MC920 - Introdução ao Processamento Digital de Imagem - 1s2018

Fábio Takahashi Tanniguchi - RA 145980

Relatório - Trabalho 1

1. Introdução

O trabalho proposto consiste na realização de algumas operações com imagens (em formato PNG) utilizando a linguagem Python e suas diversas bibliotecas disponíveis:

- carregar uma imagem colorida transformando para imagem em escala de cinza;
- identificar as bordas dos objetos presentes na imagem;
- extrair perímetro e área dos objetos identificados;
- classificar os objetos em três conjuntos (especificados de acordo com a área) e exibir um histograma.

2. Execução, entradas e saídas

O código-fonte está contido inteiramente no arquivo *t1.py* e está implementado em Python 3.6.

Para executá-lo, é necessário colocar o caminho de uma imagem em PNG como parâmetro de execução.

Por exemplo: *python t1.py 'D:/Google Drive/MC920/objetos1.png'*

As informações de área e perímetro são exibidas no console. Já as imagens salvas pelo programa serão salvas na pasta de execução com nomes:

- *"1_grayscale_145980.png"* para a imagem em escala de cinza,
- *"2_borders_145980.png"* para a exibição das bordas detectadas,
- *"3_object_names_145980.png"* para a exibição dos objetos nomeados (referência para perímetro e área) e
- *"4_histogram_145980.png"* para o histograma com as áreas classificadas.

3. Descrição da solução

Para a leitura da imagem original, foi utilizado o método *imread* da biblioteca *SciPy*, que a armazena como um *ndarray*, tipo contido na biblioteca *NumPy*. Caso a imagem seja colorida, ela é automaticamente transformada para escala de cinza, o que é definido pelo segundo parâmetro do método *imread*. A exportação dessa imagem em escala de cinza para arquivo PNG é feita por meio da biblioteca *matplotlib*.

3.1. Identificação das bordas dos objetos

Para obter os contornos dos objetos presentes na imagem, foi utilizada a biblioteca *OpenCV*.

Primeiramente, foi aplicada uma máscara para separar claramente o fundo dos objetos. A máscara deixa o fundo preto e os objetos brancos. Dessa forma, a imagem resultante fica adequada para ser usada pelo método *findContours*, que retorna um conjunto de contornos dos objetos identificados.

Os contornos obtidos foram desenhados em uma imagem totalmente branca e essa imagem contendo apenas as bordas dos objetos foi salva utilizando a biblioteca *matplotlib*.

3.2. Numeração dos objetos, extração de área e perímetro e classificação

Como na operação 3.1 foi obtida a lista de bordas de objetos, então, para cada objeto encontrado foi dado um número sequencial. Aproveitando o laço de repetição que percorria a lista de bordas foram realizados os cálculos de área e perímetro. Além disso, em uma imagem inicialmente apenas com as bordas dos objetos, foram desenhados os números nos centróides de seus respectivos objetos.

Para a obtenção de área e perímetro dos objetos, foram utilizadas funções da biblioteca *OpenCV*. A função *arcLength* calcula o comprimento das bordas, ou seja, o perímetro de um objeto com base em um contorno obtido por *findContours*. Já a função *contourArea* calcula a área dentro de um contorno, ou seja, a área do objeto. Ambos os dados calculados são exibidos no console.

O mesmo laço de repetição citado que percorre a lista de bordas de objetos também foi aproveitado para classificar os objetos com base em suas respectivas áreas. Conforme especificações deste trabalho, os objetos deveriam ser classificados em pequenos (área < 1500 pixels), médios (1500 <= área < 3000 pixels) e grandes (área >= 3000 pixels). Portanto, assim que cada objeto tinha sua área calculada, ele era guardado em um dos três conjuntos de objetos.

Todo o processamento realizado percorrendo os contornos dos objetos permitiu posteriormente exportar uma imagem contendo os contornos dos objetos e seus respectivos números atribuídos e o histograma de objetos classificados.

4. Testes e resultados obtidos

Foram realizados testes principalmente utilizando três imagens:

- *“objetos1.png”*, contida no conjunto de amostras fornecido inicialmente e presente no enunciado;
- *“objetos2.png”*, também contida no conjunto de amostras fornecido inicialmente e presente no enunciado;
- *“car.png”*, imagem contendo um desenho simples de um carro.

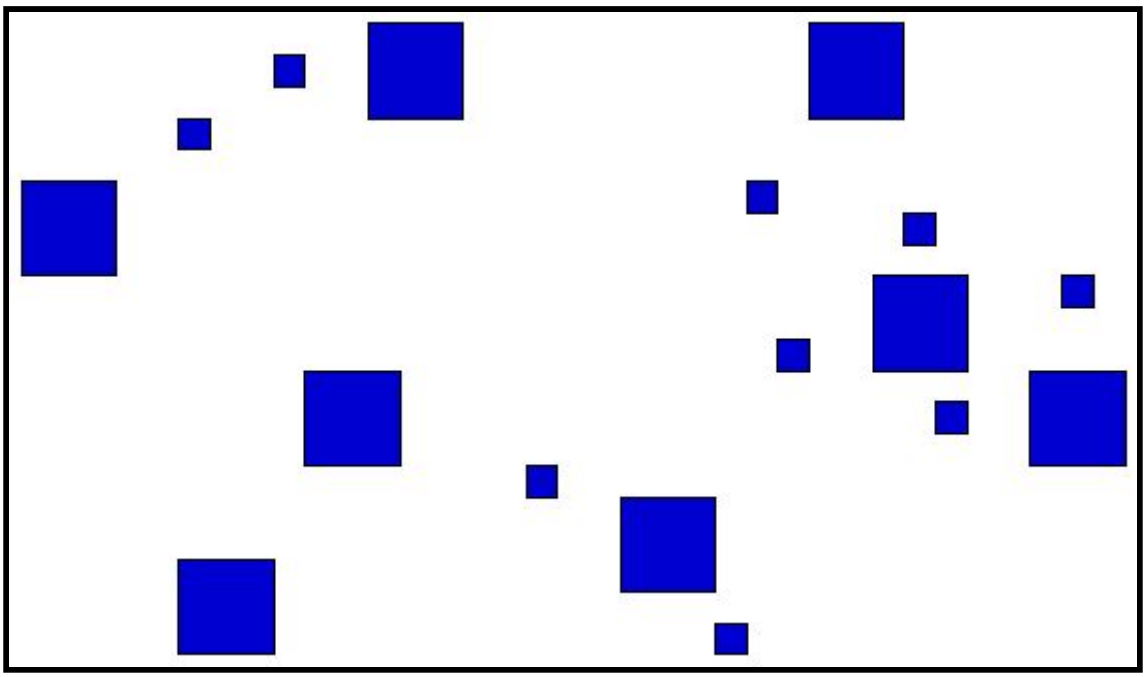

Figura 1 - Imagem “objetos1.png” (bordas da imagem colocadas posteriormente para melhor visualização neste relatório)

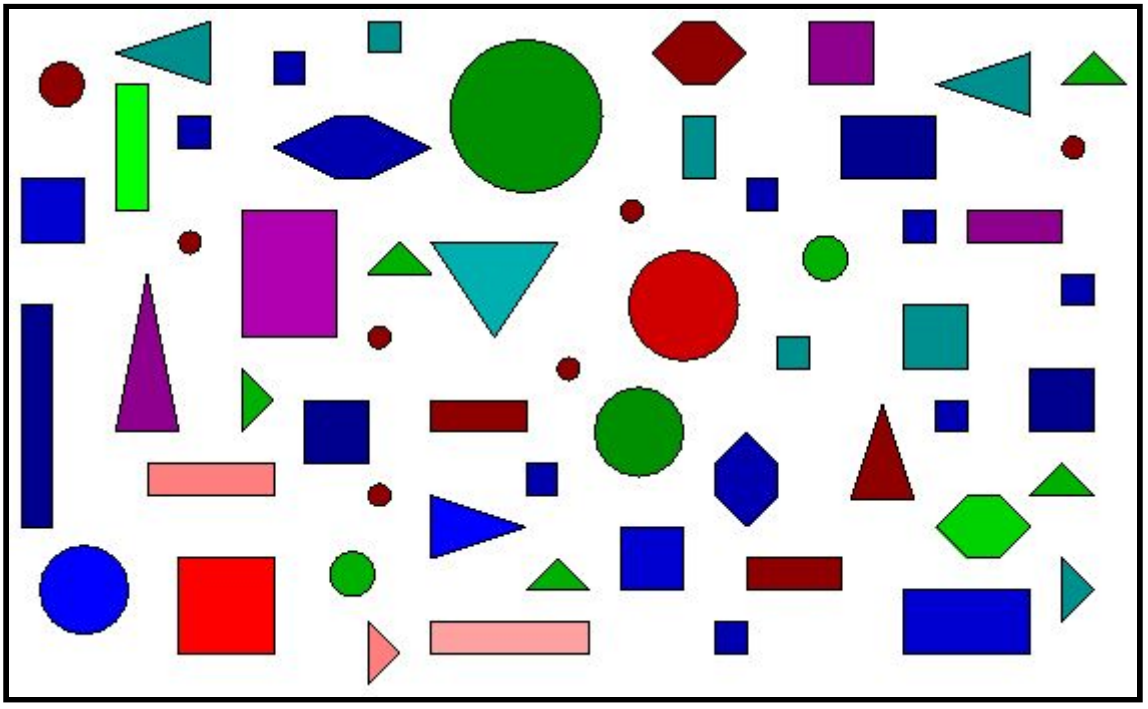

Figura 2 - Imagem “objetos2.png” (bordas da imagem colocadas posteriormente para melhor visualização neste relatório)

Figura 3 - Imagem "car.png" (bordas da imagem colocadas posteriormente para melhor visualização neste relatório)

Como se pode perceber, a imagem *"objetos1.png"* é caracterizada por possuir quadrados desenhados de cor azul. Além de ser a imagem utilizada como exemplo no enunciado, trata-se de um caso de teste básico para verificar a conversão para escala de cinza e a detecção dos objetos.

Já a imagem *"objetos2.png"* possui uma quantidade bastante considerável de objetos desenhados e de diversas cores. Essas características poderiam confirmar se o algoritmo está funcionando a contento em uma imagem mais densa.

A imagem *"car.png"* foi utilizada com o objetivo de ser um caso de teste além das imagens fornecidas previamente.

4.1. Imagem *"objetos1.png"*

A imagem dos quadrados azuis, como já foi explicado, foi utilizada para comparar as saídas do programa com os resultados esperados mostrados no enunciado do trabalho.

Não houve nenhuma grande dificuldade em utilizá-la. O carregamento da imagem e a transformação automática em escala de cinza durante o carregamento não foram difíceis pois já haviam sido trabalhados na atividade anterior opcional.

Quanto às demais operações, o único ponto a ser destacado é o cálculo de área e perímetro dos objetos. Ambos não ficaram exatamente iguais aos exibidos no enunciado, o que pode ter sido causado por diversos fatores.

A biblioteca utilizada para detectar os contornos e realizar os cálculos de área e perímetro pode ter sido diferente da utilizada para gerar os exemplos mostrados no enunciado. Além disso, a própria função de identificação de contornos tem um parâmetro que indica o tipo de otimização de bordas que deve ser realizado. Qualquer algoritmo para reduzir o número de pontos que compõem o contorno afeta diretamente os dados calculados. No caso deste trabalho, foi utilizado o parâmetro *CHAIN_APPROX_NONE*, que, segundo a documentação do *OpenCV*, deveria dispensar qualquer otimização.

Indo ainda além, a imagem pode ter sido carregada utilizando bibliotecas diferentes ou parâmetros diferentes, podendo alterar levemente os resultados das operações seguintes.

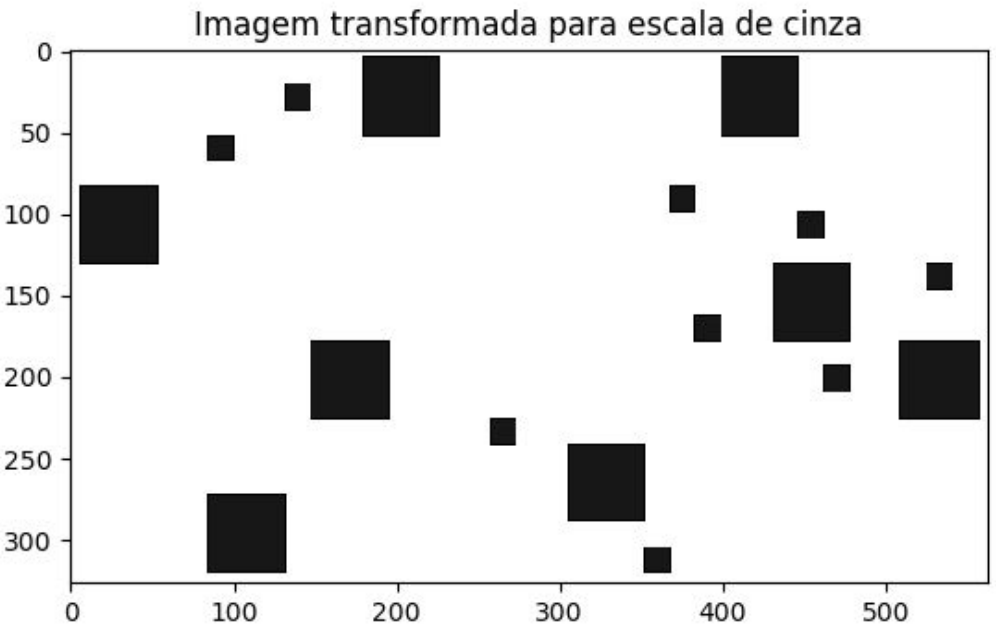

Figura 4 - "objetos1.png" em escala de cinza

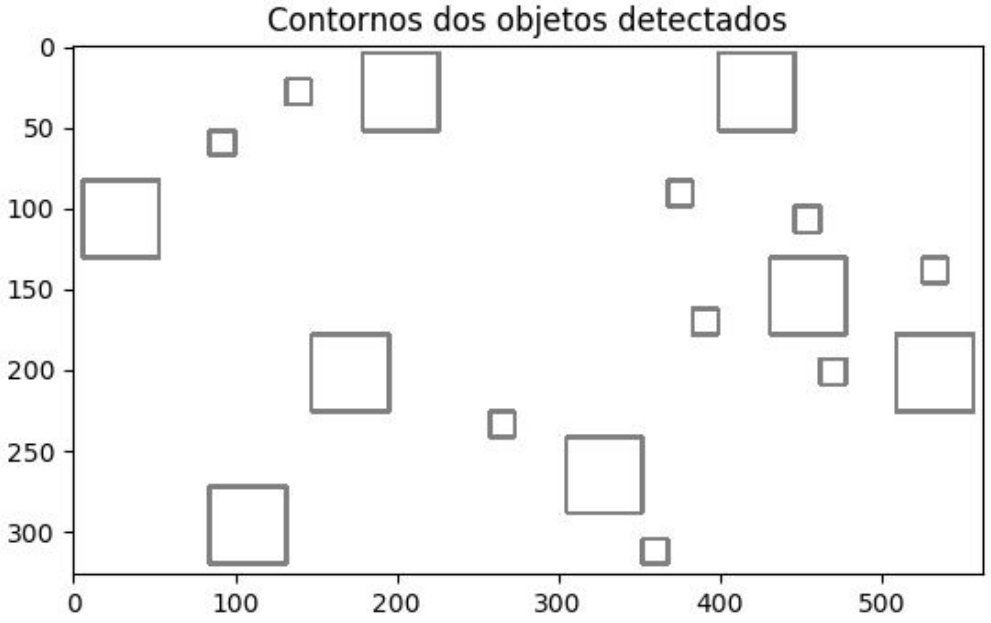

Figura 5 - Bordas detectadas em "objetos1.png"

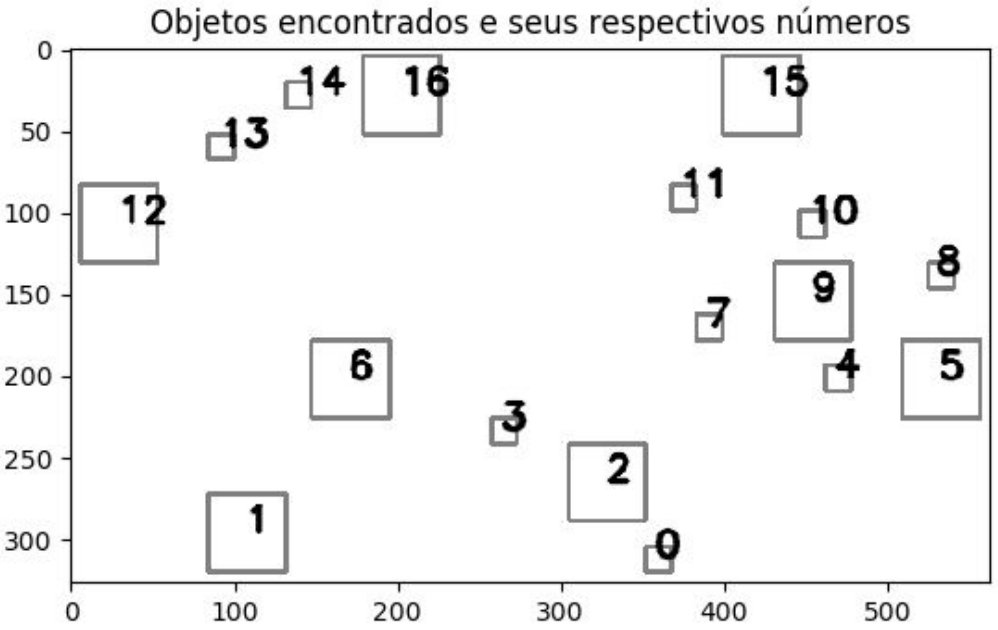

Figura 6 - Objetos numerados em “objetos1.png”

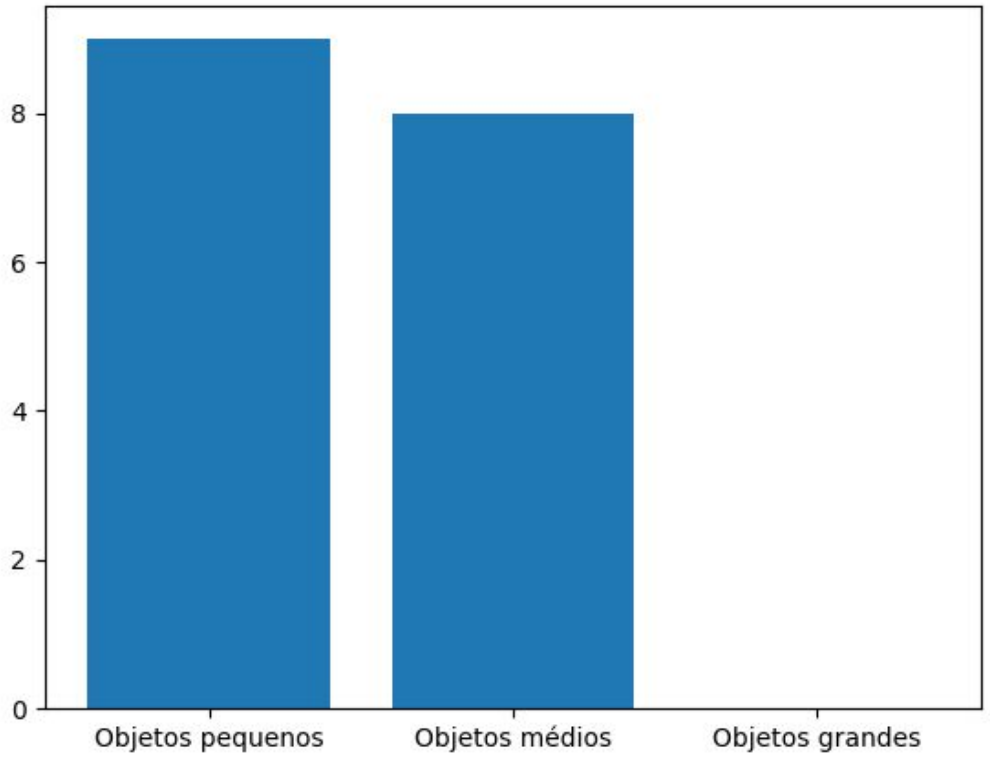

Figura 7 - Histograma dos objetos classificados em “objetos1.png”

Os dados de perímetro e área dos objetos da imagem são exibidos no console estão reproduzidos abaixo:

```
Número de regiões:  17
região:  0  | perímetro:  62.0  | área:  240.0
região:  1  | perímetro:  190.0  | área:  2256.0
região:  2  | perímetro:  188.0  | área:  2209.0
região:  3  | perímetro:  62.0  | área:  240.0
região:  4  | perímetro:  64.0  | área:  256.0
região:  5  | perímetro:  190.0  | área:  2256.0
região:  6  | perímetro:  190.0  | área:  2256.0
região:  7  | perímetro:  64.0  | área:  256.0
região:  8  | perímetro:  64.0  | área:  256.0
região:  9  | perímetro:  190.0  | área:  2256.0
região:  10  | perímetro:  64.0  | área:  256.0
região:  11  | perímetro:  62.0  | área:  240.0
região:  12  | perímetro:  188.0  | área:  2209.0
região:  13  | perímetro:  62.0  | área:  240.0
região:  14  | perímetro:  62.0  | área:  240.0
região:  15  | perímetro:  190.0  | área:  2256.0
região:  16  | perímetro:  190.0  | área:  2256.0
```

4.2. Imagem *“objetos2.png”*

Apesar da alta densidade de objetos e da variedade de cores dos objetos na imagem, não houve problemas nas operações realizadas.

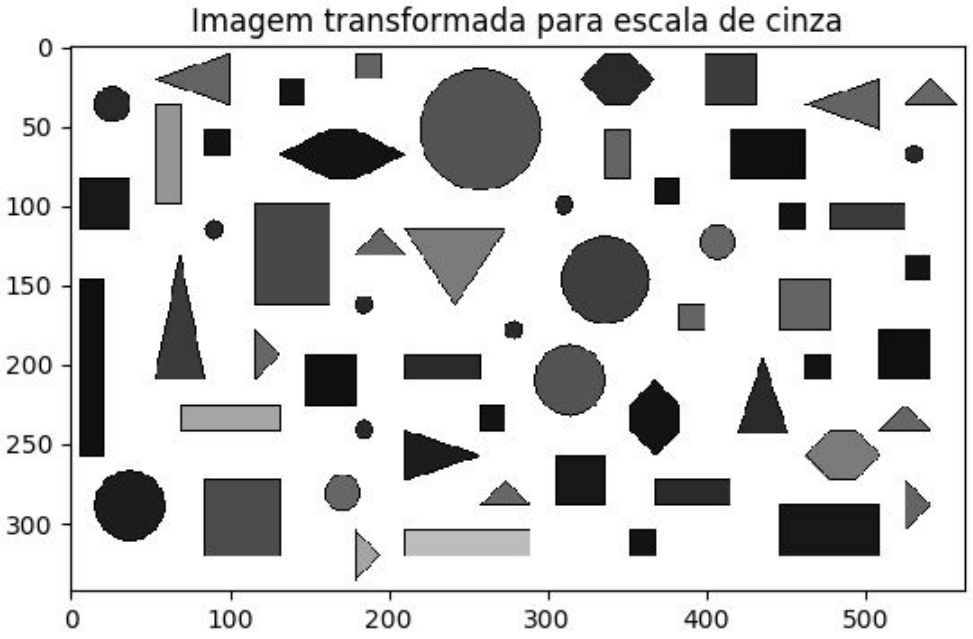

Figura 8 - “objetos2.png” em escala de cinza

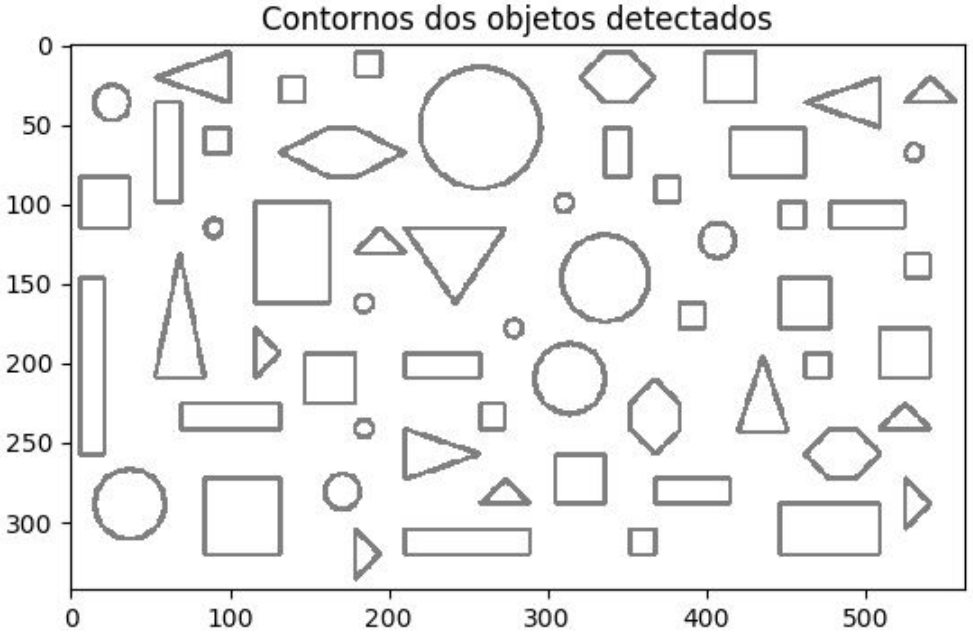

Figura 9 - Contornos dos objetos identificados em “objetos2.png”

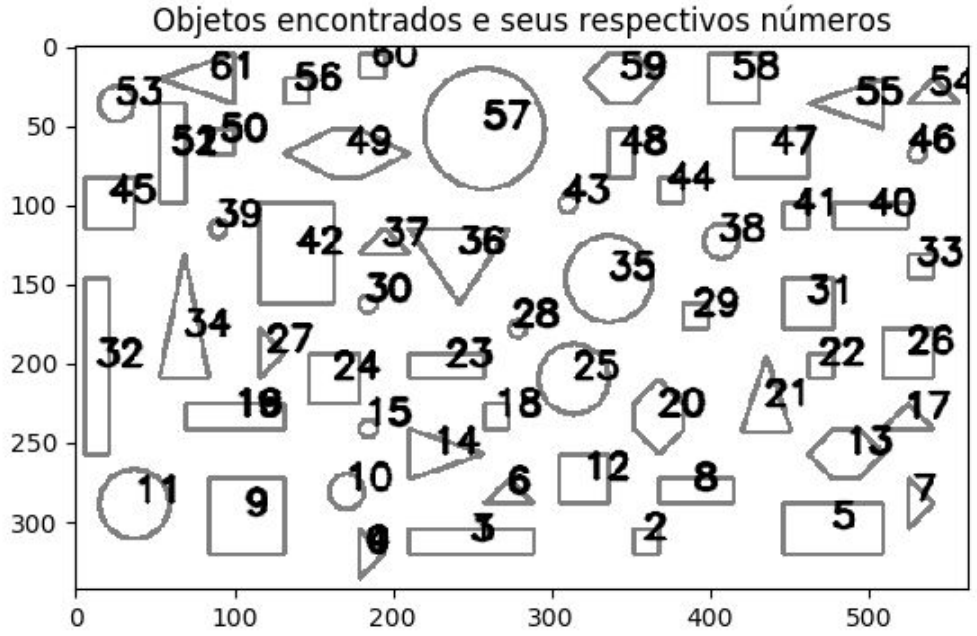

Figura 10 - Objetos identificados em “objetos2.png” numerados

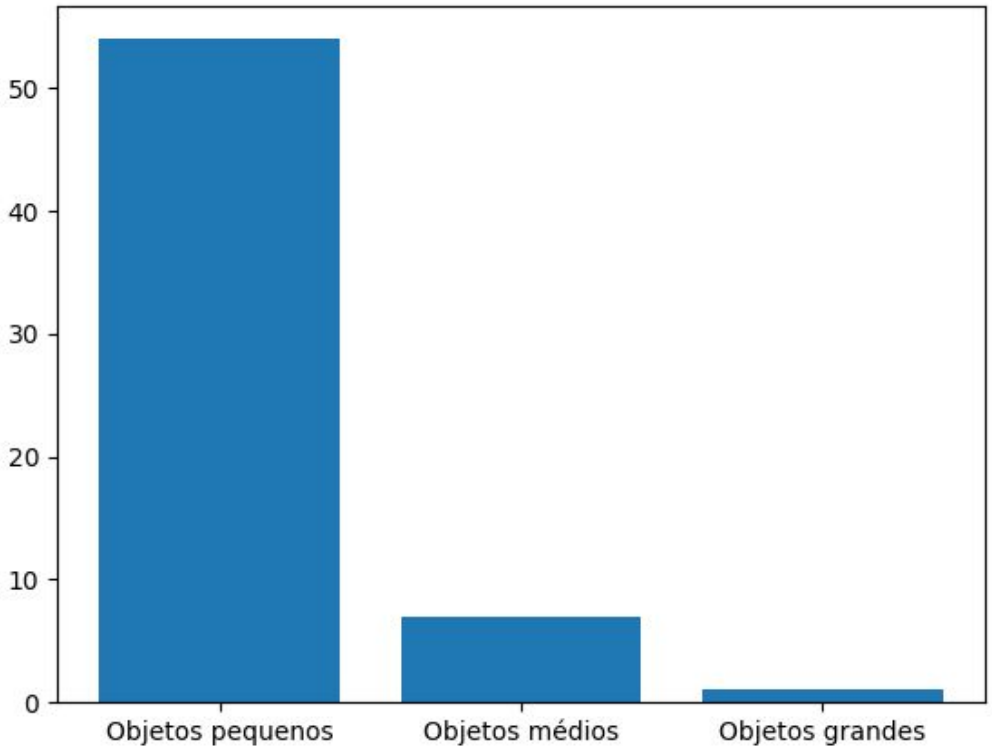

Figura 11 - Histograma dos objetos classificados em “objetos2.png”

Os dados de perímetro e área dos objetos da imagem são exibidos no console estão reproduzidos abaixo:

```
Número de regiões:  62
região:  0  | perímetro:  70.4  | área:  238.0
região:  1  | perímetro:  187.7  | área:  1262.0
região:  2  | perímetro:  64.0  | área:  256.0
região:  3  | perímetro:  190.0  | área:  1264.0
região:  4  | perímetro:  74.4  | área:  240.0
região:  5  | perímetro:  190.0  | área:  2016.0
região:  6  | perímetro:  74.4  | área:  240.0
região:  7  | perímetro:  77.3  | área:  256.0
região:  8  | perímetro:  126.0  | área:  752.0
região:  9  | perímetro:  192.0  | área:  2304.0
região:  10  | perímetro:  73.9  | área:  386.0
região:  11  | perímetro:  146.7  | área:  1512.0
região:  12  | perímetro:  124.0  | área:  961.0
região:  13  | perímetro:  120.3  | área:  991.5
região:  14  | perímetro:  137.8  | área:  751.5
região:  15  | perímetro:  37.0  | área:  95.0
região:  16  | perímetro:  155.7  | área:  1006.0
região:  17  | perímetro:  77.3  | área:  256.0
região:  18  | perímetro:  62.0  | área:  240.0
região:  19  | perímetro:  158.0  | área:  1008.0
região:  20  | perímetro:  119.4  | área:  990.5
região:  21  | perímetro:  138.4  | área:  752.0
região:  22  | perímetro:  62.0  | área:  240.0
região:  23  | perímetro:  126.0  | área:  720.0
região:  24  | perímetro:  126.0  | área:  992.0
região:  25  | perímetro:  147.5  | área:  1519.0
região:  26  | perímetro:  126.0  | área:  992.0
região:  27  | perímetro:  74.4  | área:  240.0
região:  28  | perímetro:  37.8  | área:  95.0
região:  29  | perímetro:  64.0  | área:  256.0
região:  30  | perímetro:  36.4  | área:  96.5
região:  31  | perímetro:  128.0  | área:  1024.0
região:  32  | perímetro:  252.0  | área:  1665.0
região:  33  | perímetro:  62.0  | área:  240.0
região:  34  | perímetro:  200.4  | área:  1233.0
região:  35  | perímetro:  181.3  | área:  2324.0
região:  36  | perímetro:  183.1  | área:  1488.5
região:  37  | perímetro:  76.4  | área:  271.0
região:  38  | perímetro:  73.9  | área:  384.0
região:  39  | perímetro:  37.8  | área:  94.0
região:  40  | perímetro:  126.0  | área:  752.0
região:  41  | perímetro:  64.0  | área:  256.0
região:  42  | perímetro:  220.0  | área:  2961.0
```

região: 43 | perímetro: 36.4 | área: 96.5
região: 44 | perímetro: 62.0 | área: 240.0
região: 45 | perímetro: 126.0 | área: 992.0
região: 46 | perímetro: 37.0 | área: 96.0
região: 47 | perímetro: 156.0 | área: 1457.0
região: 48 | perímetro: 94.0 | área: 496.0
região: 49 | perímetro: 182.3 | área: 1472.5
região: 50 | perímetro: 64.0 | área: 256.0
região: 51 | perímetro: 155.7 | área: 1006.0
região: 52 | perímetro: 158.0 | área: 1008.0
região: 53 | perímetro: 73.9 | área: 386.0
região: 54 | perímetro: 77.3 | área: 256.0
região: 55 | perímetro: 139.3 | área: 721.0
região: 56 | perímetro: 62.0 | área: 240.0
região: 57 | perímetro: 251.3 | área: 4508.0
região: 58 | perímetro: 126.0 | área: 992.0
região: 59 | perímetro: 118.9 | área: 976.0
região: 60 | perímetro: 62.0 | área: 240.0
região: 61 | perímetro: 138.4 | área: 752.0

4.3. Imagem *“car.png”*

A imagem *“car.png”*, que consiste em um desenho simples de um automóvel, apesar de bastante simples, inicialmente o programa implementado não identificava adequadamente os objetos.

Após uma pequena análise na imagem, foi detectado que ela estava com o fundo transparente, o que afetava a forma como o método *imread* convertia em escala de cinza. Foi preenchido o fundo com cor branca e então o programa passou a detectar adequadamente as bordas.

Figura 8 - “car.png” em escala de cinza

Figura 9 - Contornos identificados em “car.png”

Figura 10 - Objetos identificados em “car.png” numerados. Deve ser notado que o início da numeração de cada objeto está posicionado em seu centróide

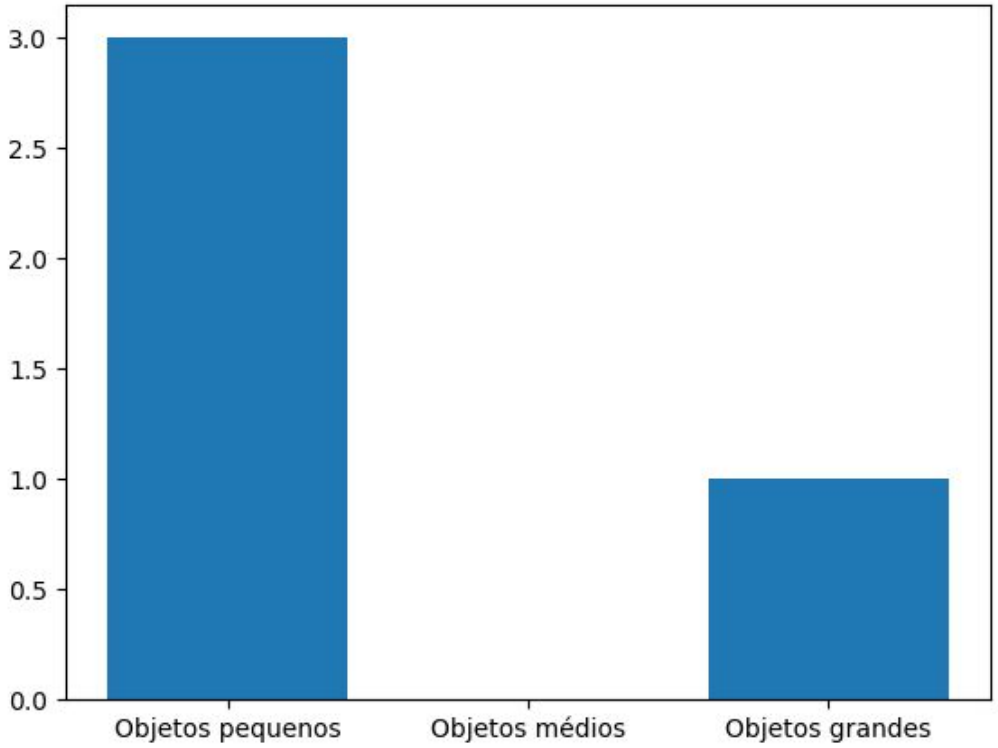

Figura 11 - Histograma dos objetos classificados em “objetos2.png”

Os dados de perímetro e área dos objetos da imagem são exibidos no console estão reproduzidos abaixo:

```
Número de regiões:  4
região:  0  | perímetro:  48.6  | área:  166.0
região:  1  | perímetro:  48.6  | área:  166.0
região:  2  | perímetro:  138.3  | área:  1044.0
região:  3  | perímetro:  268.5  | área:  3867.5
```

5. Conclusão

A detecção de objetos em imagens não é uma tarefa difícil em Python, uma vez que as bibliotecas disponíveis oferecem inúmeros recursos prontos. Ainda assim, as técnicas abordadas neste trabalho parecem ser essenciais para conseguir posteriormente desenvolver trabalhos mais complexos, como reconhecimento em imagens.