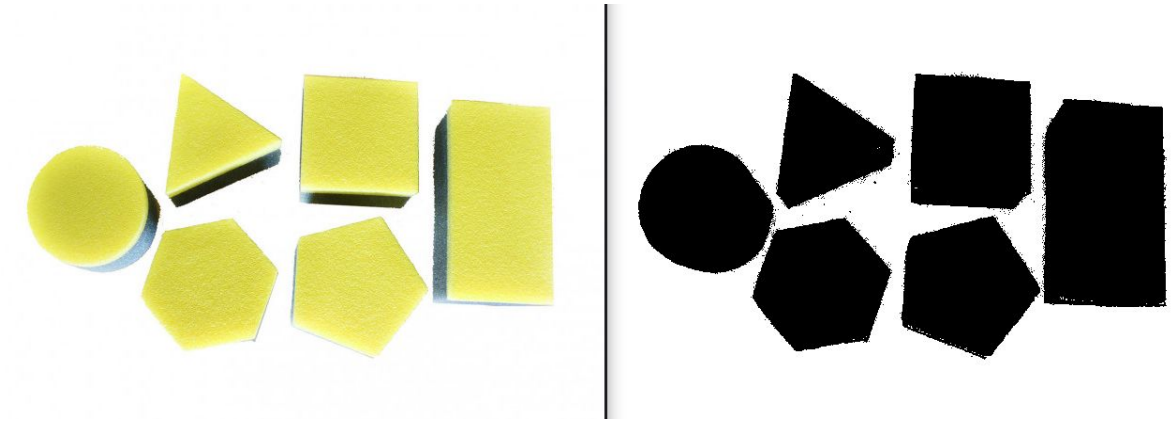
Computer Vision
PS-03

Nome: Caio Silva Gonçalves
Professor: Flávio Luis Cardeal Pádua

Centro Federal de Educação Tecnológica de Minas Gerais

Recebi auxílio do colega Matheus Moraes Machado

1) Conversão imagem binária:



Para a identificação dos componentes, foram utilizadas as funções: “threshold”, “findContours”, “contourArea” e “arcLength” do "opencv". Foi exibido o número de componentes identificados, a área total e a área de cada componente. Coloquei um limiar de área mínima no valor de 1000, ou seja, só é exibido o componente que possui área maior que 1000 (os componentes principais). Exemplo a seguir:

```
Total area: 317200
Number of black components: 341

Componente 148:
Area: 15361.0
Perimetro: 527.6711337566376

Componente 151:
Area: 15904.0
Perimetro: 538.8427066802979

Componente 297:
Area: 15127.0
Perimetro: 519.4701231718063

Componente 328:
Area: 27692.0
Perimetro: 809.8965302705765

Componente 338:
Area: 17566.5
Perimetro: 593.5462441444397

Componente 339:
Area: 10779.0
Perimetro: 510.6416963338852
```

Referências:
https://docs.opencv.org/master/d4/d73/tutorial_py_contours_begin.html

2)

A resolução dessa questão foi realizada com o auxílio da biblioteca “skimage.feature”. A função “greycomatrix” calcula a matriz de co-ocorrência e a “greycoprops” retorna as propriedades dessa matriz, como: homogeneidade e uniformidade(segundo momento angular).Com esse auxílio foram obtidos os seguintes resultados (as fotos se encontram na pasta com os arquivos):

Foto Indoor (com pouca interferência luminosa):

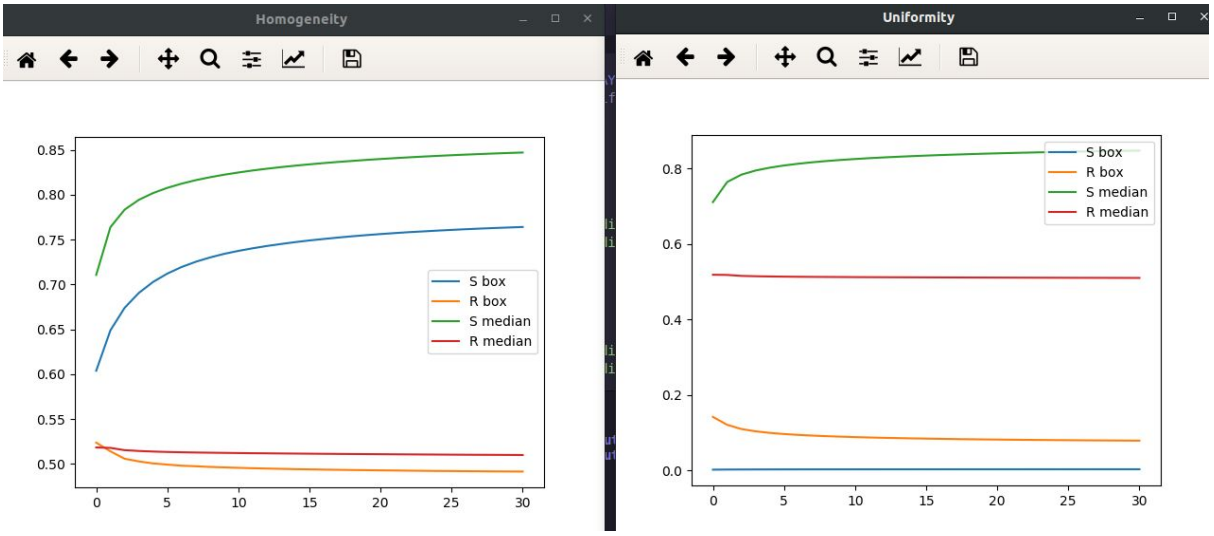
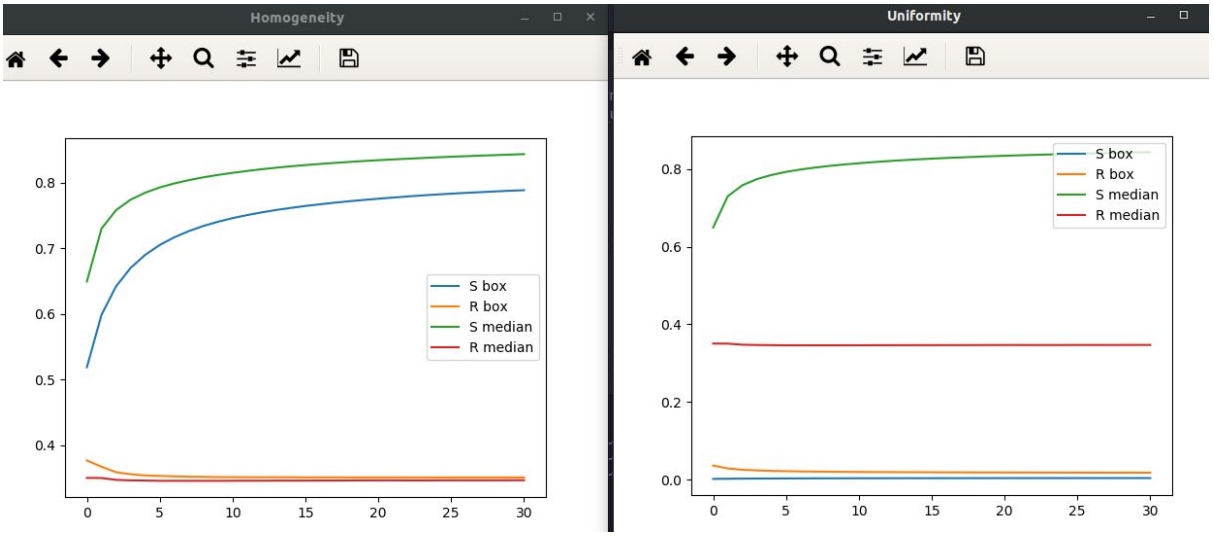


Foto Outdoor (com pouca interferência luminosa):



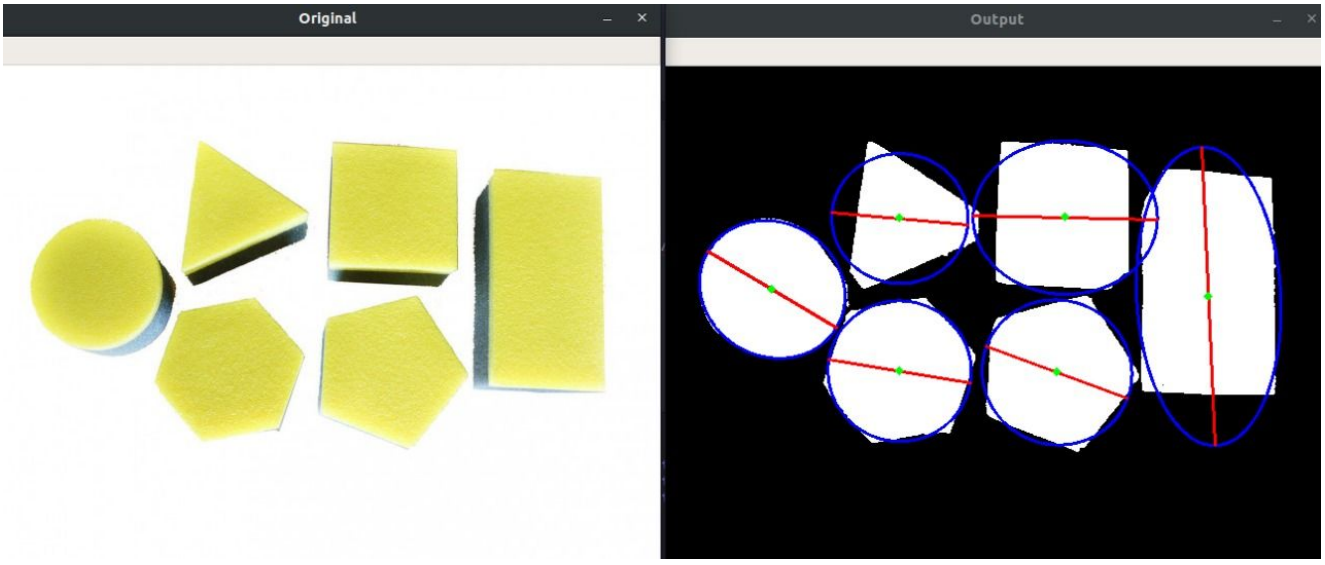
Referências:

http://www.ajnr.org/content/ajnr/suppl/2015/09/10/ajnr.A4455.DC1/15-00131.pdf

https://scikit-image.org/docs/0.7.0/api/skimage.feature.texture.html

https://stackoverflow.com/questions/37761411/how-do-i-mention-the-direction-of-neighbours-to-calculate-the-glcm-in-skimage-py

3) Para auxiliar na resolução desse exercício foram utilizadas as seguintes funções do “opencv”: “connectedComponentsWithStats”, “findContours”, “fitEllipse” e “ellipse”.



Referências:

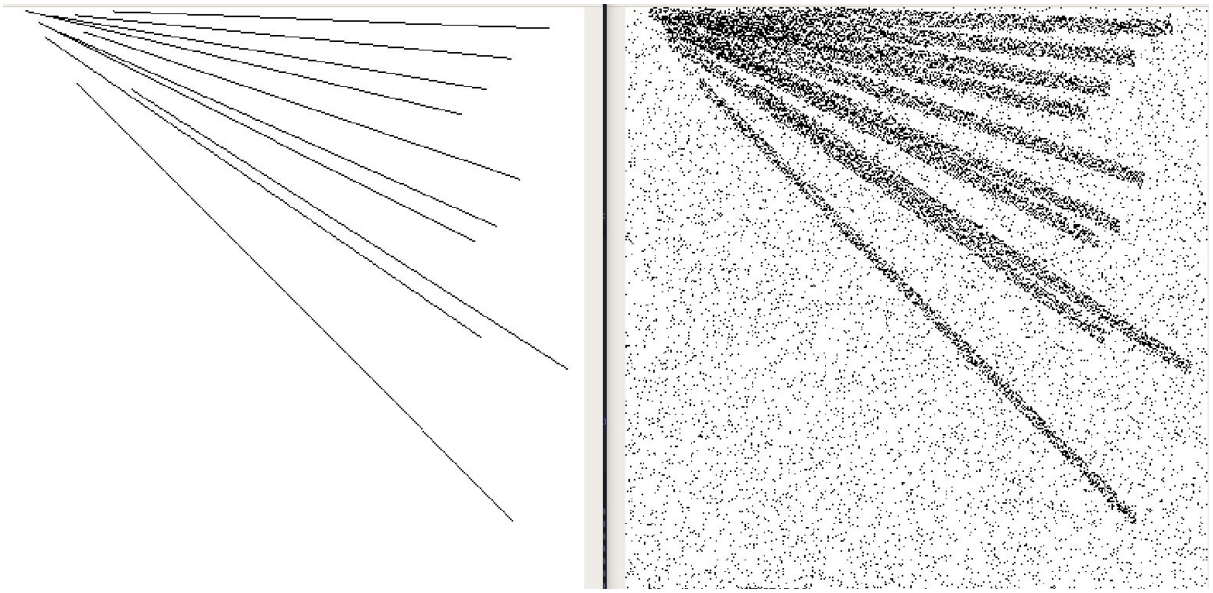
https://docs.opencv.org/2.4/doc/tutorials/imgproc/shapedescriptors/bounding_rotated_ellipses/bounding_rotated_ellipses.html

https://docs.opencv.org/3.0-beta/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=connectedcomponents

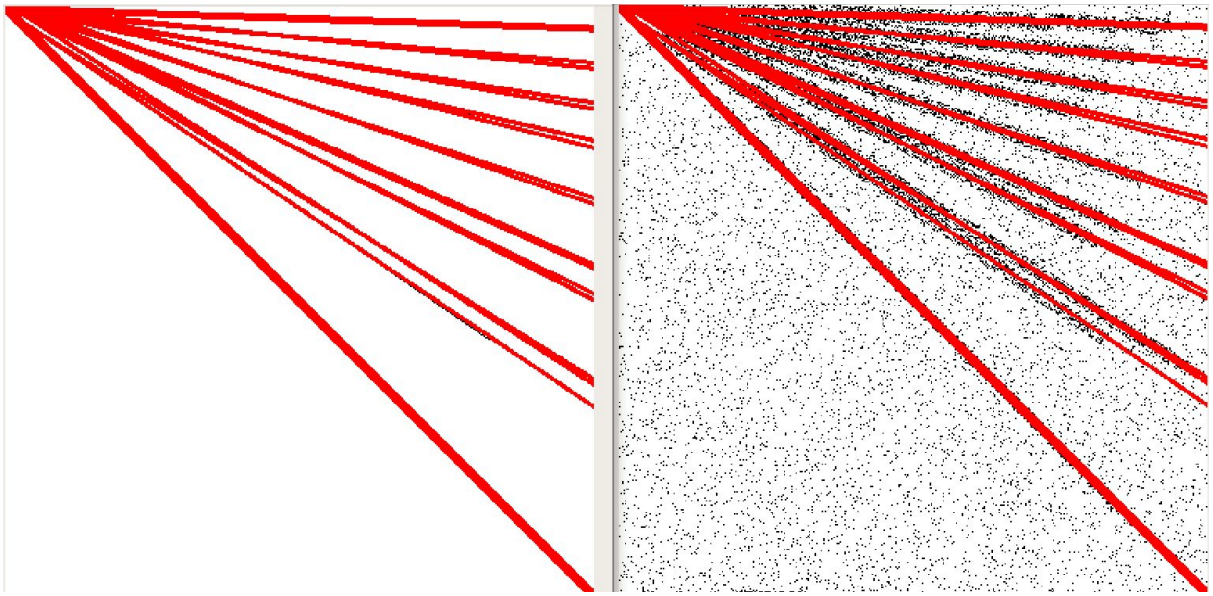
https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html

4) O algoritmo de geração de linhas foi baseado no método “HoughLines”. E o mesmo método foi utilizado para a detecção das linhas, com o auxílio da função HoughLines” do “opencv”. A geração de linhas recebe como parâmetro o número de linhas a ser gerado e a magnitude do ruído a ser inserido. Exemplo a seguir:

Linhas verdadeiras e com ruído geradas:



Detecção:



Referências:
https://docs.opencv.org/3.4.0/d6/d10/tutorial_py_houghlines.html

https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html