MC920 - Trabalho 1

Nome: Bruno Orsi Berton
RA: 150573

Versões usadas

Python - 3.5.2
NumPy - numpy-1.14.1
Matplotlib - latest
Scikit-image - 0.13.1
Scipy - 1.0.0

Como executar

Para executar o programa, basta executar o comando:

python3 trabalho1.py <caminho relativo para a imagem>

Exemplo:

Caso a imagem esteja na mesma pasta do programa:
python3 trabalho1.py objetos1.png

Ou caso a imagem esteja em outra pasta
python3 trabalho1.py ../images/objetos1.png

Saída do programa

O programa final gerará 4 imagens:

- Uma com a imagem colorida convertida para níveis de cinza “**gray-image.png**”
- Uma com os contornos de todos os objetos encontrados na imagem “**contours.png**”
- Uma com os labels de cada objeto identificado pelo programa de 0 a N “**labeled-image.png**”
- E a última com o histograma de objetos pequenos, médios e grandes “**histogram.png**”

E o programa exibirá as estatísticas necessárias no próprio console também, exemplo:

número de regiões: N

região: 0 perímetro: 190.00 área: 2352
região: 1 perímetro: 190.00 área: 2352
região: 2 perímetro: 62.00 área: 272
região: 3 perímetro: 62.00 área: 272
...

número de regiões pequenas: X
número de regiões médias: Y
número de regiões grandes: Z

Análise do programa

Para a conversão da imagem colorida em níveis de cinza foi utilizado uma transformação linear na qual é atribuído pesos a cada banda de cor, e depois é calculado a média ponderada, e o resultado da média é o nível de cinza para o pixel.

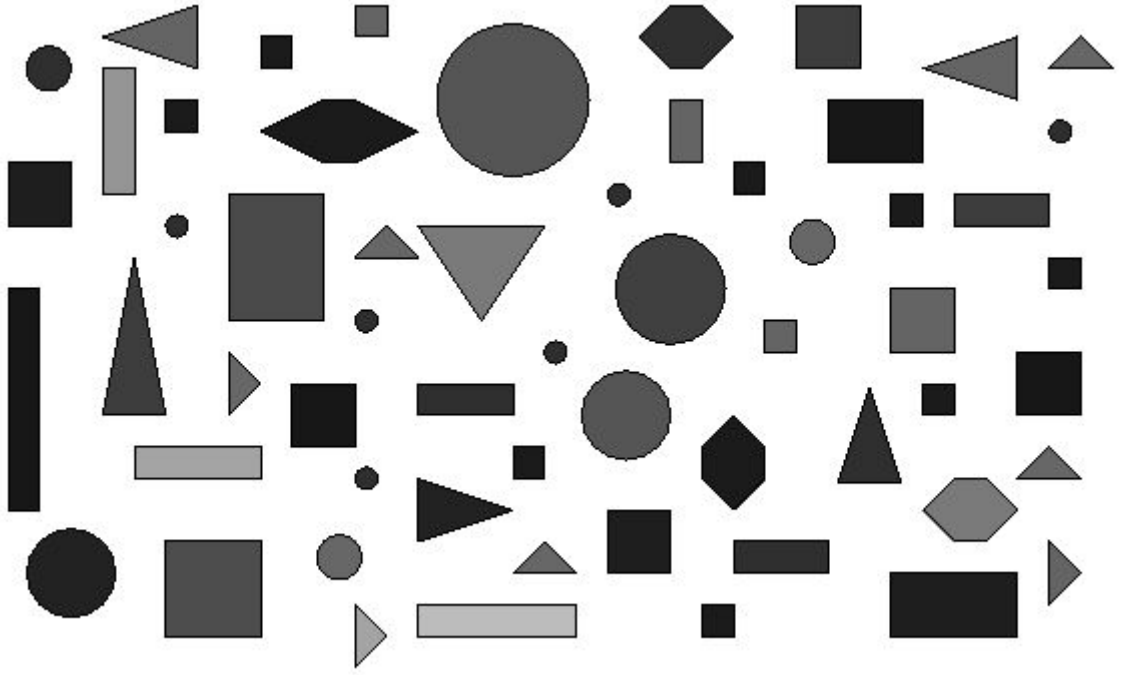
A seguinte fórmula foi usada para a transformação linear:

$$Yi = 0.299 * Ri + 0.587 * Gi + 0.114 * Bi$$

Onde Ri, Gi e Bi são os níveis de vermelho, verde e azul do pixel i, e Yi é o nível de cinza atribuído ao pixel i, e Yi é arredondado.

Dessa forma, é garantido uma boa nitidez a imagem após a conversão para a escala de cinza, é claro que outros pesos poderiam ser usados, mas estes pesos são os que geralmente são usados nos pacotes disponíveis.

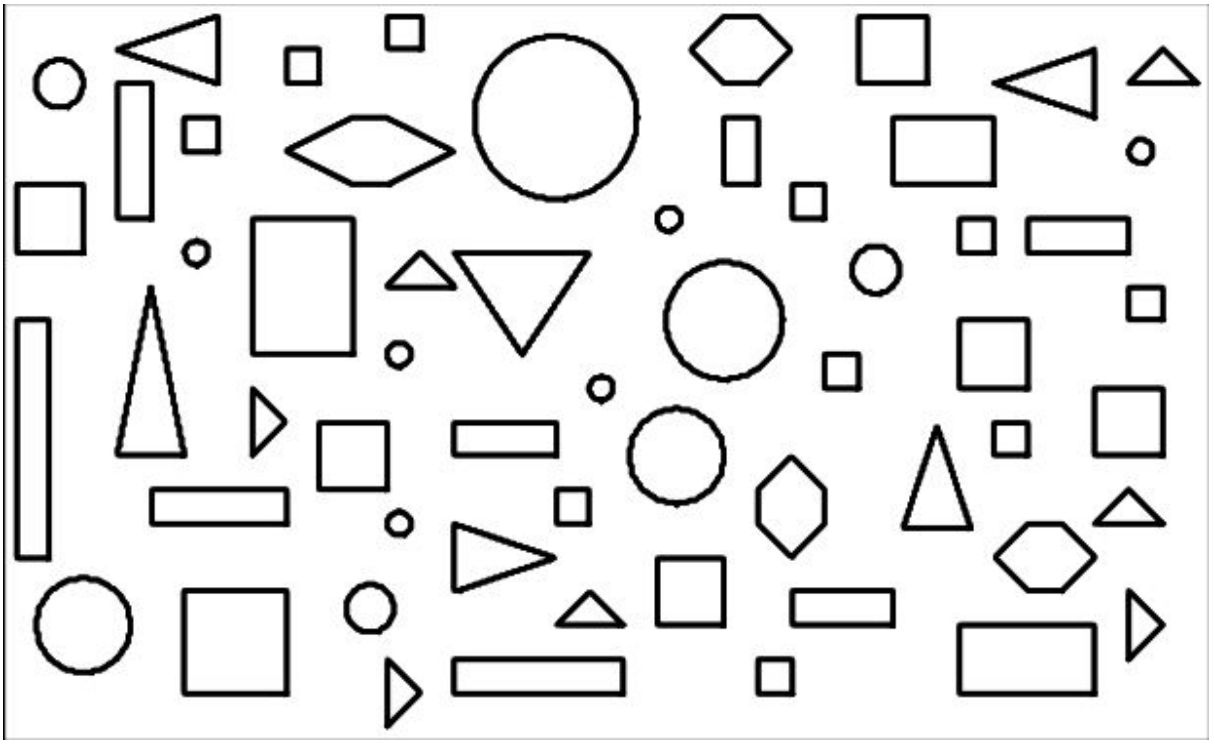
Segue o resultado da imagem objetos2.png:



Para encontrar os contornos, foi utilizado o modo **measure** do pacote **skimage**, que fornece várias funções prontas para se trabalhar com objetos em uma imagem. A extração dos contornos pode ser feita através da função **find_contours**, o que nos retorna uma matriz em que cada linha representa um contorno encontrado, contendo todas as coordenadas dos pixels que representam o contorno.

Depois de termos todos os contornos, os pontos dos contornos são plotados em um fundo branco para que possamos ver os contornos.

Segue o resultado dos contornos da imagem objetos2.png:



Como podemos ver, o método é bem satisfatório quando há uma divisão muito clara em fundo e objetos.

Para a geração dos labels e das estatísticas, foi feito uma alteração na imagem cinza, na qual todos os pixels que estiverem com a intensidade menor que 255, será atribuído a intensidade 0, dessa maneira, as funções do módulo de **measure** parecem funcionar como esperado, pois sem isso, a borda dos objetos não é considerada nas estatísticas.

Foi utilizado a função **label**, juntamente com a função **regionprops**, do mesmo módulo de **measure**, para a geração das estatísticas. A função **regionprops** já calcula tudo o que precisamos, área dos objetos, centróides, perímetro e entre muitas outras medidas.

Segue algumas das estatísticas da imagem objetos2.png:

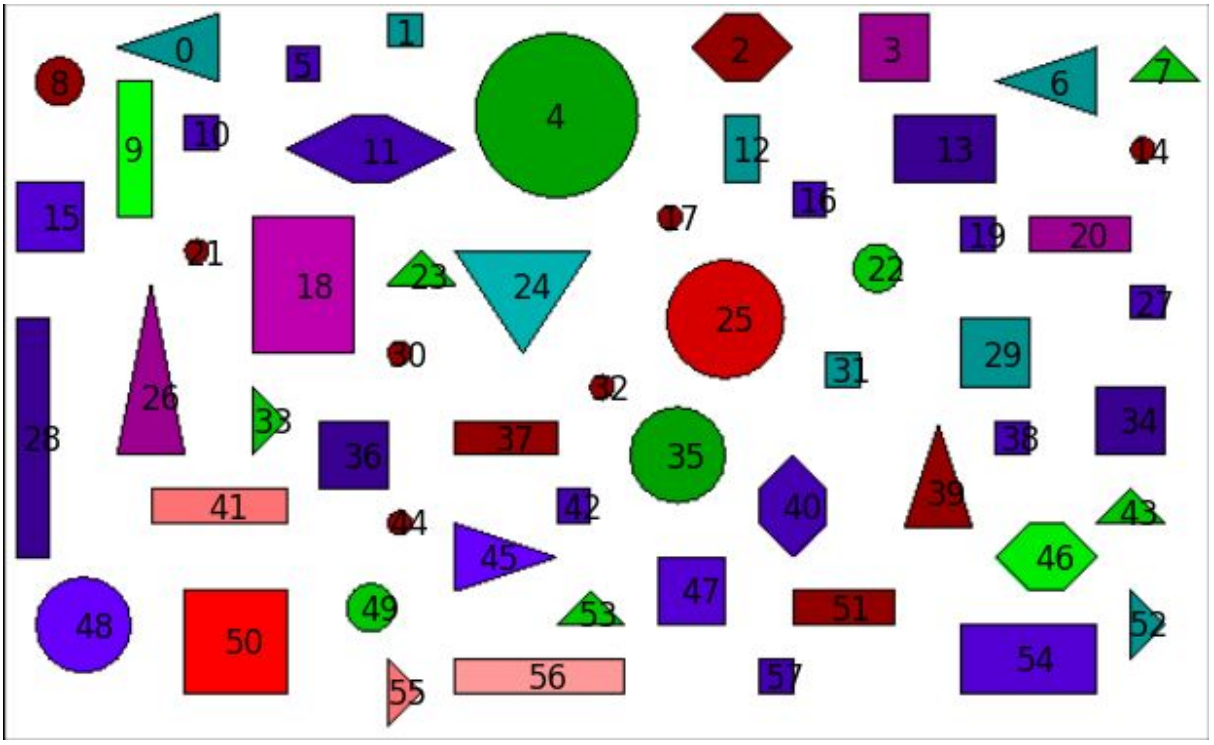
número de regiões: 58

região: 0 perímetro: 138.01 área: 816
região: 1 perímetro: 62.00 área: 272
...

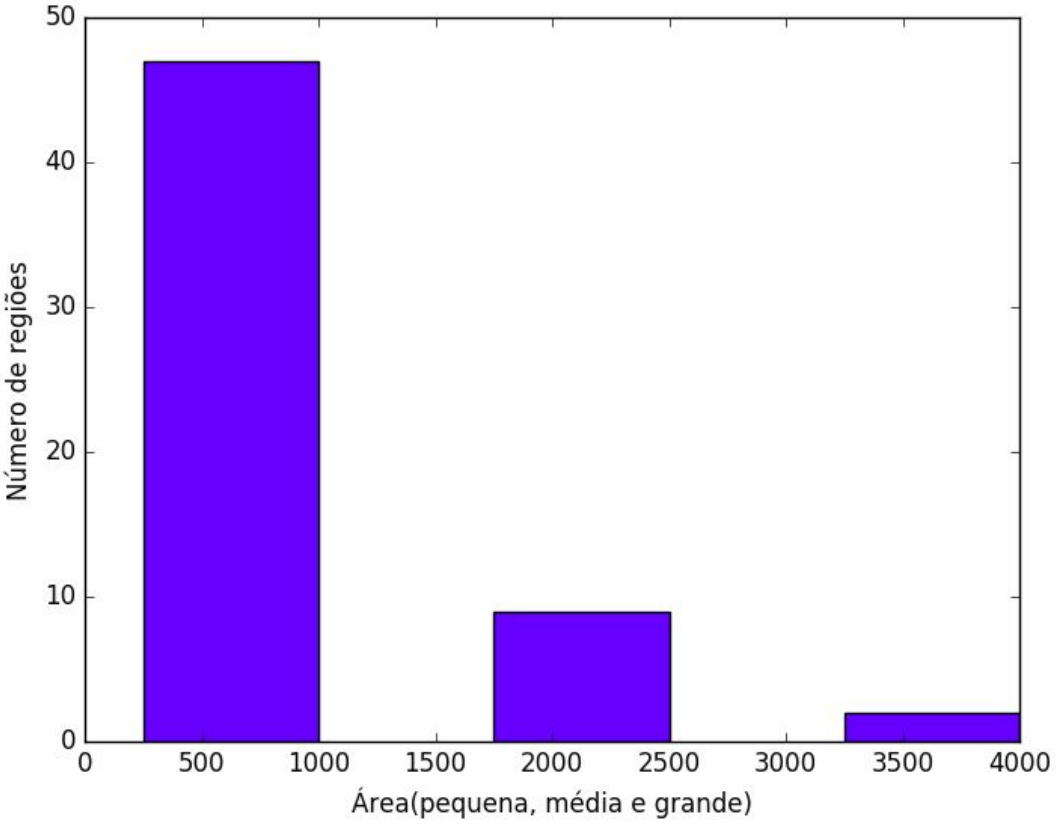
região: 56 perímetro: 190.00 área: 1360
região: 57 perímetro: 64.00 área: 289

número de regiões pequenas: 47
número de regiões médias: 9
número de regiões grandes: 2

E a imagem com os labels:



Depois de calculado quantas imagens são pequenas, médias e grandes, foi plotado o histograma, segue a imagem:



Limitações

Para esse trabalho, é esperado que a imagem passada ao programa tenha o fundo perfeitamente branco e que a imagem passada seja colorida, com cada banda de cor com 8 bits.