Exercício Programa 3

Parte 0: *Requisitos do projeto*

Foi necessária a criação dos requisitos mínimos do projeto devido a alguns equívocos já vistos nas entregas do primeiro EP:

- Para correção, não apenas será visto o código, como o Jupyter notebook (JN) também será executado. Certifique que não existirá erros ao executarmos todas as células do JN;
- Todos os arquivos necessários para executar seu JN **devem** estar no Github;
- **Não** devem existir figuras **duplicadas**. Por exemplo, para você saber quais classes pertencem a determinada imagem, deve-se olhar os metadados;
- Existem 3 opções de metadados, em nenhuma delas o metadado deve estar nos dados do arquivo de imagem, favor atentar as opções dadas no EP1;
- Lembre-se que os EPs são cumulativos, erros não corrigidos antes, podem afetar resultados dos outros EPs.
- Deve ter um JN para cada parte, a saída de um JN será a entrada do próximo;
- Deve-se imprimir todas as imagens com o resultado final das arestas, estas imagens serão utilizadas para a seleção da database para a parte 2.

O Exercício Programa 3 (EP3) consiste em 3 partes: detecção das arestas, extração de características e classificação. Cada parte deverá ser feita em um JN diferente.

Parte 1 - Segmentação dos objetos de interesse

O objetivo desta etapa é a segmentação do objeto, produzindo uma imagem binária: 0 para o fundo e 1 para as arestas do objeto. Iremos utilizar a segmentação automática para identificar as arestas objetos nas imagens.

A segmentação deverá ser executada para todos os datasets de imagens, com exceção do dataset original (*originalGrayDataset*, *augmentedDataset* e *normalizedDataset*)

Passos para a segmentação:

1. Faça a suavização das imagens: **Gaussian Blurring**, **Median Blurring** e **Bilateral Filtering**. Pode testar os 3 até a última etapa e visualmente determinar qual encontrou as arestas dos objetos em mais imagens;

2. Gere as imagens binárias: **Gradient + Thresholding**, **Canny Edge Detection e Marr-Hildreth Edge Detection.** Da mesma forma que antes, teste os 3 métodos até o último passo, e veja qual encontrou mais imagens com arestas de todos os objetos. Ao final, teremos as arestas com intensidade de 255 e o restante da imagem com intensidade 0. A imagem terá várias arestas encontradas no background, iremos remover elas no passo 4;

3. Trace o contorno do objeto:

```
import cv2
contours, hierarchy = cv2.findContours(img, cv2.RETR_TREE,
                                                cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(img, contours, -1,(255,0,255),3)
```

As imagens usadas, são as imagens que contém apenas as arestas encontradas. O parâmetro RETR_TREE significa que a função recupera todos os contornos da imagem e, além disso, cria uma estrutura que mostra quais contornos estão dentro de outros contornos. Usando CHAIN_APPROX_SIMPLE, serão removidos todos os pontos redundantes, ou seja, reduzirá o tamanho final da imagem. O retorno da função dado pela variável *contours*, serão os contornos encontrados da imagem contendo arestas, e cada contorno será uma componente conexa.

4. Calcule a área do contorno:

```
import cv2
areas = []
for c in contours:
    areas.append(cv2.contourArea(c))
```

A lista *areas*, terá os valores do cálculo da área de todos os contornos encontrados, é esperado que caso os contornos dos objetos tenham sido encontrados corretamente, as maiores áreas representem os contornos dos objetos da imagem. Ordene a lista do maior para o menor valor.

5. Trace apenas os contornos das maiores áreas:

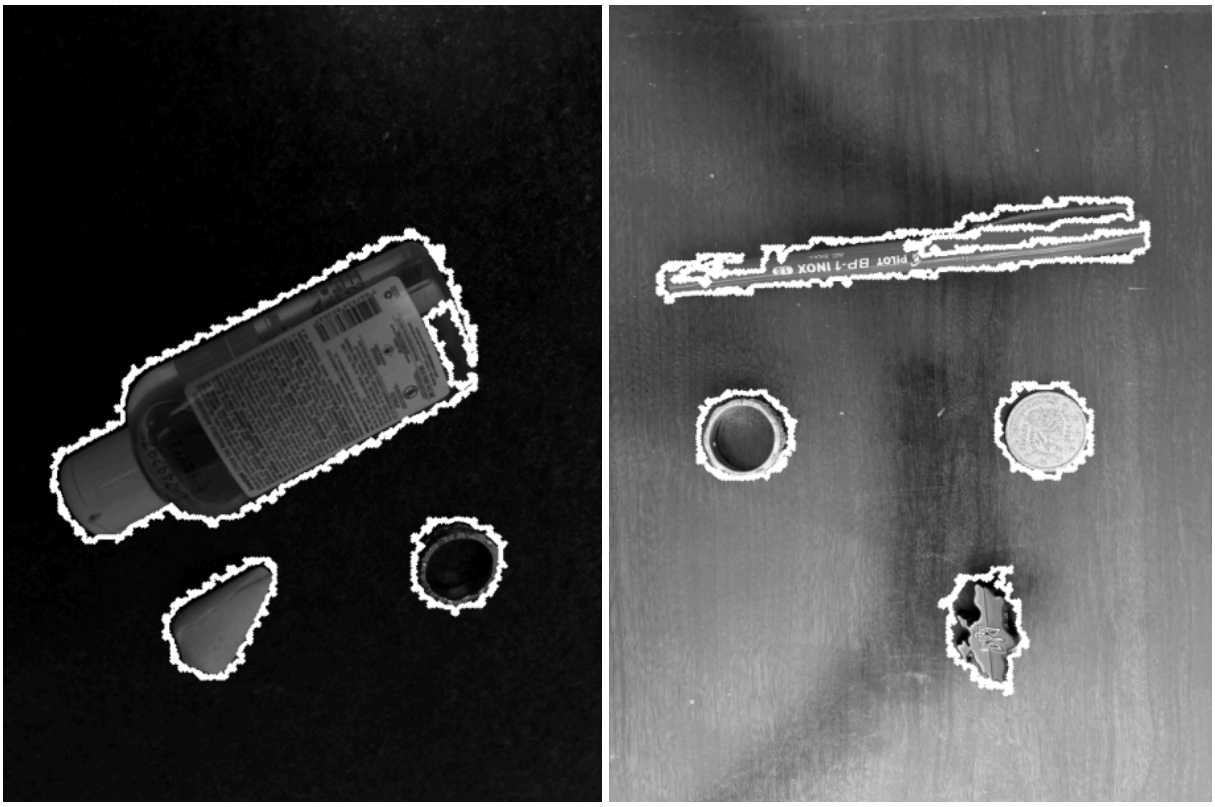
```
import cv2
maiores_areas = []
for c in contours:
    if (cv2.contourArea(c)) > areas_ordenada[n]:
      maiores_areas.append(c)

img_contours = cv2.drawContours(img, maiores_areas, -1,(255,0,255),3)
```

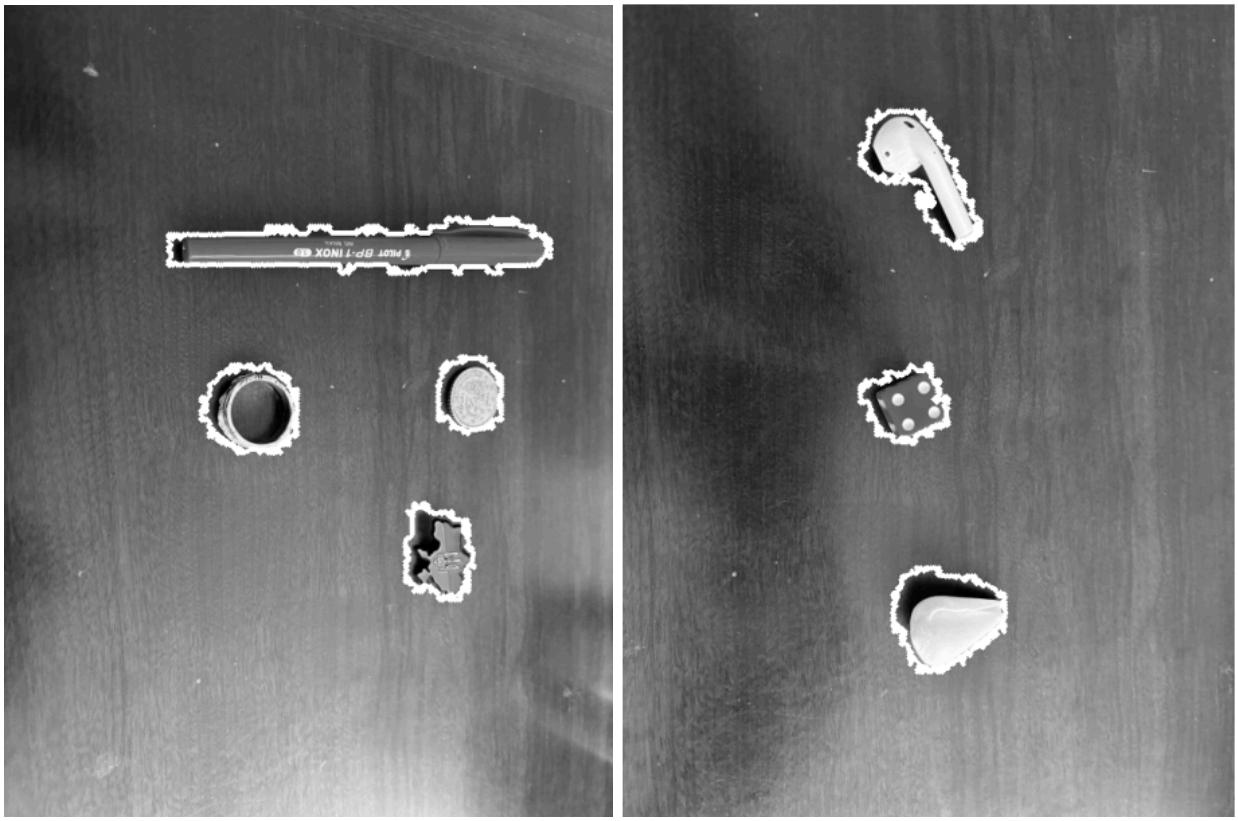
Neste código, serão selecionados os maiores contornos encontrados, utilizando a lista *areas* que encontra-se ordenada na variável *areas_ordenada*. Os **n** maiores contornos com as maiores áreas deverão ser desenhados na imagem em cinza pela função *cv2.drawContours*. Note que no código acima, **n** é o número de objetos que cada imagem possui. Caso tenha imagens com diferentes quantidades de objetos, use seus metadados para determinar o **n**.

6. Seleção manual das imagens: Ao final, é esperado que em diversas imagens, as arestas não sejam corretamente encontradas. Iremos fazer um filtro manual, onde

selecionaremos apenas as imagens onde teremos o contorno correto em todos os objetos.

Abaixo, temos um exemplo usando imagens de uma pessoa da turma para mostrar as imagens que deverão ser selecionadas:





Note que o contorno não precisa traçar perfeitamente a borda do objeto, mas precisa estar próximo, caso tenha alguma alguma região que esteja mais distante, como no último caso, fica a critério o uso ou não. Caso tenha contornos dentro da imagem como nos dois primeiros casos, a imagem pode ser usada, desde que tenha encontrado corretamente o contorno do objeto.

As imagens selecionadas devem ser salvas em outra pasta. É possível fazer um script para copiar as imagens selecionadas dos datasets, mas se preferir, pode copiar manualmente as imagens e fazer o upload no GitHub juntamente com os JN.

Parte 2 - Extração de características

Das imagens selecionadas, serão extraídas 3 características das arestas: a área, o diâmetro e a maior perpendicular ao diâmetro (minor).

Iremos apenas usar os dados dos contornos das imagens para a extração. A extração poderia ser realizada manualmente, porém, iremos usar uma biblioteca para otimizar o tempo necessário para fazer o EP. Usaremos o *skimage.measure* para a extração das 3 características dos objetos.

O código abaixo pode ser utilizado para o cálculo das 3 características, dado os valores dos contornos obtidos pelo opencv.

```
from skimage.measure import label, regionprops

label_img = label(img_contours, connectivity=2)
props = regionprops(label_img)
props_img = props
```

No código acima, *img_contours* deve ser a imagem com apenas os contornos dos objetos, e *connectivity* é o métido de seleção de vizinhos, 1 para 4-conectado e 2 para 8-conectado. Ao final, *props_img* será uma lista com cada uma das características para cada componente conexa da imagem.

Com os dados das 3 características, deve-se montar uma tabela com os mesmas colunas, como abaixo:

Objeto	Classe	n_contorno	Área	Diâmetro	Minor	Imagem

Deve-se associar manualmente qual é o contorno encontrado com o objeto, pode ser pela ordem das áreas, da maior para a menor. Então, a maior área será o contorno 0, após 1, e assim por diante. O importante é conseguir recuperar o contorno do objeto, para encontrar as coordenadas do objeto na imagem.

Parte 3 - Treinamento do modelo e classificação do objeto

Importe a sua matriz, utilizando o pandas. Para a classificação, siga os seguintes passos:

1. Na matriz, separe imagens, para ao menos termos 2 objetos diferentes para usarmos como validação ao final. Teremos 2 objetos pandas: um com os dados para treino e teste, outro com as imagens separadas manualmente para a classificação;

2. Iremos dividir os dados em treinamento e teste:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    stratify=y,
                                                    test_size=0.15,
                                                    random_state=0)
```

Acima, *X* é um numpy array com os dados, e *y* um numpy array com cada uma das classes (As classes usadas da matriz serão apenas classe, área, diâmetro e minor). A divisão acima separa 15% das amostras para teste, mas é possível alterar este valor se julgar necessário;

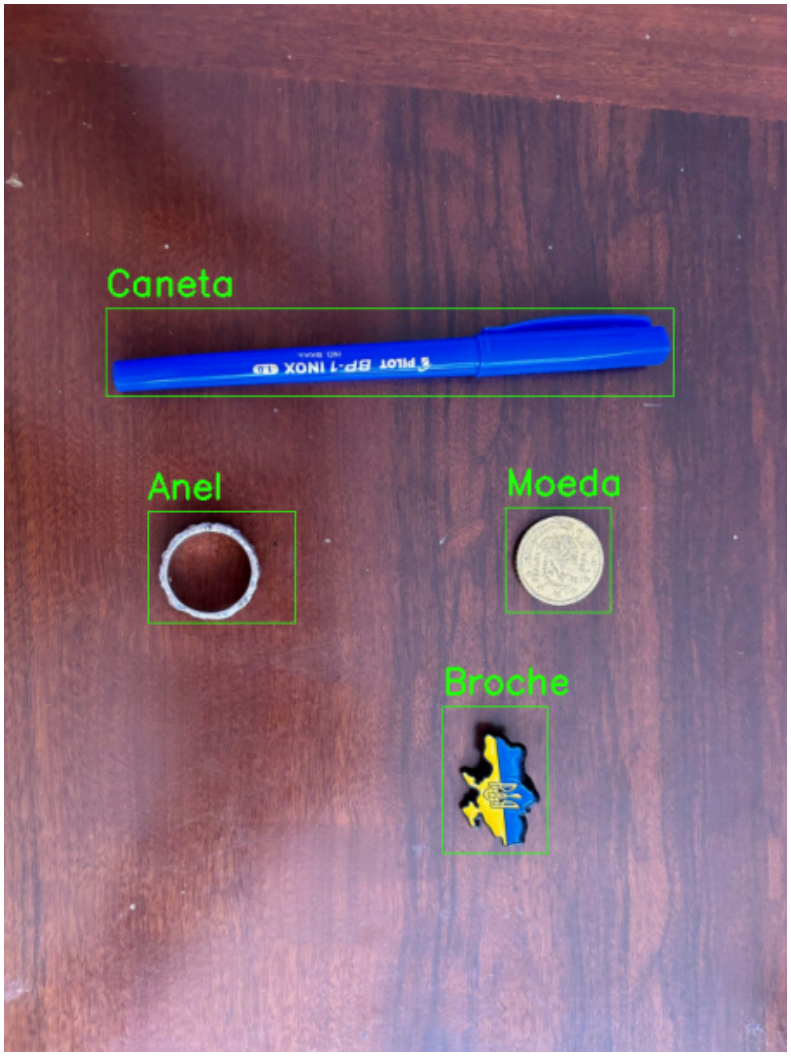
3. Use os dados de *X_train* e *y_train*, para o cálculo dos protótipos e *X_test e y_test* para o cálculo da acurácia;

4. Repita os passos 2 e 3, alterando o valor de random_state. Repita por uma quantidade grande de vezes;

5. Ao final, determine o modelo como o protótipo médio e mostre a acurácia média do classificador;

6. Com o uso do modelo final do protótipo médio, classifique as imagens que foram separadas no passo 1;

7. Ao final, com os dados das imagens classificadas, desenhe o retângulo envolvente com a classificação dos objetos. Como na imagem abaixo:



Com os dados do contorno é possível construir o retângulo envolvente acima com o seguinte código:

```
x, y, w, h = cv2.boundingRect(c)
cv2.rectangle(img_rgb,(x,y), (x+w,y+h), (36,255,12), 1)
cv2.putText(img_rgb, “Classe”, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX,
                                          0.9, (36,255,12), 2)
```

Entrega

Teremos na entrega:

- 3 arquivos de JN com cada uma das partes acima;
- Caso não tenha um script que gere automaticamente a nova pasta com as imagens manualmente selecionadas, faça upload das imagens selecionadas no github;
- Deverá ser entregue um relatório de 1 ou **no máximo** 2 páginas com a explicação do projeto, decisões que foram tomadas e resultados finais.

Ao terminar o EP, submeta o link do *GitHub* na atividade do EP3.

Cabeçalho para a primeira célula do JN:

```
# @title
### EP2 MAC0417 / MAC5768
#####################################################################
#  AO PREENCHER ESSE CABEÇALHO COM O MEU NOME E O MEU NÚMERO USP,#
#  DECLARO QUE SOU O ÚNICO AUTOR E RESPONSÁVEL PELA RESOLUÇÃO    #
#  DESTE EP.                                                      #
#  TODAS AS PARTES FORAM DESENVOLVIDAS E IMPLEMENTADAS POR MIM,   #
#  SEGUINDO AS INSTRUÇÕES E QUE PORTANTO, NÃO CONSTITUEM          #
#  DESONESTIDADE ACADÊMICA OU PLÁGIO.                             #
#                                                                 #
#  DECLARO TAMBÉM, QUE SOU RESPONSÁVEL POR TODAS AS CÓPIAS        #
#  DESSE PROGRAMA, E QUE EU NÃO DISTRIBUI OU FACILITEI A          #
#  SUA DISTRIBUIÇÃO. ESTOU CIENTE QUE OS CASOS DE PLÁGIO E        #
#  DESONESTIDADE ACADÊMICA SERÃO TRATADOS SEGUNDO OS CRITÉRIOS    #
#  DEFINIDOS NO CÓDIGO DE ÉTICA DA USP.                           #
#                                                                 #
#  ENTENDO QUE JUPYTER NOTEBOOKS SEM ASSINATURA NÃO SERÃO         #
#  CORRIGIDOS E, AINDA ASSIM, PODERÃO SER PUNIDOS POR             #
#  DESONESTIDADE ACADÊMICA.                                       #
#                                                                 #
#                                                                 #
#  Nome :                                                         #
#  NUSP :                                                         #
#  Turma:                                                         #
#  Prof.:                                                         #
#####################################################################
```