Visão por Computador 2016-17, Guia Prático N.º 11

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.º11 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, hough transform, object dectetor, 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 aula11. Para a resolução dos exercícios foi utilizado o CodeBlocks IDE. .

II. PROBLEMAS PROPOSTOS

A. *Problema #1*

A.1 Enunciado

Adapt the examples to read images from your digital camera and add the capability do display the edges of the acquired images (that are used internally by the algorithm). Include also a simple way to change the most important parameters of the hough functions (sliders, keyboard).

A.2 Resolução e principais conclusões

Para a resolução deste exercício foram realizados através do tutoriail `http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html` 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` desta vez com parametros variáveis
  4. E por fim, o método `line` para desenhar as linhas calculadas.

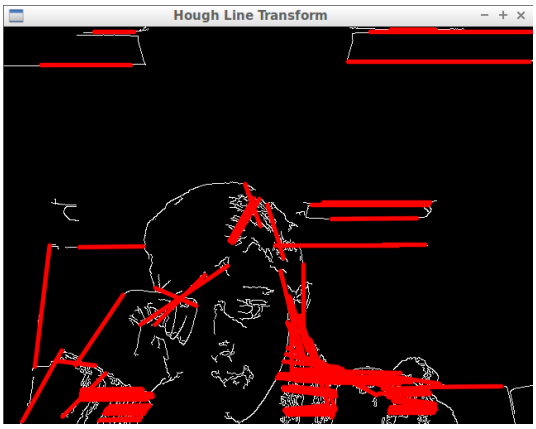

Figura 1: Resultado obtido após exercício 1

Aqui está lado a lado um exemplo de duas images com parâmetros diferentes: `xHoughLinesP(dst, lines, 1, CV_PI/180, 50, 50, 10 );` `HoughLinesP(dst, lines2, 1, CV_PI/180, 100, 0, 0 );` Num caso com o srn e o stn a zero e um threshold maior o que faz com que a deteção da linhas a desenhar não seja tão abrangente nas linhas detetadas como ”interessantes”

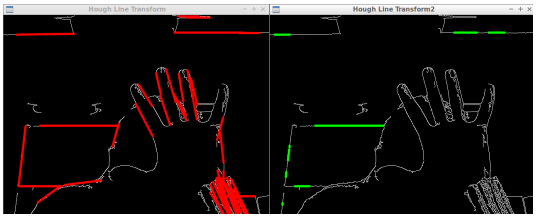

Figura 2: Resultado obtido após mudança de parâmetros na transformada

B. Problema #2

B.1 Enunciado

In some of the projects in this course, you have to perform detection of objects. Explore the feature detection and description provided by OpenCV, namely the Scale Invariant Feature Transform (SIFT), the Speeded Up Robust Features (SURF), FAST, among others.

B.2 Resolução e principais conclusões

Para a resolução deste problema foram criadas três funções:

- Função que nos permite verificar as diferenças entre o SIFT e o FAST
- Função que nos permite observar o resultado o SURF
- Função que nos permite observar o resultado o SURF com FLANN

As diferenças do FAST e SIFT aplicadas a uma imagem podem ser observadas na figura seguinte. Foram utilizados os objetos do tipo `SiftFeatureDetector` e `FastFeatureDetector`. Posteriormente foram criados vetores `KeyPoint` e de seguida utilizado o método `detect()`.

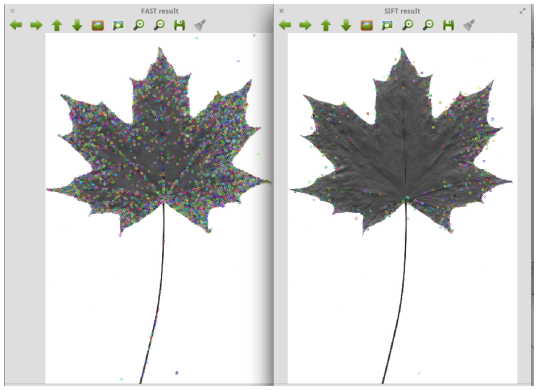

Figura 3: Resultado obtido após FAST e SIFT

Tal como no procedimento anterior foi criado um objeto do tipo `SurfFeatureDetector`.

Listing 1: Procedimento utilizado

```
std::vector<KeyPoint> keypoints_1;
detector.detect( img_1, keypoints_1 );
Mat img_keypoints_1;
drawKeypoints( img_1, keypoints_1,
    img_keypoints_1, Scalar::all(-1),
    DrawMatchesFlags::DEFAULT );
...
```

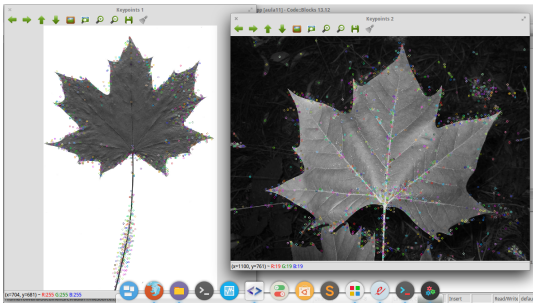

Figura 4: Resultado obtido após SURF

Para o exemplo seguinte recorremos ao tutorial disponibilizado pelo Opencv `http://docs.opencv.org/3.0-beta/doc/tutorials/features2d/feature_flann_matcher/feature_flann_matcher.html`

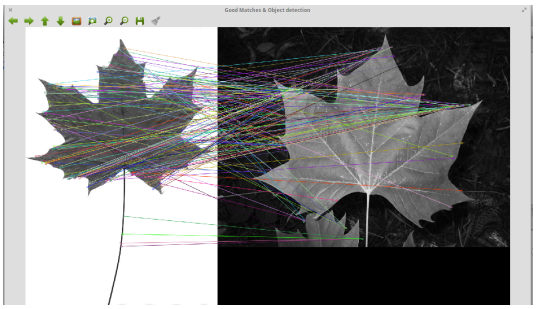

Figura 5: Resultado obtido após SURF com FLANN

REFERÊNCIAS

[1] Neves, A. J. R.; Dias, P. Slides teóricos Visão por Computador - Aula 11 (2016)

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

[3] Github. traffic-sign-detection. 8 Setember 2015.