MC920 - Relatorio Trabalho 1

Daniel Helu Prestes de Oliveira RA:166215

Março 2018

1 Introdução

Neste trabalho, implementamos um programa que analisa imagens no formato PNG e retorna informações relacionadas aos contornos da objetos que aparecem na imagem. As imagens utilizadas foram restringidas a imagens que possuem figuras geométricas coloridas distribuidas sobre um fundo branco.

2 Observações do Código

O programa foi desenvolvido em Python 3 utilizando o Jupyter Notebook. Para executar o programa, basta chama-lo por linha de comando no terminal e passar como argumento o nome da imagem sobre qual o programa executará as transformações. O programa retornará prints de informações das figuras, como por exemplo, o número de regiões e um histrograma sobre area das regioes. Além disso, o programa salva as imagens resultantes de cada etapa do processo na pasta onde foi executado com o nome da imagem mais uma flag de indentificação da etapa. Exemplo de chamada: python trabalho1.py imagefile

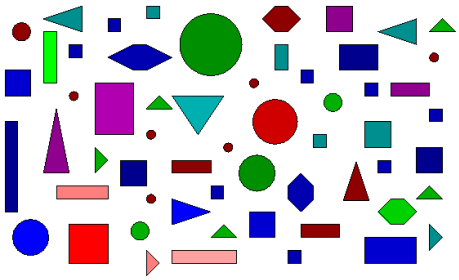
3 Desenvolvimento

3.1 Transformação de Cores

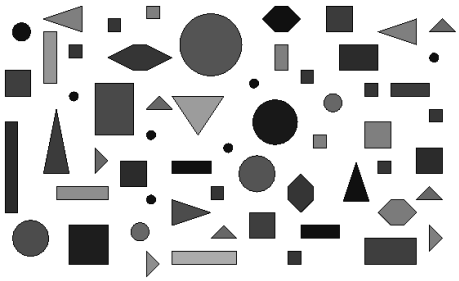
Nesse primeiro exercicio, nosso programa recebe uma imagem fonte no formato png com o padrão de cores RGB e converte para o padrão de cores escala cinza, como mostrado na Figure 1. O algoritimo utilizado foi o da função cvtColor, que recebe a imgfonte e a flag que define a transformação de cores como parametros, da biblioteca OpenCV.

3.2 Contorno dos Objetos

Nesse exercicio, o programa utiliza da função threshold da biblioteca OpenCV para identificar as bordas da figuras geométricas da imagem. A função verifica se cada pixel da imagem convertida para escala cinza, possui o nivel cinza 0 e muda o valor para 255 e todos os valores maiores do que preto muda para preto. Com isso obtemos uma figura em que a borda da das figuras possui uma cor branca e todo o resto da imagem é preto. Assim obtemos os contornos das formas geométricas num fundo preto para trabalhar com o algoritimo de identificação dos contornos. A restrição dessa abordagem é que para identificar a borda da figuras da imagem é necessário que na conversão para escala cinza não haja nada além da borda no nivel de de cinza 0, pois isso faria com que outras partes da imagem ficassem com o nivel branco e não só as bordas das figuras. Na figura 2 verficamos como foi essa transformação.



(a) Original



(b) Escala Cinza

Figure 1: Transformação de Cores (a)RGB para (b)Cinza

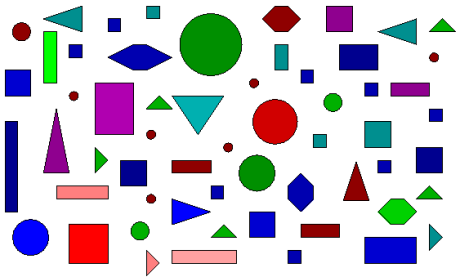
3.3 Extração de Propriedades dos Objetos

Nessa etapa, o programa utiliza da função findContours() da biblioteca OpenCV para fazer a análise da imagem gerada a partir da etapa Contorno de Objetos, ou seja, a imagem que possue o fundo preto e o contorno branco dos objetos. FindContours() é capaz de identificar os contornos presentes na imagem passada para ela e retornar cada um dos contornos e alguns atributos desses contornos em um vetor de ponteiros. Com essa lista pudemos calcular o número de regiões identificadas e os centroide, perimetro e área de cada região. Para encontrar o numero de regiões, utilizamos a função len() de python para calcular o tamanho do vetor de contornos, o perimetro e a área foram calculadas com a função arcLength() e contourArea() respectivamente para cada um dos contornos. No caso do cálculo do centroide foi utlizado a função Moments() que retorna que retorna uma estrutura de dados dos momentos espaciais, centrais e centrais normalizados da figura geométrica. Com esses dados conseguimos calcular o centro de massa da figura geométrica, isto é, seu centroide, utlizando a equação

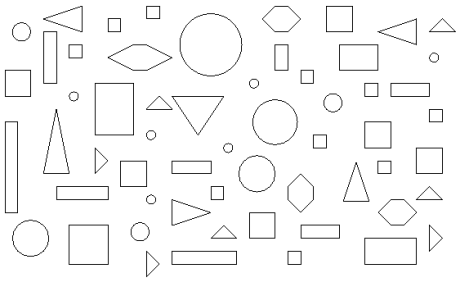
$$\bar{x} = \frac{\mathtt{m}_{10}}{\mathtt{m}_{00}}, \; \bar{y} = \frac{\mathtt{m}_{01}}{\mathtt{m}_{00}} \tag{1}$$

sendo m_{ij} elementos da estrura de dados retornada de Moments().

Além de trazer as informações referentes as regiões, o programa retorna uma imagem com a numeração de cada região utilizando a função putText() do



(a) Original



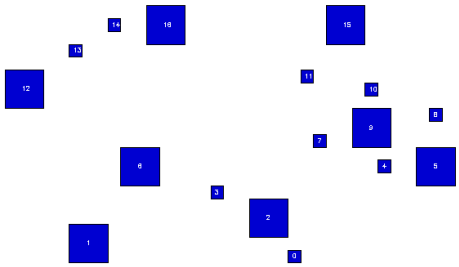
(b) Contorno das Figuras Geométricas

Figure 2: Contorno dos objetos. Vale observar que em (b) os contornos estão em preto e o fundo em branco para a redução do consumo de tinta preta na hora da impressão

OpenCV utilizando as coordenadas do centroide para a escrita no local correto da imagem. No entato, foi-se necessário deslocar manualmente a posição de escrita em relação ao centroide devido ao fato do texto ficar deslocado, por causa do tamanho da fonte. Um exemplo da saída do programa pode ser visto na Figura 3.

3.4 Histograma de Area dos Objetos

Nessa ultima etapa, o programa retorna um histograma mostrando o numero de regiões com área menor que 1500px, o numero de regiões com área maior igual a 1500px e menor do que 3000px e o numero de regiões com área maior igual a 3000px. Além disso, retorna o numero de regiões para cada uma dessas condições. Nessa parte foi utilizado a função de geração de histogramas da biblioteca matplotlib que recebe como um dos parâmetros o vetor que possui o tamanho de área de cada uma das regiões da imagem. Para gerar esse vetor foi usado um laço em que montamos o vetor iterando sobre o vetor dos contornos e obtendo suas áreas. Um exemplo do retorno programa encontra-se na Figura 4.



(a) Imagem com as regiões numeradas

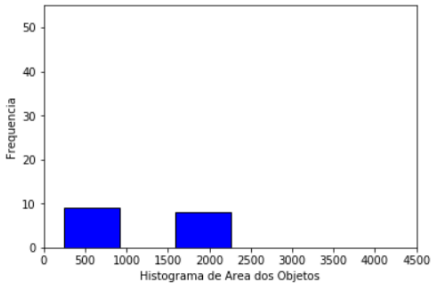
```
Numero de regioes: 17
regiao: 0               centroide: (360, 312)          perimetro: 62          area: 240
regiao: 1               centroide: (108, 296)          perimetro: 190         area: 2256
regiao: 2               centroide: (328, 265)          perimetro: 188         area: 2209
regiao: 3               centroide: (265, 234)          perimetro: 62          area: 240
regiao: 4               centroide: (470, 202)          perimetro: 64          area: 256
regiao: 5               centroide: (533, 202)          perimetro: 190         area: 2256
regiao: 6               centroide: (171, 202)          perimetro: 190         area: 2256
regiao: 7               centroide: (391, 171)          perimetro: 64          area: 256
regiao: 8               centroide: (533, 139)          perimetro: 64          area: 256
regiao: 9               centroide: (454, 155)          perimetro: 190         area: 2256
regiao: 10              centroide: (454, 108)          perimetro: 64          area: 256
regiao: 11              centroide: (375, 92)           perimetro: 62          area: 240
regiao: 12              centroide: (29, 107)           perimetro: 188         area: 2209
regiao: 13              centroide: (92, 60)            perimetro: 62          area: 240
regiao: 14              centroide: (139, 29)           perimetro: 62          area: 240
regiao: 15              centroide: (422, 29)           perimetro: 190         area: 2256
regiao: 16              centroide: (202, 29)           perimetro: 190         area: 2256
```

(b) Informações sobre cada uma das regiões

Figure 3: Extração de Propriedades dos Objetos

```
Numero de Regioes Pequenas: 9
Numero de Regioes Medias: 8
Numero de Regioes Grandes: 0
```

(a) Retorno com o número de regiões



(b) Histograma sobre o número de regiões pequenas, medias e grandes

Figure 4: Histograma de Area dos Objetos. Nesse caso a função foi aplicada sobre regiões presentes na figura 3

4 Conclusão

Nesse trabalho foi possível aprender a utlizar algumas das funções presentes na biblioteca OpenCV, como também aprender sobre a utlização de filtragens para

obtenção de certos detalhes na figura, como foi o caso da função threshold().

5 Referências

[1] OpenCV 3.4.1 documentation - https://docs.opencv.org/3.4.1/