GrabCut para segmentação de Objetos

João Eduardo Santo Farias
Luiz Henrique Botega Cervantes

O que é segmentação

- Tem como objetivo dividir uma imagem em regiões ou objetos que a compõem.
- O nível de detalhe depende do objetivo a ser alcançado;
- É uma tarefa muito difícil na área de processamento de imagens;
- Várias estratégias podem ser utilizadas.

Introdução

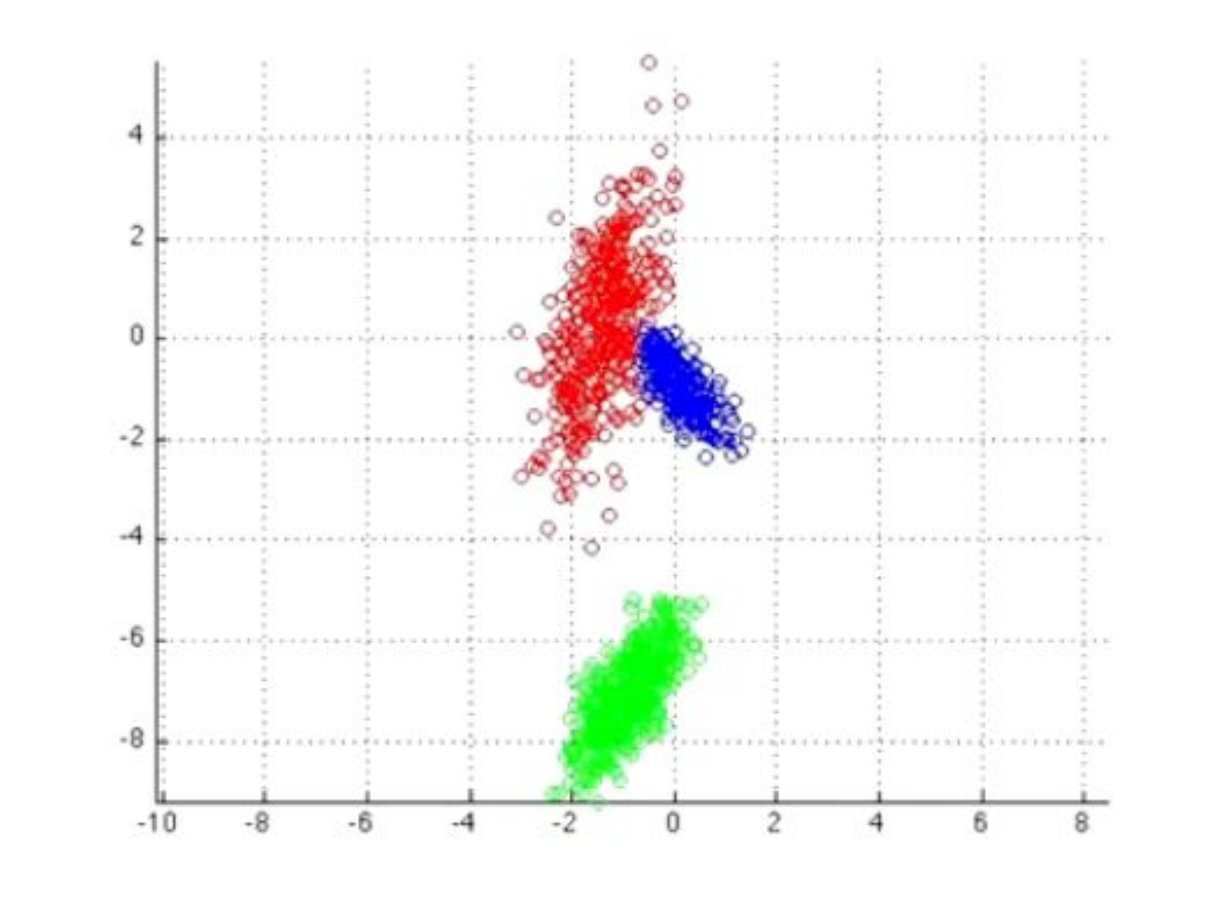
- Carsten Rother, Vladimir Kolmogorov e Andrew Blake;
- Extração do objeto.

FUNCIONAMENTO

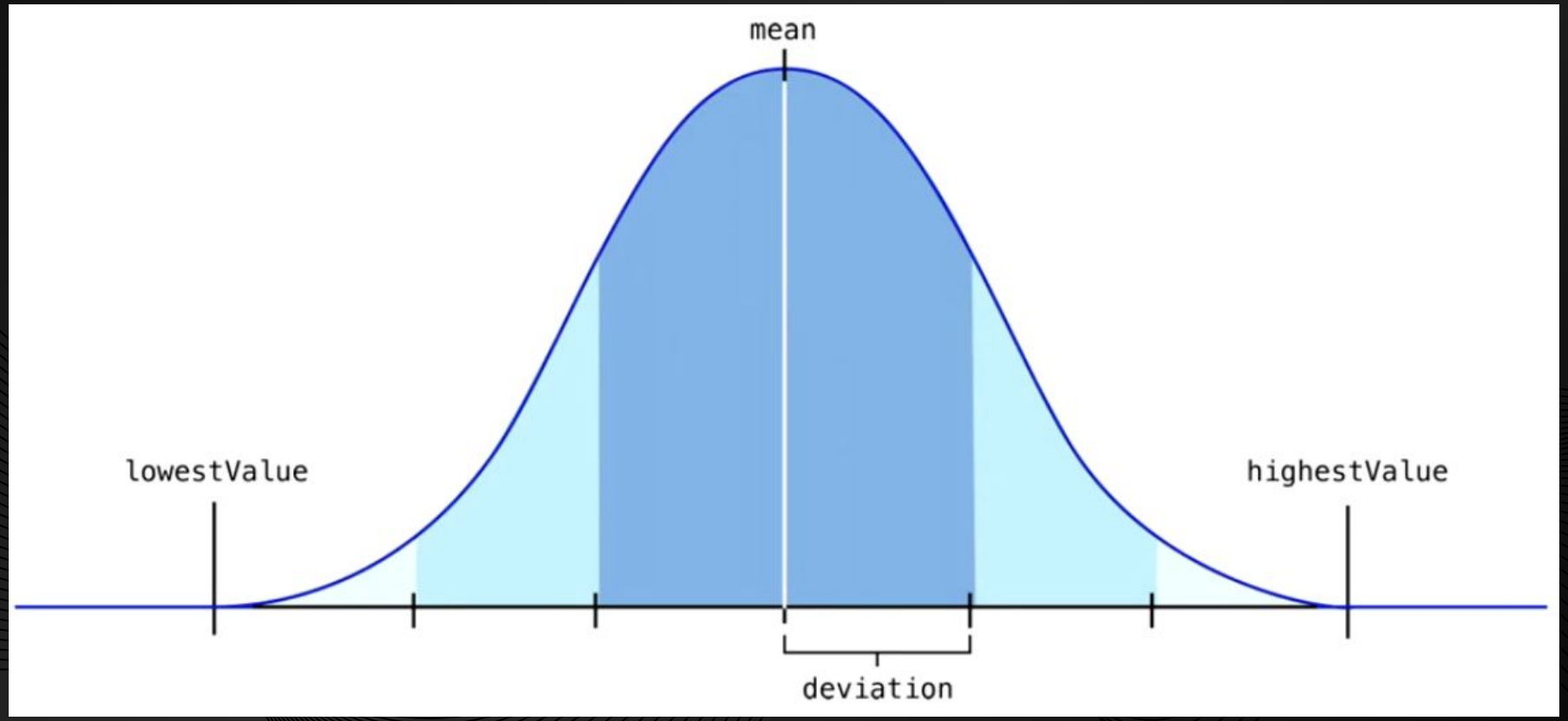
Início

- Seleção da área que o objeto se encontra;

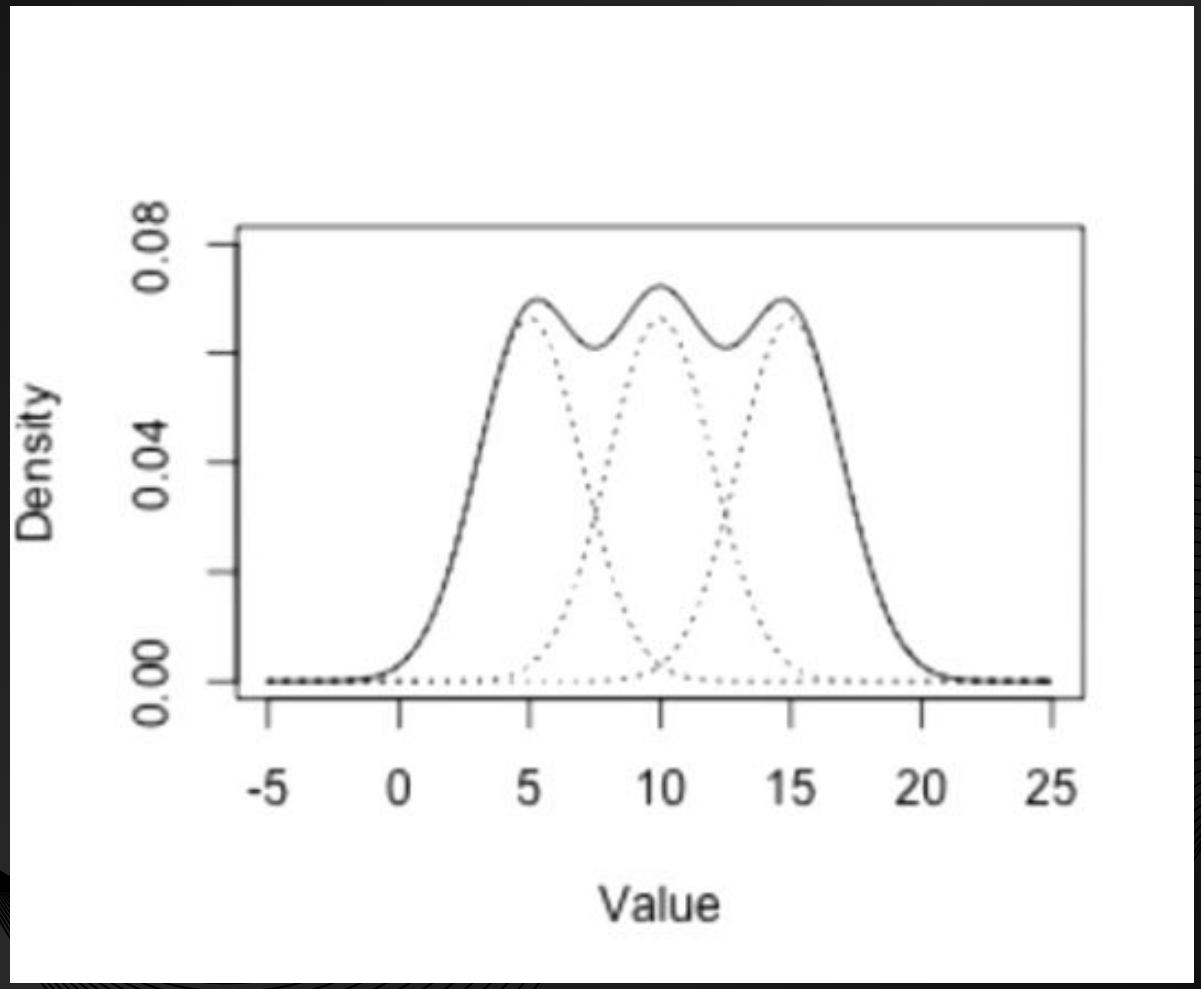
Modelo de mistura gaussiana



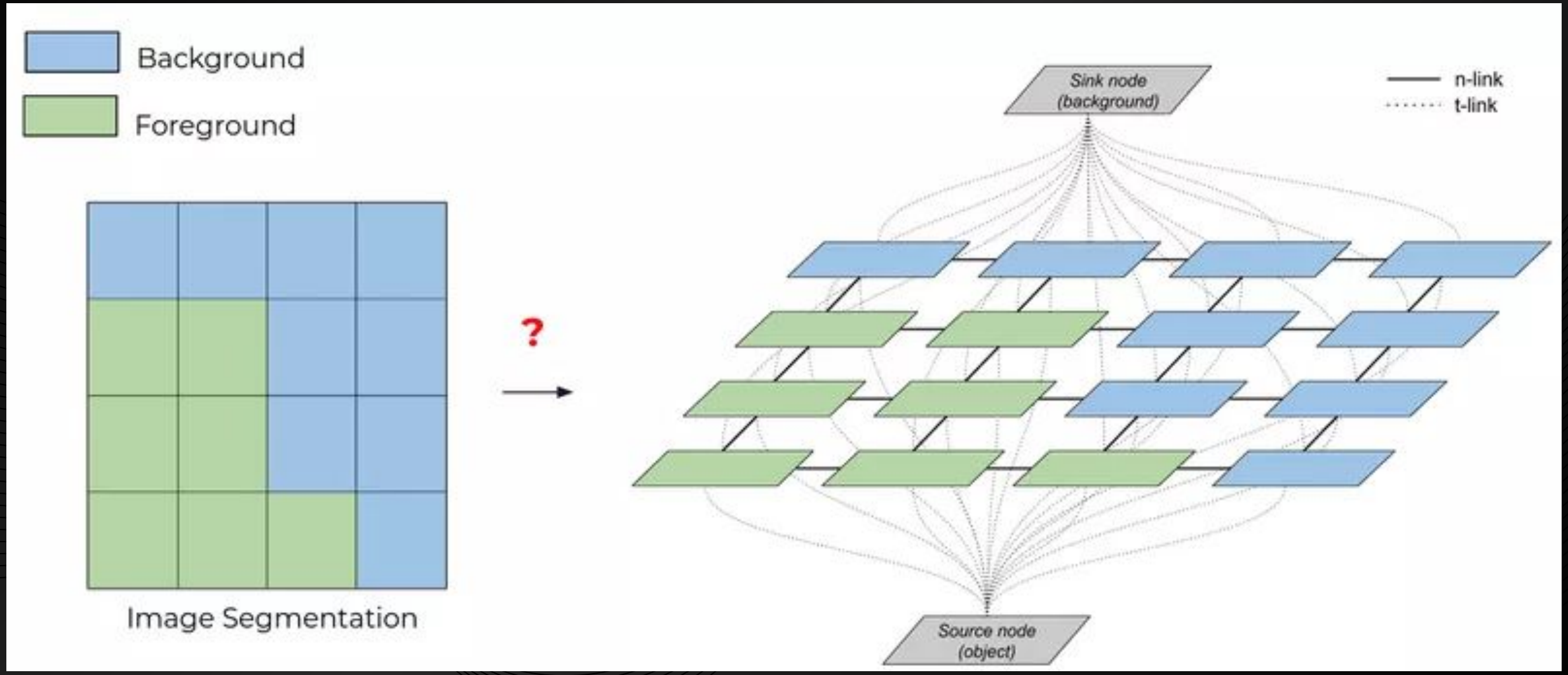
Modelo de mistura gaussiana



Modelo de mistura gaussiana



Construção do grafo



Termo de dados (V)

- Leva em consideração o peso n-link entre pixels cortados pela segmentação. Alpha representa os rótulos dos pixels e z representa as intensidades dos pixels.

$$V(\underline{\alpha}, \mathbf{z}) = \gamma \sum_{(m,n)\in \mathbf{C}} [\alpha_n \neq \alpha_m] \exp -\beta \|z_m - z_n\|^2$$

Termo de Suavidade (U)

- O termo de suavidade leva em consideração a modelagem da cor de fundo. Alpha representa os rótulos dos pixels, k e theta são parâmetros do Modelo de Mistura Gaussiana (GMM), e z representa as intensidades dos pixels.

$$U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) = \sum_{n} D(\alpha_n, k_n, \underline{\theta}, z_n)$$

MinCut

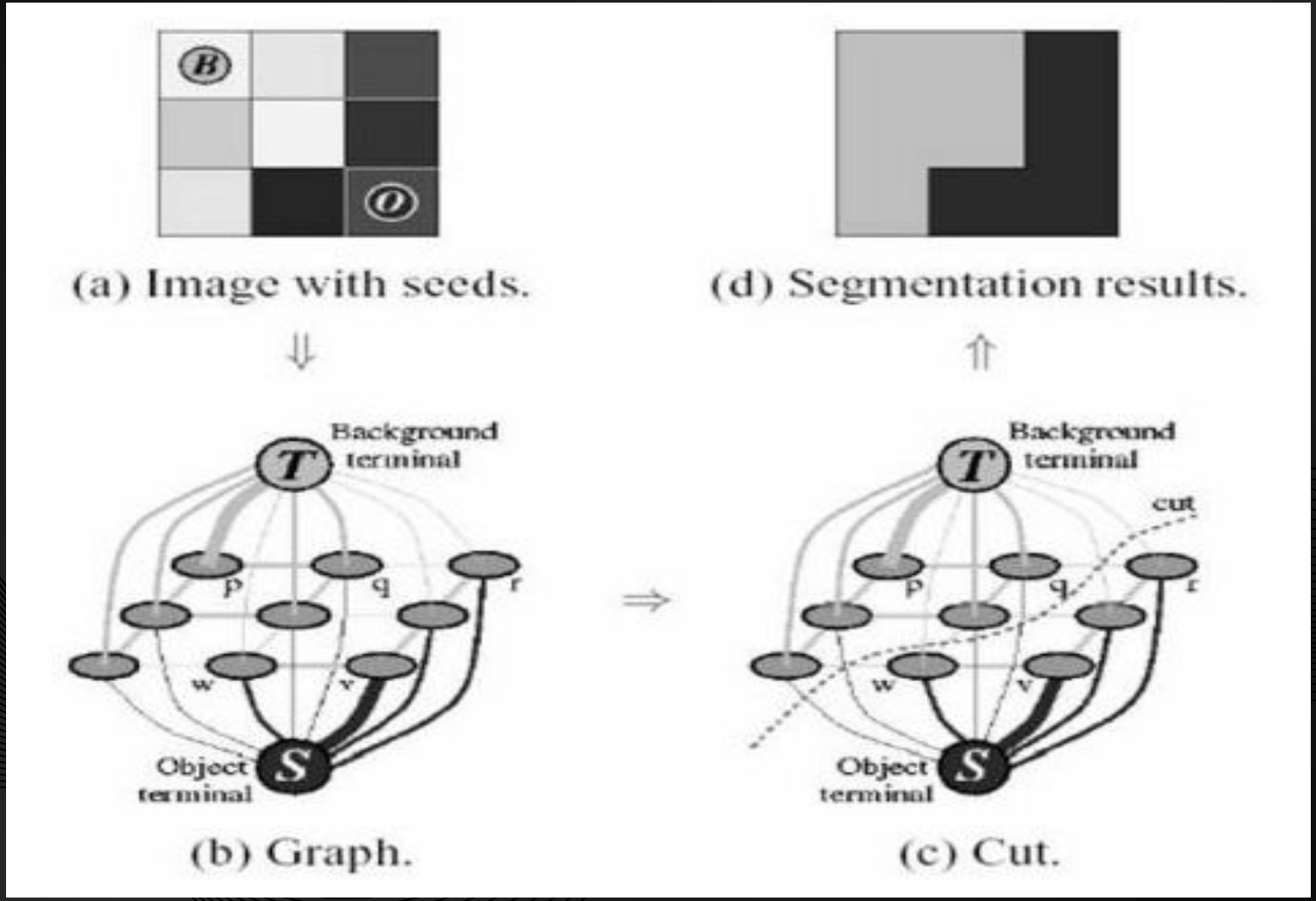
- Utilizado para separar os pixels que representam o primeiro plano dos que representam o plano de fundo, contribuindo assim para a segmentação da imagem.

MinCut

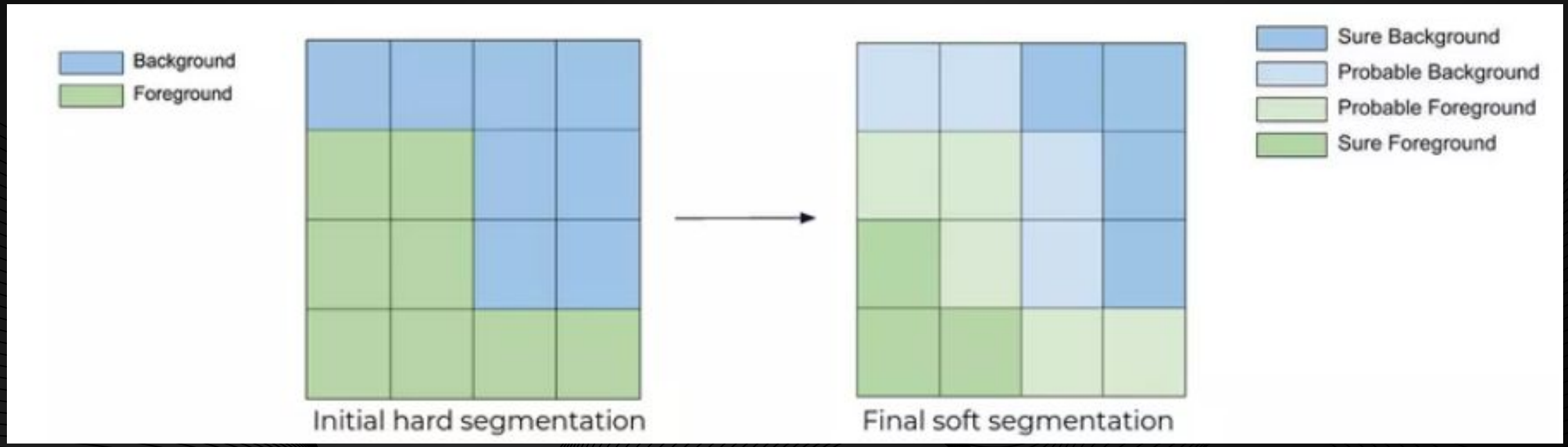
- Uma vez que os pesos são definidos, a função de custo ou função de energia é a soma desses pesos sobre o grafo:
- Alpha representa os rótulos dos pixels, k e theta são parâmetros dos Modelos de Mistura Gaussiana (GMM), e z representa as intensidades dos pixels.

$$\mathbf{E}(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) = U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) + V(\underline{\alpha}, \mathbf{z}),$$

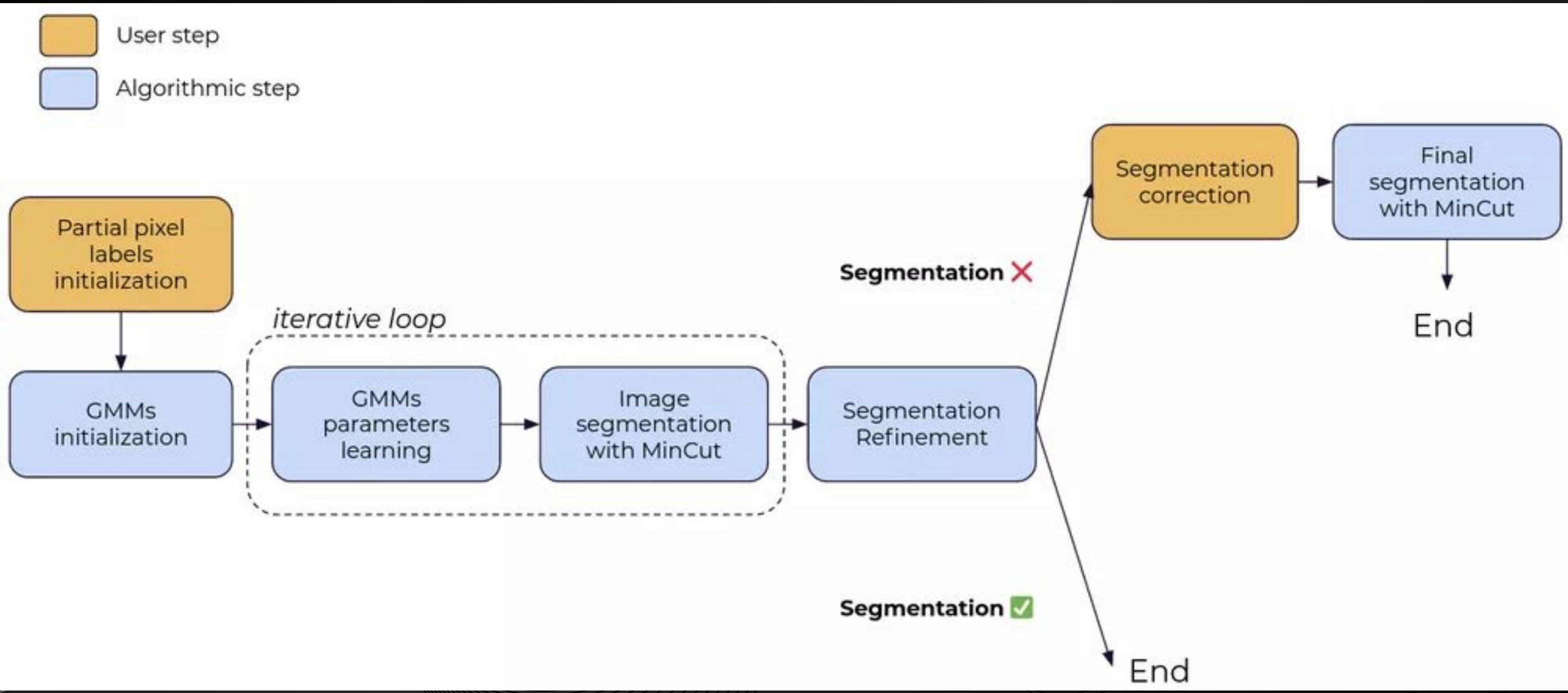
MinCut



Refinamento da segmentação

- 

Funcionamento



Primeira Implementação

Retângulo de seleção

Bibliotecas e caminho da imagem

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

#Caminho da imagem
image_path = r'C:\Users\hrick\Documents\VsCode\Python\PDI\segmentacao\objetos2.png'
```

Remoção do background

```python
def remove_background():

    #Carregar a imagem
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    #Inicializar a máscara como zeros
    mask = np.zeros(image.shape[:2], np.uint8)

    #Definir a região de interesse (ROI)
    rect = cv2.selectROI('Selecione a Região de Interesse', image)

    #Inicializar o modelo GrabCut
    bgd_model = np.zeros((1, 65), np.float64)
    fgd_model = np.zeros((1, 65), np.float64)
```

Remoção do background

```
#Aplicar o GrabCut
cv2.grabCut(image, mask, rect, bgd_model, fgd_model, 5, cv2.GC_INIT_WITH_RECT)

#Modificar a máscara para obter a segmentação final
mask2 = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')

#Aplicar a máscara à imagem original
segmented_image = image * mask2[:, :, np.newaxis]
```

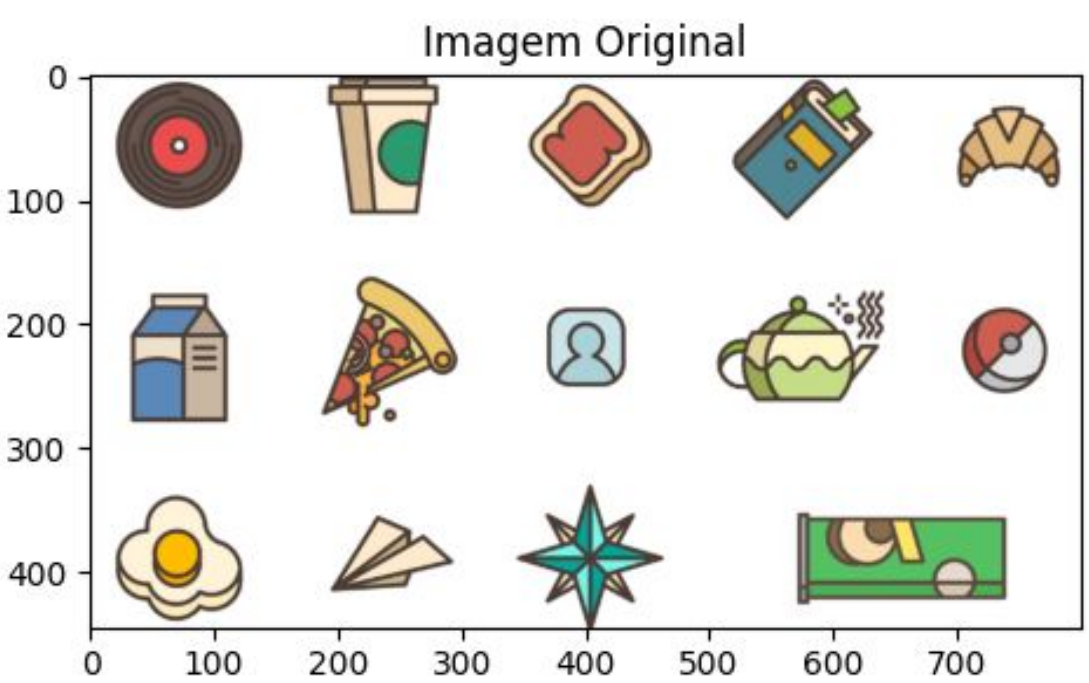
Remoção do background

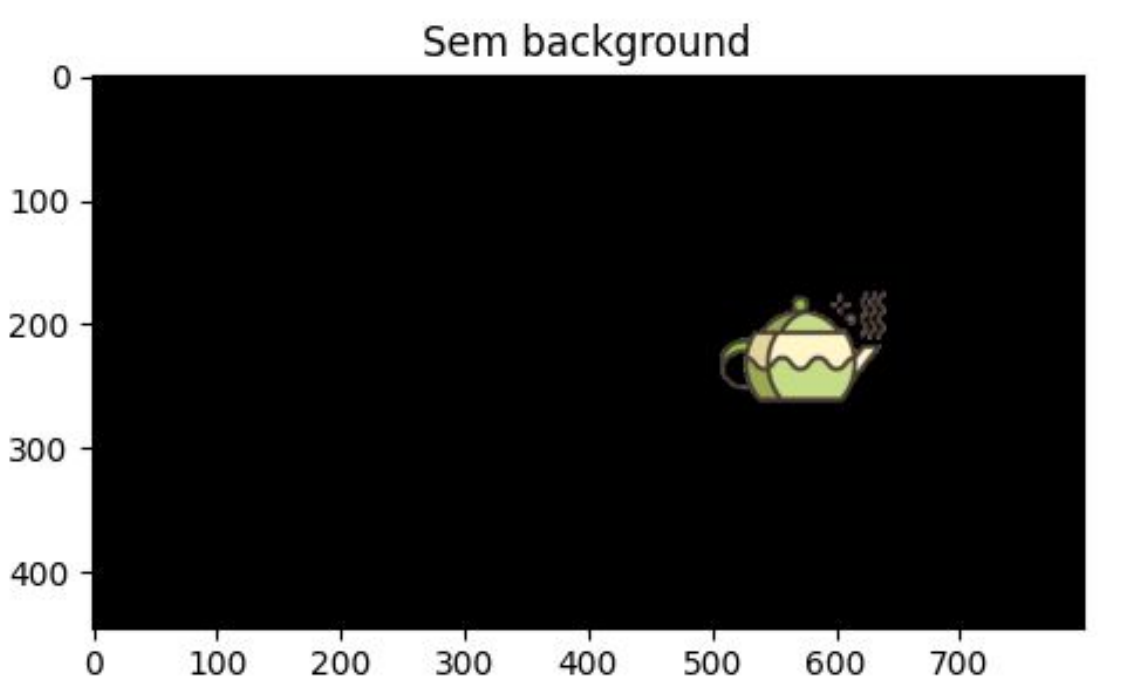
```
    #Plotar imagem original
    plt.subplot(1, 2, 1)
    plt.imshow(image)
    plt.title('Imagem Original')

    #Plotar sem o background
    plt.subplot(1, 2, 2)
    plt.imshow(segmented_image)
    plt.title('Sem background')

    plt.show()

remove_background()
```

Primeira imagem

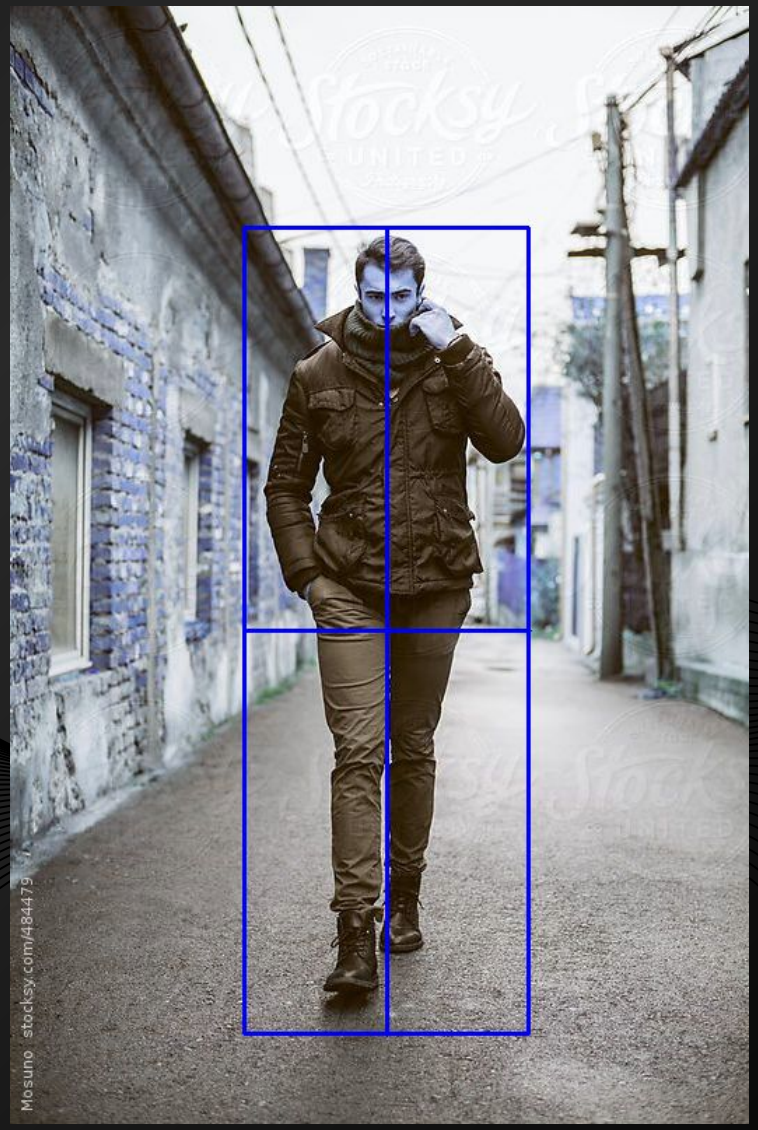
Primeira imagem

Primeira imagem





Segunda imagem



Segunda imagem

Segunda Implementação

Retângulo de seleção + correção manual

Definição das variáveis

```
#Cor do retangulo de seleção
BLUE = [255, 0, 0]

#Background
RED = [0, 0, 255]
DRAW_BG = {'color': RED, 'val': 0}

#Foreground
GREEN = [0, 255, 0]
DRAW_FG = {'color': GREEN, 'val': 1}

#Flags
rect = (0, 0, 1, 1)
drawing = False             # flag para desenhar as curvas
rectangle = False           # flag para desenhar o retângulo
rect_over = False           # flag para verificar se o retangulo ficou completo
rect_or_mask = 100          # flag selecionar entre rect ou mask
value = DRAW_FG             # atribuir o valor do DRAW_FG
thickness = 3               # espessura do pincel
```

Ações do mouse

```python
#Ações do mouse
def onmouse(event, x, y, flags, param):
    global img, img2, drawing, value, mask, rectangle, rect, rect_or_mask, ix, iy, rect_over

    #Desenhar o retangulo
    if event == cv2.EVENT_RBUTTONDOWN:
        rectangle = True
        ix, iy = x, y
    elif event == cv2.EVENT_MOUSEMOVE:
        if rectangle == True:
            img = img2.copy()
            cv2.rectangle(img, (ix, iy), (x, y), BLUE, 2)
            rect = (min(ix, x), min(iy, y), abs(ix - x), abs(iy - y))
            rect_or_mask = 0
    elif event == cv2.EVENT_RBUTTONUP:
        rectangle = False
        rect_over = True
        cv2.rectangle(img, (ix, iy), (x, y), BLUE, 2)
        rect = (min(ix, x), min(iy, y), abs(ix - x), abs(iy - y))
        rect_or_mask = 0
        print(" Aperte 'n' para ver o resultado \n")
```

Ações do mouse

```
#Desenhar com o pincel
if event == cv2.EVENT_LBUTTONDOWN:
    if rect_over == False:
        print("Desenhe o retângulo primeiro \n")
    else:
        drawing = True
        cv2.circle(img, (x, y), thickness, value['color'], -1)
        cv2.circle(mask, (x, y), thickness, value['val'], -1)
elif event == cv2.EVENT_MOUSEMOVE:
    if drawing == True:
        cv2.circle(img, (x, y), thickness, value['color'], -1)
        cv2.circle(mask, (x, y), thickness, value['val'], -1)
elif event == cv2.EVENT_LBUTTONUP:
    if drawing == True:
        drawing = False
        cv2.circle(img, (x, y), thickness, value['color'], -1)
        cv2.circle(mask, (x, y), thickness, value['val'], -1)
```

Imagens

```
#Pega a imagem do diretório
img = cv2.imread(r'C:\Users\hrick\Documents\VsCode\Python\PDI\segmentacao\carro_esportivo.jpeg')

#Faz uma copia da imagem original
img2 = img.copy()

#Cria uma matriz bidimensional preenchida com zeros
mask = np.zeros(img.shape[:2], dtype=np.uint8)

#Output image to be shown
output = np.zeros(img.shape, np.uint8)

#Crias as janelas
cv2.namedWindow('output')
cv2.namedWindow('input')
cv2.setMouseCallback('input', onmouse)
cv2.moveWindow('input', img.shape[1] + 10, 90)
```

Instruções

```
print(" Instruções: \n")
print(" Selecione a área com o botão direito do mouse \n")
while True:
    cv2.imshow('output', output)
    cv2.imshow('input', img)

    #Aguarda receber um valor do teclado
    k = 0xFF & cv2.waitKey(1)

    # Atalhos para controlar o menu
    if k == 27:          #Esc para sair
        break
    elif k == ord('0'):  #Desenhar o plano de fundo
        print(" Marque as regiões a serem retiradas com o botão esquerdo \n")
        value = DRAW_BG
    elif k == ord('1'):  #Desenhar a imagem a ser mantida
        print(" Marque as regiões a serem mantidas com o botão esquerdo \n")
        value = DRAW_FG
```

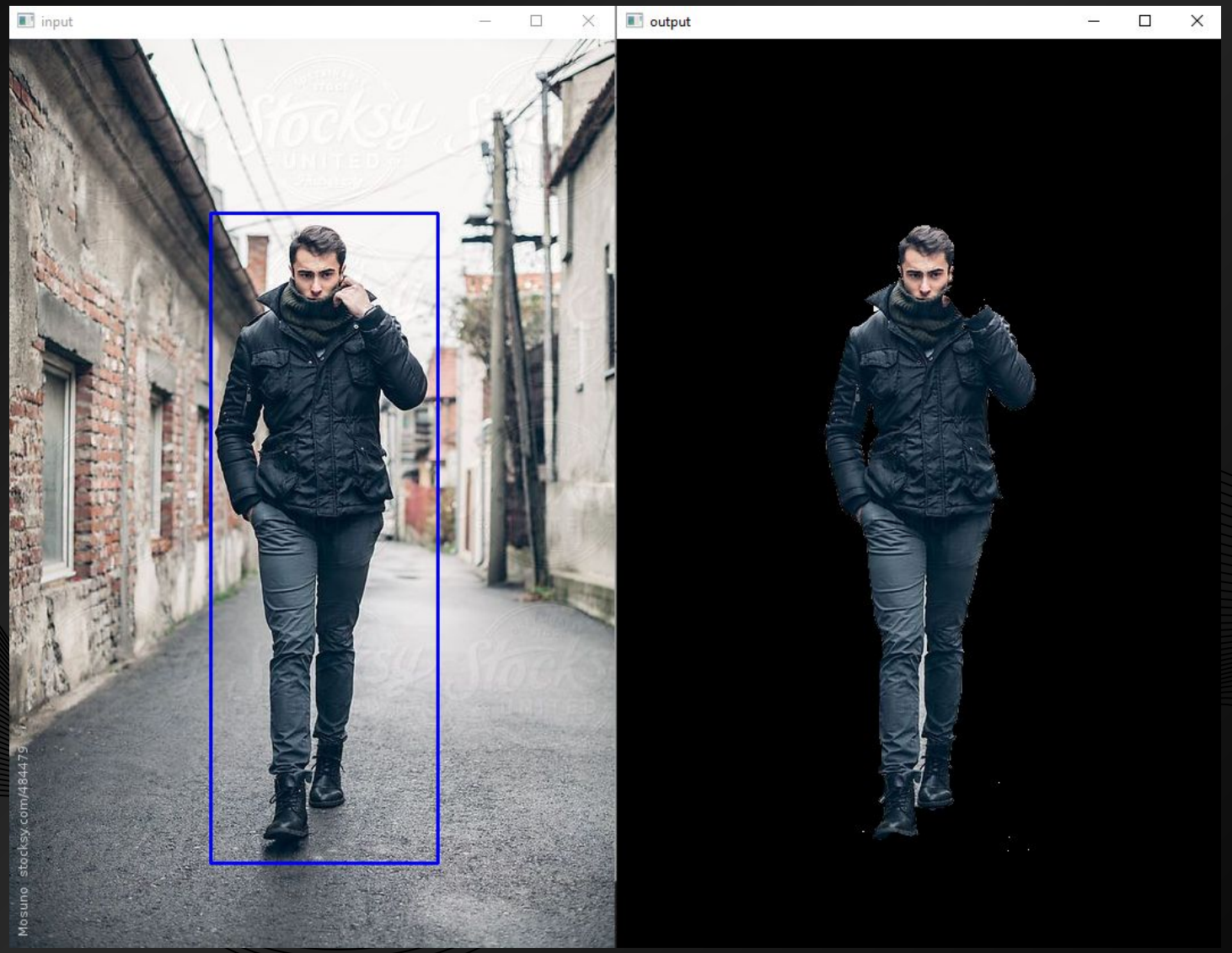
Instruções

```
elif k == ord('r'):  #Reseta para as configurações padrão
    print(" Resetando \n")
    rect = (0, 0, 1, 1)
    drawing = False
    rectangle = False
    rect_or_mask = 100
    rect_over = False
    value = DRAW_FG
    img = img2.copy()
    mask = np.zeros(img.shape[:2], dtype=np.uint8)
    output = np.zeros(img.shape, np.uint8)
```

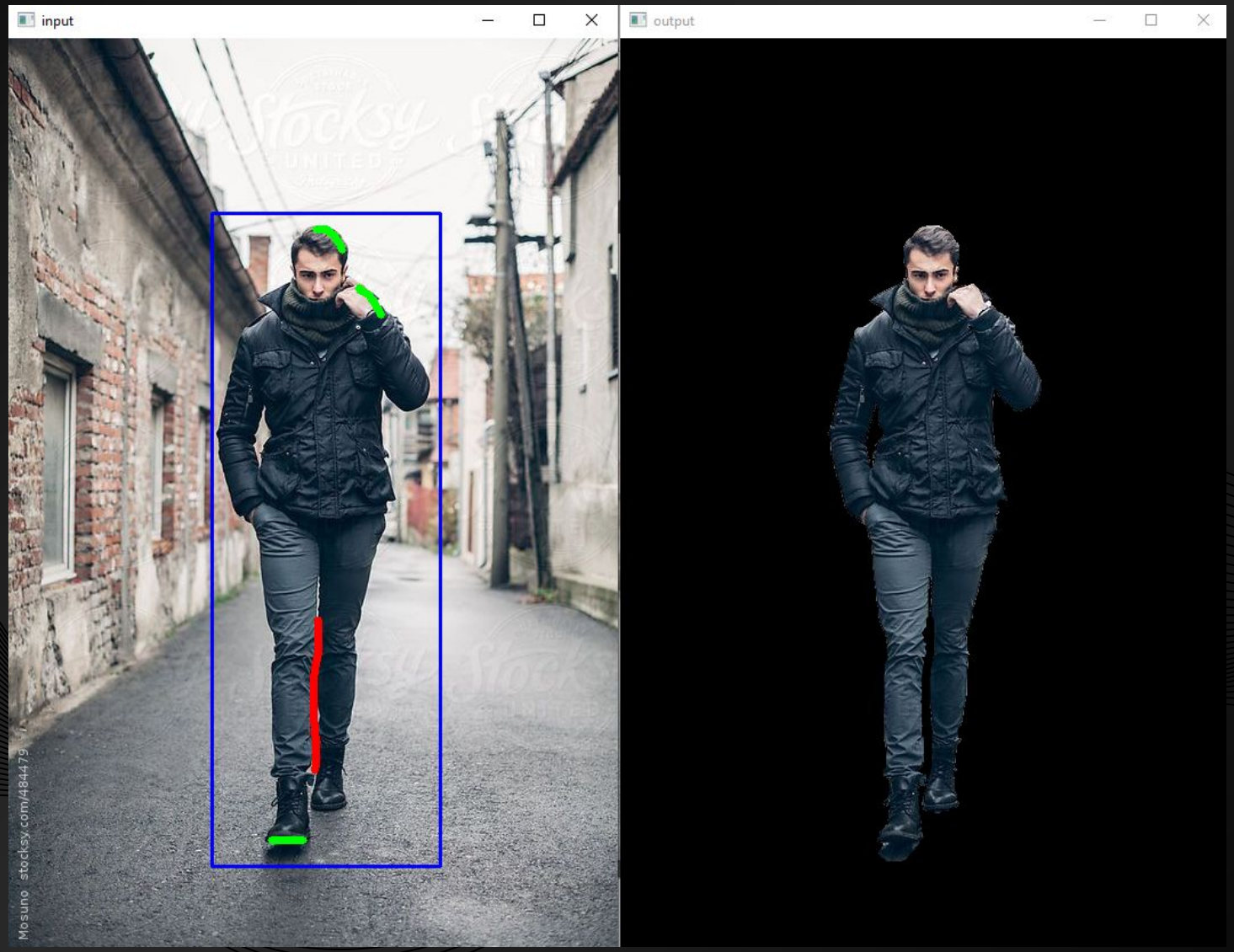
Instruções

```
    elif k == ord('n'):          #Roda o GrabCut
        print(" Marque os planos utilizando 0 ou 1 e a seguir pressione 'n' \n")
        if rect_or_mask == 0:    #GrabCut com retângulo
            bgdmodel = np.zeros((1, 65), np.float64)
            fgdmodel = np.zeros((1, 65), np.float64)
            cv2.grabCut(img2, mask, rect, bgdmodel, fgdmodel, 1, cv2.GC_INIT_WITH_RECT)
            rect_or_mask = 1
        elif rect_or_mask == 1: #GrabCut com a mascara
            bgdmodel = np.zeros((1, 65), np.float64)
            fgdmodel = np.zeros((1, 65), np.float64)
            cv2.grabCut(img2, mask, rect, bgdmodel, fgdmodel, 1, cv2.GC_INIT_WITH_MASK)

    mask2 = np.where((mask == 1) + (mask == 3), 255, 0).astype('uint8')
    output = cv2.bitwise_and(img2, img2, mask=mask2)

cv2.destroyAllWindows()
```

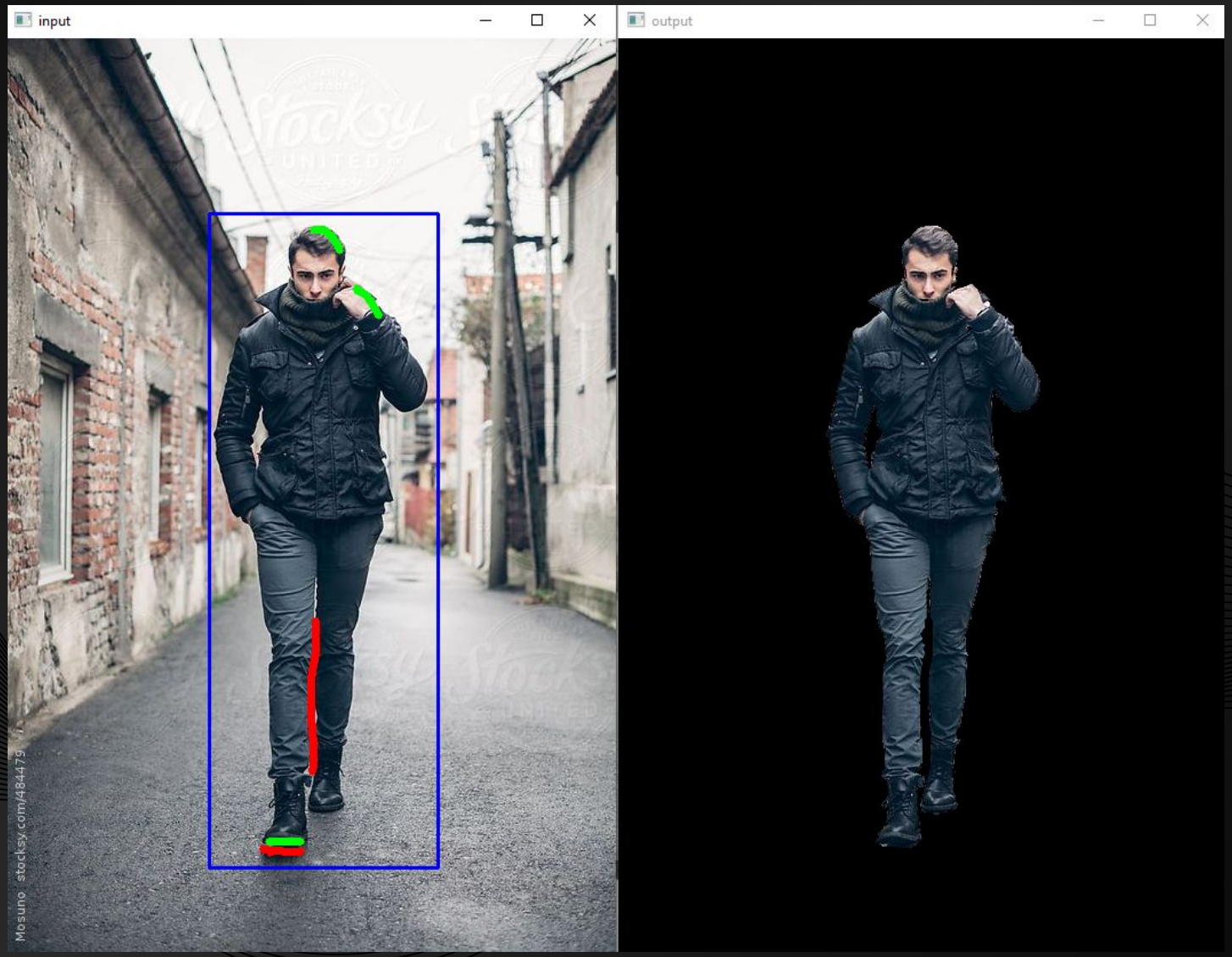
Primeira imagem



Primeira imagem



Primeira imagem

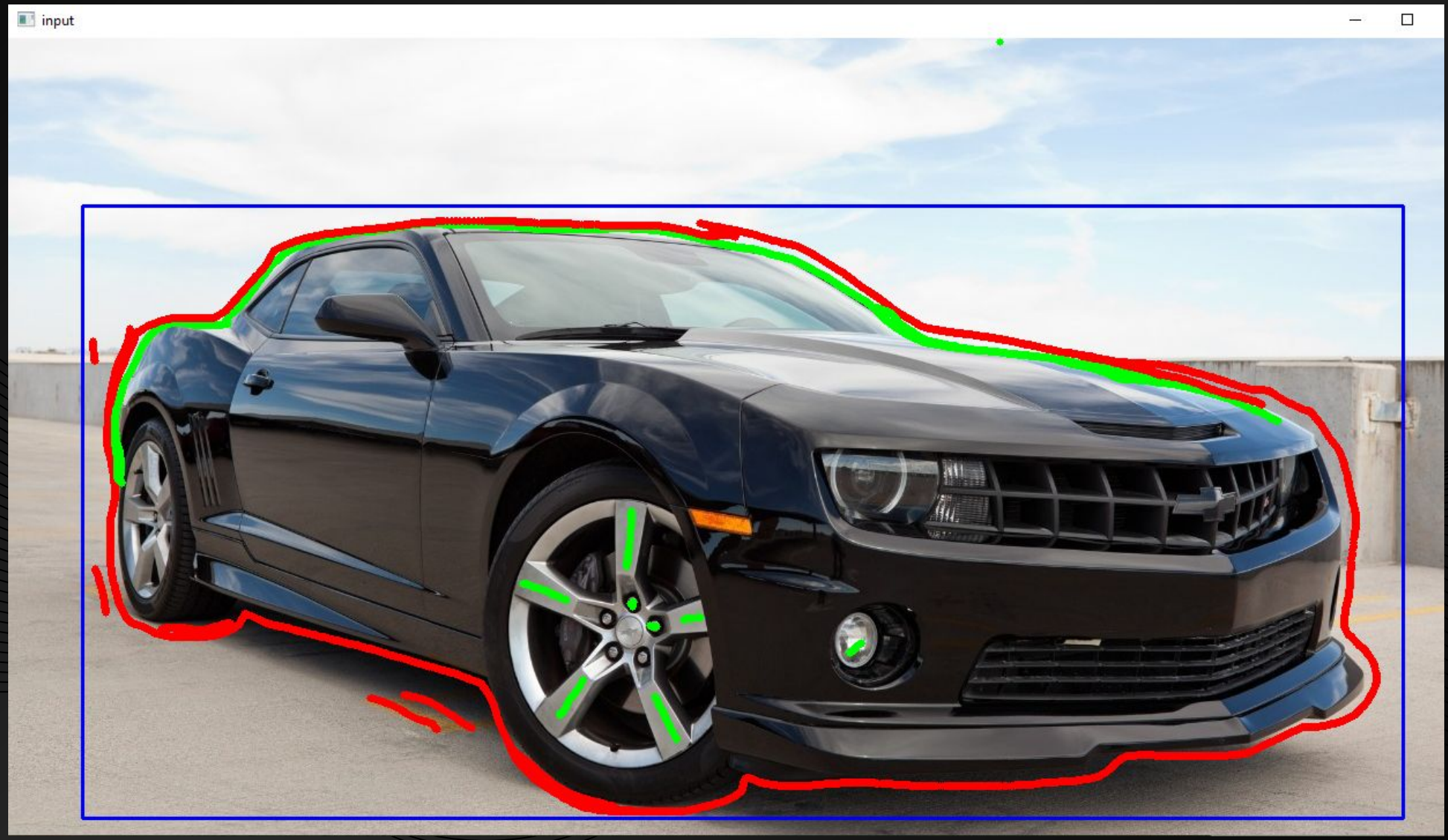


Segunda imagem

Segunda imagem

Segunda imagem

Segunda imagem



Segunda imagem

Referências

Visão Computacional. Identificação, Detecção, Reconhecimento e Segmentação de Imagem e Objetos. Disponível em: <https://visaocomputacional.com.br/identificacao-deteccao-reconhecimento-e-segmentacao-de-imagem-e-objetos/>. Acesso em 1 de dezembro de 2023.

PROST, Julie. GrabCut for Automatic Image Segmentation [OpenCV Tutorial]. Disponível em: <https://www.sicara.fr/blog-technique/grabcut-for-automatic-image-segmentation-opencv-tutorial>. Acesso em 1 de dezembro de 2023.