Trabalho 3 - Extração e Caracterização de Objetos

MC920 - Introdução ao Processamento de Imagem Digital

Victor Ferreira Ferrari
RA 187890
vferrari@mpc.com.br

17 de Outubro de 2019

1 Introdução

Uma importante parte do processamento de imagens é a detecção e extração de objetos em uma imagem. Pode-se extrair diversas informações de uma imagem a partir da segmentação de objetos. A depender da entrada, pode-se interpretar diversas dessas propriedades (análise) ou apenas utilizar essa divisão para gerar novas imagens processadas.

Para processamento de imagens, diversas bibliotecas foram criadas com funções prontas implementando diferentes operações importantes, desde limiarização (global ou local) e operações morfológicas até funções mais complexas.

O intuito do projeto é extrair propriedades de objetos em imagens, explorando para isso bibliotecas com funções auxiliares. Para análise da imagem, um histograma diferenciando objetos pequenos, médios e grandes será feito também.

2 Características do Programa

O programa foi feito na linguagem *Python*, explorando as bibliotecas externas OpenCV (CV2) e Scikit-Image, com auxílio das bibliotecas externas NumPy e MatPlotLib. Os argumentos são passados na execução via `argv`, consistindo na imagem de entrada e opcionalmente a pasta de saída. A entrada deve ser uma imagem colorida. Serão usadas imagens que possuem objetos coloridos em um fundo branco.

3 Implementação

A leitura da imagem colorida é feita via `OpenCV`, pela função `imread`. Com a imagem lida, pode-se começar o processamento dividindo a imagem em regiões (objetos). Para isso, o primeiro passo é transformar a imagem em escala de cinza, por meio da função `cvtColor`.

Feito isso, para separar a imagem entre objetos e fundo, foi feita uma limiarização global da imagem utilizando a função `threshold`. A limiarização feita foi inversa, ou seja, todos os pixels com intensidade acima do limiar foram transformados em **preto** e os outros foram transformados em **branco**. Como o fundo da imagem é branco, o limiar utilizado foi 254, ou seja, apenas os pixels brancos estão acima do limiar.

Com a imagem binária, a identificação e segmentação dos objetos pode ser feita de duas maneiras, diferenciando a biblioteca utilizada. Com o OpenCV, pode-se encontrar os contornos dos objetos utilizando a função `findContours`, e os contornos resultantes foram apresentados em uma imagem branca pela função `drawContours`. Com o `Scikit-Image`, pode-se dividir a

imagem em regiões utilizando a função `label`. Como esta biblioteca possui formato diferente, a função `img_as_float` foi usada para conversão.

A partir dos contornos, é possível encontrar o centróide de cada objeto. Com o objetivo de alterar a imagem para atribuir um número a cada objeto, o centróide de cada objeto foi calculado a partir dos momentos, com a função `moments`. Temos que:

$$x_c = \frac{m_{10}}{m_{00}}$$

$$y_c = \frac{m_{01}}{m_{00}}$$

Nesse ponto, foi inserido na imagem o número correspondente àquela região utilizando a função `putText`. Foi ainda aplicada uma correção baseada no tamanho do texto a ser inserido (`getTextSize`), pois a função utiliza o ponto passado como canto inferior esquerdo do texto. Com a correção, o número passa a ser centralizado no centróide.

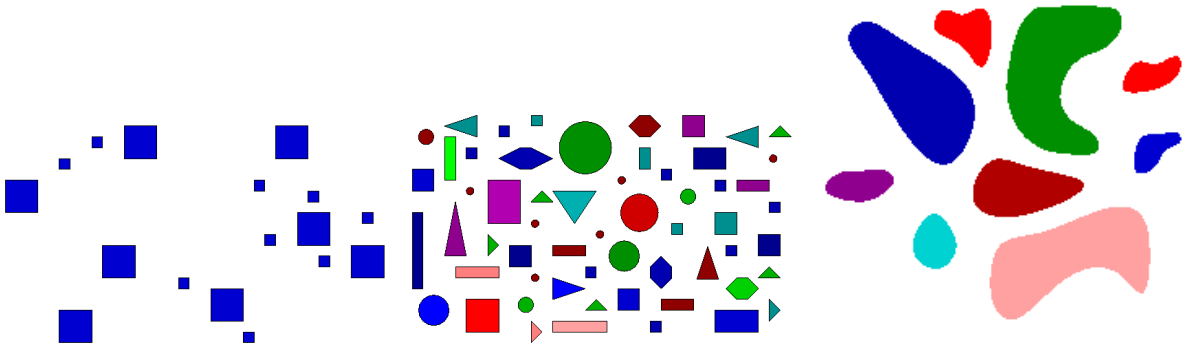
A partir das regiões rotuladas pelo Scikit-Image, diversas propriedades podem ser obtidas pela função `regionprops`, que gera uma lista de regiões com diversos atributos. Desses atributos, são impressos na saída padrão a área, o perímetro, a excentricidade, a solidez e as coordenadas do centróide, todos disponíveis diretamente pela lista de regiões.

Enfim, a partir da área de cada região foram feitas subdivisões em áreas pequenas (menos de 1500 pixels), médias (entre 1500 e 3000 pixels) e grandes (mais de 3000 pixels). O resultado é impresso na saída padrão e, com auxílio da biblioteca MatPlotLib, é construído um histograma com essas informações.

A imagem contendo os contornos, a imagem rotulada e o histograma são salvos na pasta "Outputs/"(caso outra pasta não tenha sido passada como segundo argumento na chamada).

4 Resultados e Comparação

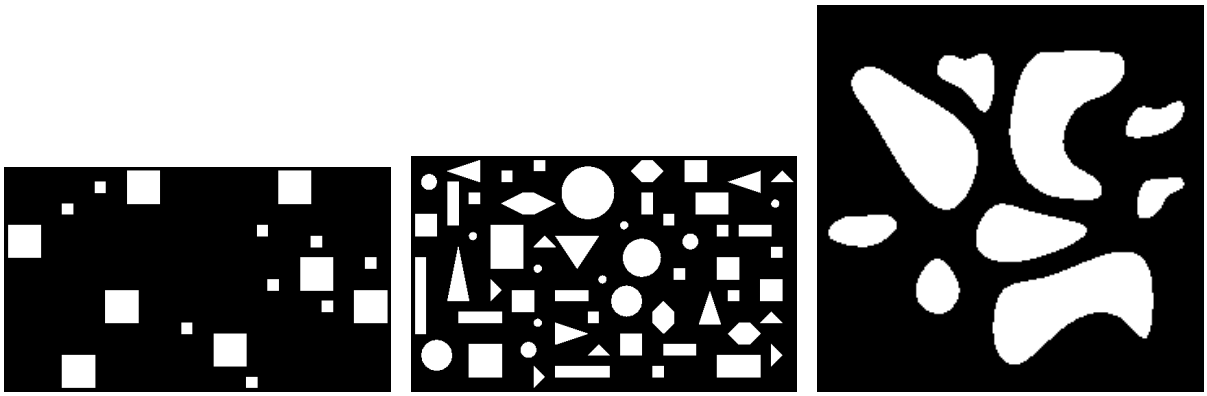
Foram testadas 3 imagens de objetos coloridos dispostos em um fundo branco, em formato PNG. Todas as imagens de entrada então disponíveis em `https://www.ic.unicamp.br/~helio/imagens_objetos_coloridos/`. As imagens utilizadas se encontram na figura 1.



(a) objetos1.png (b) objetos2.png (c) objetos3.png

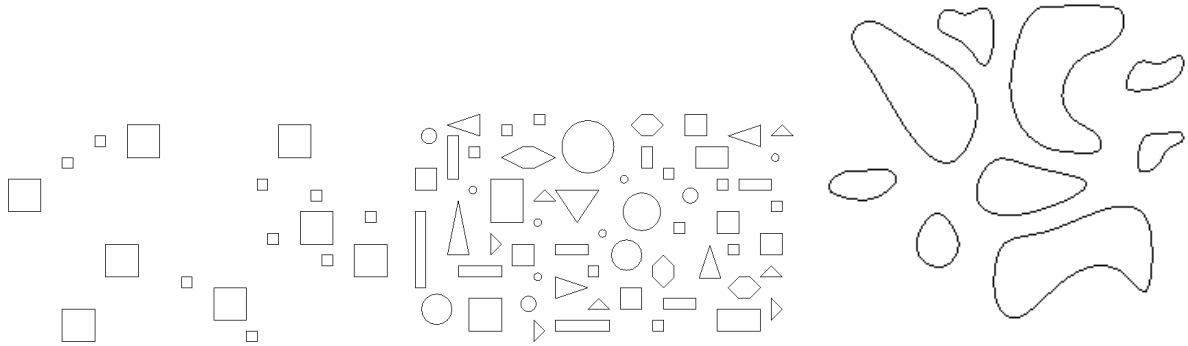
Figura 1: Imagens de entrada testadas.

Como pode ser visto, as imagens possuem objetos coloridos em um fundo branco, em diferentes tamanhos e formatos. Para encontrar as regiões, primeiro foi necessário transformar a imagem em binária, com objetos brancos em um fundo preto. As imagens binárias podem ser vistas na figura 2.



(a) objetos1.png (b) objetos2.png (c) objetos3.png

Figura 2: Imagens binárias com fundo preto e objetos brancos, a partir das imagens de entrada.

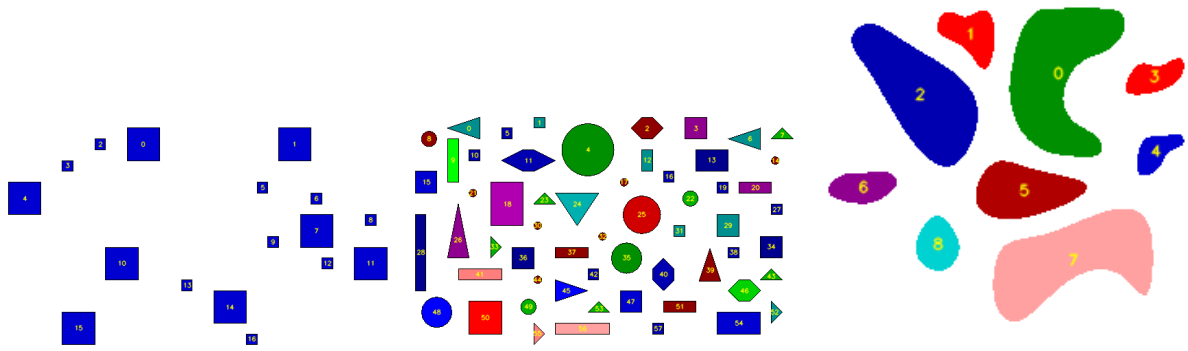


(a) objetos1.png (b) objetos2.png (c) objetos3.png

Figura 3: Contornos dos objetos nas imagens de entrada.

Após o processamento inicial, os contornos retornados foram desenhados e podem ser vistos na figura 3.

Então, tomando cada um dos objetos como uma região, as propriedades foram tomadas. Para referência, as regiões com seus identificadores (posicionados nos centroides) para cada imagem de entrada podem ser vistas na figura 4.



(a) objetos1.png (b) objetos2.png (c) objetos3.png

Figura 4: Imagens de entrada com regiões numeradas.

Dividindo a imagem em regiões, as propriedades de área, perímetro, excentricidade e solidez

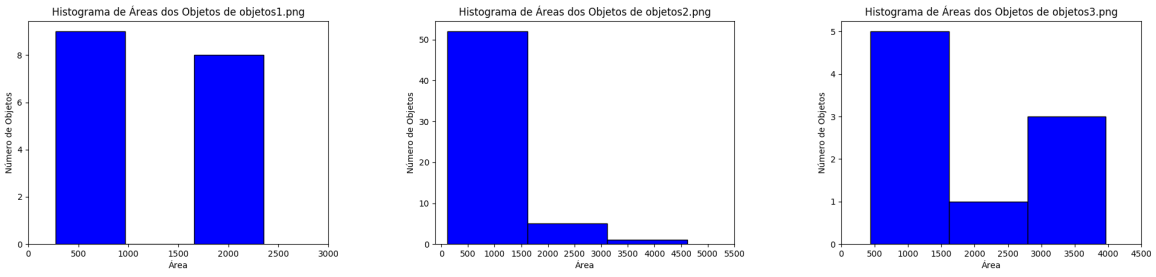
foram extraídas, assim como as coordenadas do centróide. Como exemplo, a seguir há a saída da imagem `objetos2`, que possui 9 objetos/regiões:

- região 0: área: 3969 perímetro: 313.764502 excentricidade: 0.816362 solidez: 0.747739 centróide: (69, 148)
- região 1: área: 791 perímetro: 119.982756 excentricidade: 0.741103 solidez: 0.898864 centróide: (44, 94)
- região 2: área: 3584 perímetro: 259.462987 excentricidade: 0.898073 solidez: 0.977899 centróide: (81, 63)
- região 3: área: 540 perímetro: 99.254834 excentricidade: 0.889586 solidez: 0.910624 centróide: (70, 207)
- região 4: área: 438 perímetro: 88.769553 excentricidade: 0.855923 solidez: 0.916318 centróide: (116, 208)
- região 5: área: 1684 perímetro: 174.124892 excentricidade: 0.868169 solidez: 0.972286 centróide: (140, 127)
- região 6: área: 642 perímetro: 103.012193 excentricidade: 0.890242 solidez: 0.969789 centróide: (138, 28)
- região 7: área: 3934 perímetro: 305.421356 excentricidade: 0.910992 solidez: 0.774257 centróide: (183, 157)
- região 8: área: 675 perímetro: 96.325902 excentricidade: 0.620380 solidez: 0.976845 centróide: (173, 74)

Por fim, as regiões das imagens foram divididas por tamanho, entre pequenas, médias e grandes. A divisão encontrada pode ser vista a seguir:

- Pequenas: Objetos1 - 9, Objetos2 - 47, Objetos3 - 5;
- Médias: Objetos1 - 8, Objetos2 - 9, Objetos3 - 1;
- Grandes: Objetos1 - 0, Objetos2 - 2, Objetos3 - 3;

Os histogramas resultantes dessas divisões podem ser encontrados na figura 5.



(a) objetos1.png (b) objetos2.png (c) objetos3.png

Figura 5: Histogramas com áreas de regiões na imagem.

Pode-se perceber que todas as imagens de saída produzidas estão condizentes com as de entrada, e que a divisão e os histogramas feitos também estão coerentes, mesmo não sendo possível verificar o valor exato das áreas de outro modo.

5 Conclusão

Dados os resultados corretos em poucos passos, vemos que as bibliotecas utilizadas `OpenCV` e `Scikit-Image` são poderosas alternativas para processamento de imagens, realizando operações complexas de segmentação com apenas simples chamadas.

Claro que, como as imagens utilizadas têm uma disposição simples e fácil para segmentação de regiões, os resultados são melhores e menos pré-processamento é necessário. O processo para imagens quaisquer seria mais complexo.

Ainda assim, o resultado demonstra o poder das bibliotecas para processamento de imagens e a quantidade de informações que podem ser extraídas a partir de objetos em uma imagem.