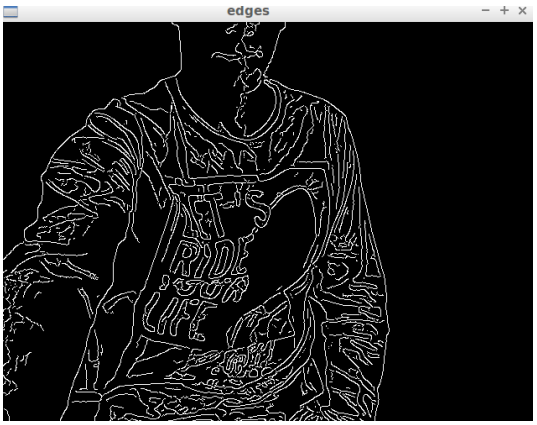

Figura 5: Resultado após execução do problema 3.

D. Problema #4

D.1 Enunciado

Implement a program to capture images from your digital camera and perform corner detection. Try for example the Harris's algorithm (explore the function cornerHarris). For each corner detected, draw a circle or a square in the image (explore the drawing functions of OpenCV).

D.2 Resolução e principais conclusões

No problema 4 detetámos os cantos diretamente com o algoritmo de Harris abordado na aula. Pensámos não ter conseguido uma boa solução mesmo quando tentamos ajustar alguns parâmetros como o `blocksize` e o tamanho da `aperture`, mas na imagem do tutorial seguida por nós os resultados pareciam a não detetar todos os cantos também. Concluímos portanto, que a deteção de cantos é muito mais difícil que a deteção de arestas para ser executado com uma precisão elevada.

Figura 6: Resultado após execução do problema 4

E. Problema #5

E.1 Enunciado

Implement a program to detect lines and circles on an image. As suggestion, start by calculating a binary image with the edges present on the scene. Adjust the parameters to get the best edges as possible. Then explore the use of the function findContours to identify the contours corresponding to the objects of interest.

E.2 Resolução e principais conclusões

Este exercício irá permitir detetar todos os círculos existentes na imagem e assiná-los. Foram usados os métodos `findContours` e `approxPolyDP`.

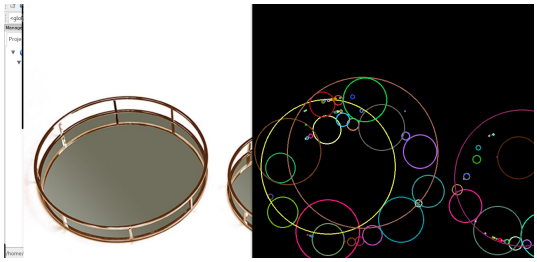

Figura 7: Resultado após execução do problema 5

F. Problema #6

F.1 Enunciado

Implement a different solution for the previous problem using the Hough Line Transform and the Hough Circle Transform.

F.2 Resolução e principais conclusões

Para a resolução deste exercício foram realizados dois tutoriais do OpenCV:

- *Hough Line Transform*
  Foram executadas as seguintes operações:
  1. Foi aplicado o método `Canny` ao imagem inicial, neste caso imagem proveniente da câmara.
  2. Foi usado o `cvtColor` para obter um conversão para `CV_GRAY2BGR`
  3. Foi usado o método `HoughLinesP`
  4. E por fim, o método `line`

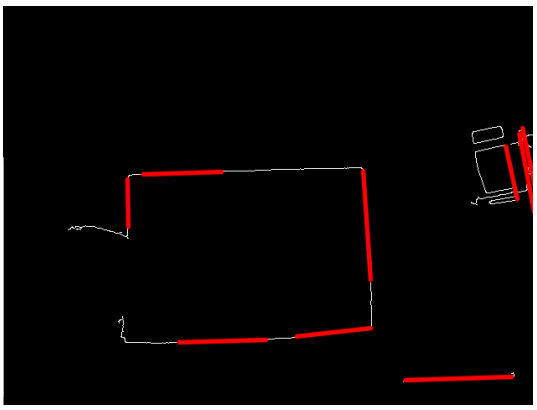

Figura 8: Resultado após execução do *Hough Line Transform*

- *Hough Circle Transform* Foram executadas as seguintes operações:
  1. Foi usado o `cvtColor` para obter um conversão para `CV_GRAY2BGR`
  2. Para reduzir o ruído foi usado o método `GaussianBlur`
  3. Foi usado o seguinte método `HoughCircles` para encontrar todos os círculos presentes na imagem
  4. Por fim, foi usado o método `circle` para desenhar o circulo detectado.

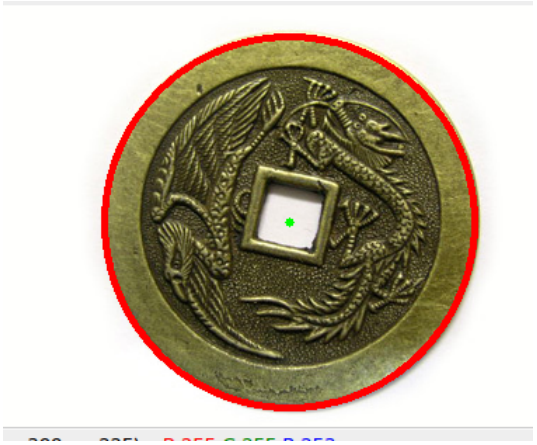

Figura 9: Resultado após execução do *Hough Circle Transform*

REFERÊNCIAS

[1] Neves, A. J. R.; Dias, P. Slides teóricos Visão por Computador - Aula 4 (2016)

[2] OpenCV. Opencv Documentation. Web. 15 Outubro 2016.