# Trabalho 3

Tiago Loureiro Chaves (187690)

*MC920A - Introdução ao Processamento de Imagem Digital - 2s2019*

**Resumo**

Este projeto teve como objetivo a aplicação de algumas transformações e obtenção de diferentes medidas de objetos/regiões em imagens (p.e. segmentação de bordas, perímetro de regiões, centróide dos objetos, etc.).

Para isso, as imagens coloridas são inicialmente convertidas para preto-e-branco, e então extrai-se os contornos (representados como uma lista de pixels) dos objetos identificados na imagem com a função `findCountours()` do OpenCV, que implementa o algoritmo [1]. A partir dos contornos obtidos então podemos calcular diferentes momentos da imagem, e com eles obter as medidas apresentadas na Seção 3.

# 1 Problema

Os momentos de uma imagem digital [2] são medidas que possuem interessantes propriedades, utilizadas na interpretação e descrição de imagens, principalmente nos processos de segmentação e reconhecimento de padrões.

Considerando uma imagem $f(x, y)$ de dimensão $M \times N$, define(m)-se o(s) momento(s) de ordem $(p + q)$ como:

$$M_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q f(x, y) \tag{1}$$

Estamos particularmente interessados nos momentos de ordem zero ($M_{00}$), que representa a área da região; nos de ordem 1 ($M_{10}$ e $M_{01}$), com os quais podemos calcular o centroide da região; e os de ordem 2, para calcularmos a excentricidade [3].

Além disso, também utilizamos o conceito de fecho convexo [4] para calcular a solidez de um objeto — definida pela razão entre a área do objeto e a área de seu fecho convexo.

# 2 Programa

## 2.1 Dependências e Organização

O trabalho foi desenvolvido em Python `3.7.4`, com as bibliotecas Numpy `1.17.0`, Matplotlib `3.1.1` e OpenCV `4.1.0`, e é dividido em três arquivos:

- `main.py` – tratamento da entrada/saída e execução do projeto
- `transformations.py` – implementação das transformações e medições de imagens
- `utils.py` – funções gerais para processamento de imagens

As imagens utilizadas para teste estão na pasta `i/`, e exemplos de saída encontram-se na pasta `o/`. O código também pode ser visto em github.com/laurelkeys/image-processing.

## 2.2 Execução

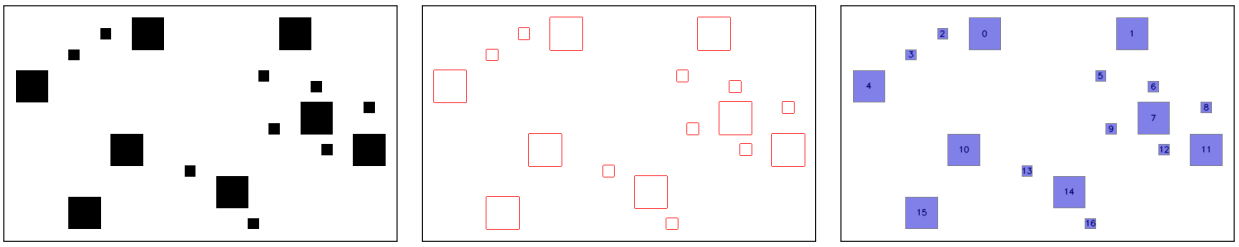
As imagens de entrada `.png` devem ser colocadas na pasta `i/`, localizada no mesmo diretório do script `main.py`, que salvará as imagens resultantes na pasta `o/`.

Por padrão, aplicam-se as funções de transformação e obtenção de medidas de regiões a todas as imagens que estão na pasta `i/`. Entretanto, é possível especificar uma imagem única para ser utilizada executando-se `main.py -img IMAGE`.

Além disso, caso seja desejável exibir as imagens processadas (além de salvá-las), a opção `--display_images` pode ser passada.

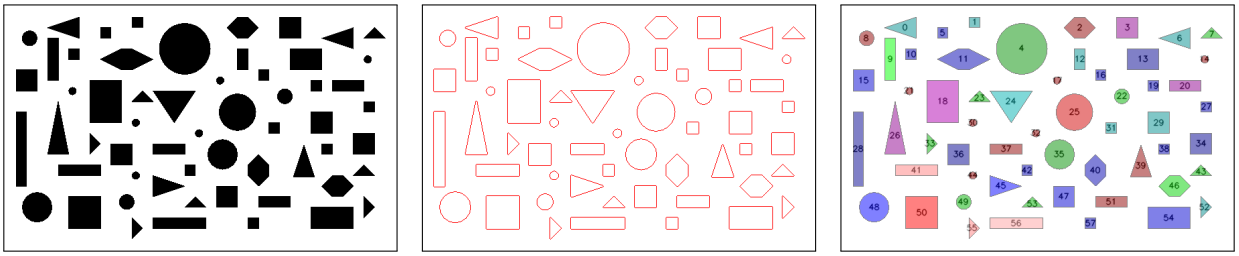
Pode-se visualizar todas as opções disponíveis com `main.py --help`.

# 3 Resultados



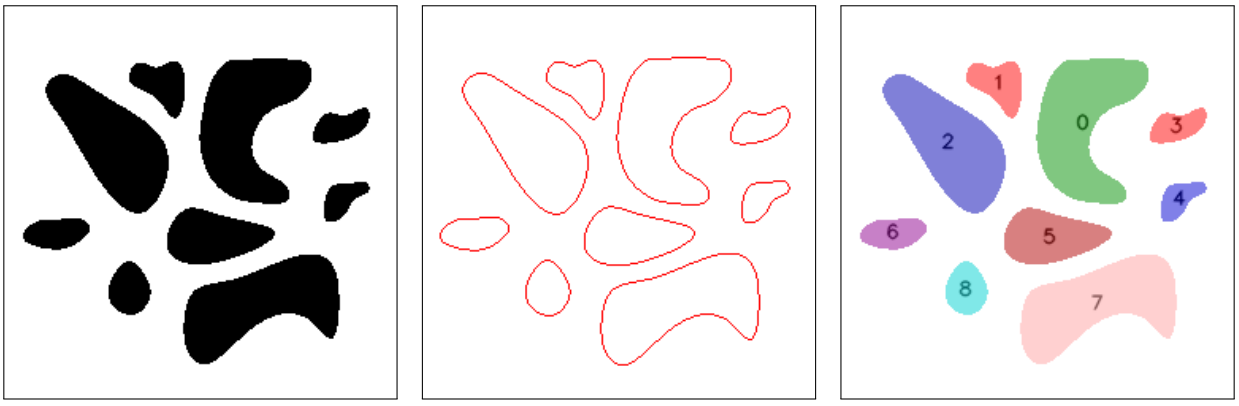
(a) Imagem binária (b) Contornos dos objetos (c) Regiões rotuladas

Figura 1: Resultados para a figura `objetos1.png`



(a) Imagem binária (b) Contornos dos objetos (c) Regiões rotuladas

Figura 2: Resultados para a figura `objetos2.png`




(a) Imagem binária (b) Contornos dos objetos (c) Regiões rotuladas

Figura 3: Resultados para a figura `objetos3.png`

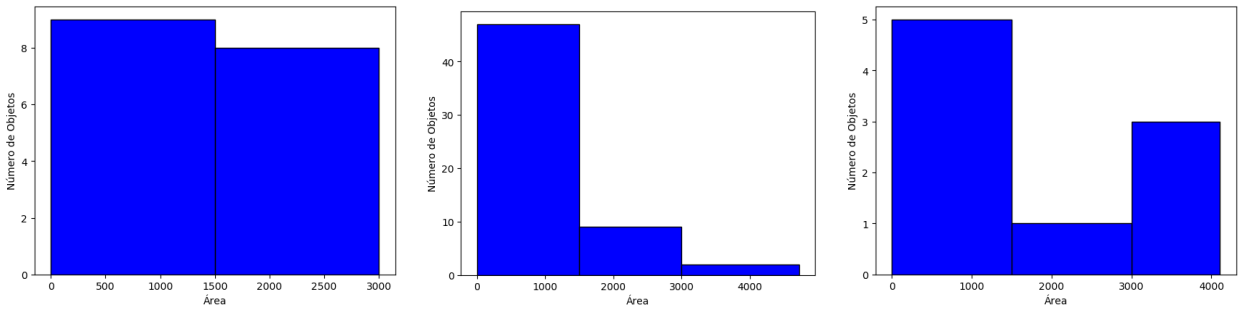
```
numero de regioes: 9
regiao 0:  area: 4107  perimetro: 319.421354  excentricidade: 0.813036  solidez: 0.754963
regiao 1:  area:  844  perimetro: 125.639609  excentricidade: 0.735146  solidez: 0.904558
regiao 2:  area: 3690  perimetro: 265.119838  excentricidade: 0.896234  solidez: 0.978264
regiao 3:  area:  584  perimetro: 104.911687  excentricidade: 0.882918  solidez: 0.913928
regiao 4:  area:  478  perimetro:  94.426406  excentricidade: 0.850570  solidez: 0.925460
regiao 5:  area: 1762  perimetro: 179.781745  excentricidade: 0.864278  solidez: 0.971862
regiao 6:  area:  688  perimetro: 108.669047  excentricidade: 0.882704  solidez: 0.972458
regiao 7:  area: 4067  perimetro: 311.078208  excentricidade: 0.908225  solidez: 0.780689
regiao 8:  area:  716  perimetro: 101.982755  excentricidade: 0.610530  solidez: 0.980164
```

Listagem 1: Propriedades dos objetos da Figura 3c

Calculando as áreas das regiões definidas pelos objetos identificados em cada imagem, podemos classificá-las segundo os seguintes critérios:

- objeto pequeno: área $< 1500$ pixels
- objeto médio: área $\geq 1500$ pixels e área $< 3000$ pixels
- objeto grande: área $\geq 3000$

Assim, obtemos os histogramas de áreas apresentados na Figura 4.




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

Figura 4: Histogramas de áreas dos objetos

| | `objetos1.png` | `objetos2.png` | `objetos3.png` |
|---|---|---|---|
| pequenos | 9 | 47 | 5 |
| médios | 8 | 9 | 1 |
| grandes | 0 | 2 | 3 |

Tabela 1: Número de objetos por grupo de área

# 4 Discussão

Analisando a Tabela 1 e os histogramas de áreas (Figura 4) com as Figuras 1, 2 e 3 vemos que os resultados são coerentes (compare, por exemplo, as Figuras 3c e 4c com o exemplo das propriedades medidas para a imagem `objetos3.png` mostrado na Listagem 1).

Utilizando funções do OpenCV [1] podemos fácil e rapidamente obter diversas medidas de objetos que foram segmentados em imagens (com a simples técnica de *thresholding* global [6], por exemplo), as quais nos possibilitam diversas aplicações, como: uso em tarefas de classificação, aplicação das medidas obtidas (veja Listagem 1) como *input* de redes neurais,

[1] `findContours`, `moments`, `arcLength` e `convexHull` (todas bem descritas em [5])

reconhecimento de padrões (p.e. distinguir os diferentes objetos presentes em cada imagem), entre diversos outros.

Assim, o conhecimento e a familiarização com tais funcionalidades disponíveis é uma interessante adição ao ferramental que podemos empregar em futuras aplicações.

# 5 Conclusões

Pudemos explorar, com esse trabalho, diferentes informações que podem ser obtidas a partir do cálculo dos momentos de imagens [2], assim como conhecer funções da biblioteca OpenCV que podem ter usos relevantes em trabalhos futuros (principalmente para a extração de *features* de imagens segmentadas).

É interessante ressaltar que as medidas feitas pelo OpenCV trabalham em cima de uma versão binarizada (i.e. em preto-e-branco) da imagem original, o que leva ao problema de gerar-se uma boa separação entre o que é objeto de interesse e o que deve ser considerado como "fundo" nas imagens utilizadas.

No caso, trabalhou-se com imagens bem comportadas (*"toy examples"*), uma vez que a tarefa de interesse era a obtenção de medidas a partir das figuras, e não seu pré-processamento, mas o problema de segmentação é um de grande importância, além de não ser trivial, e foi explorado em mais detalhes no trabalho anterior [6].

# Referências


[1] S. Suzuki and K. be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0734189X85900167 1

[2] "Image moment — Wikipedia, The Free Encyclopedia," 2019. [Online]. Available: https://en.wikipedia.org/wiki/Image_moment 1, 4

[3] Hélio Pedrini, "Representação e Descrição." [Online]. Available: http://www.ic.unicamp.br/~helio/disciplinas/MC920/aula_representacao.pdf 1

[4] "Convex hull — Wikipedia, The Free Encyclopedia," 2019. [Online]. Available: https://en.wikipedia.org/wiki/Convex_hull 1

[5] "Contours in OpenCV," 2019. [Online]. Available: https://docs.opencv.org/4.1.0/d3/d05/tutorial_py_table_of_contents_contours.html 3

[6] T. L. Chaves, "Trabalho 2," 2019. [Online]. Available: https://github.com/laurelkeys/image-processing/blob/master/reports/MC920___Trabalho_2.pdf 3, 4


Links acessados em outubro de 2019.