Visão Computacional

Segmentação de Objetos

Professor Ewerson Eduardo Martins

Segmentação de Objetos

A segmentação de objetos em imagem é uma das principais etapas de um sistema baseado em Visão Computacional. Essa etapa consiste em separar somente a área que representa o objeto de interesse em uma nova imagem, excluindo também o segundo plano dessa região. O objeto a ser estudado é considerado o primeiro plano da imagem. Os pixels que não fazem parte de sua representação são denominados como segundo plano. A segmentação é o primeiro passo para que possamos extrair características do objeto. Somente com ele segmentado, é possível obter informações e detalhes que tornam possível classificá-lo.

Segmentação de Objetos

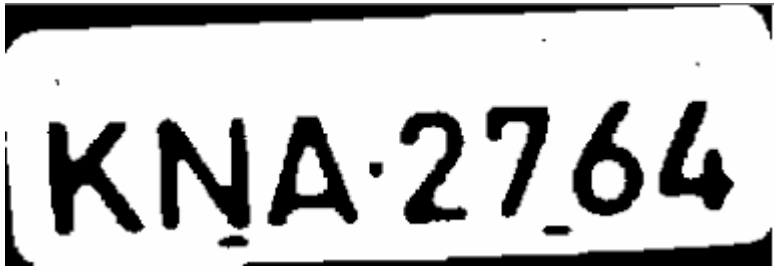
Segmentação por Binarização

A segmentação por binarização também é conhecida como aplicação de limiar de intensidade. A separação do objeto de interesse por meio desse método ocorre pela definição do valor de um limiar.

Segmentação de Objetos

Segmentação por Binarização

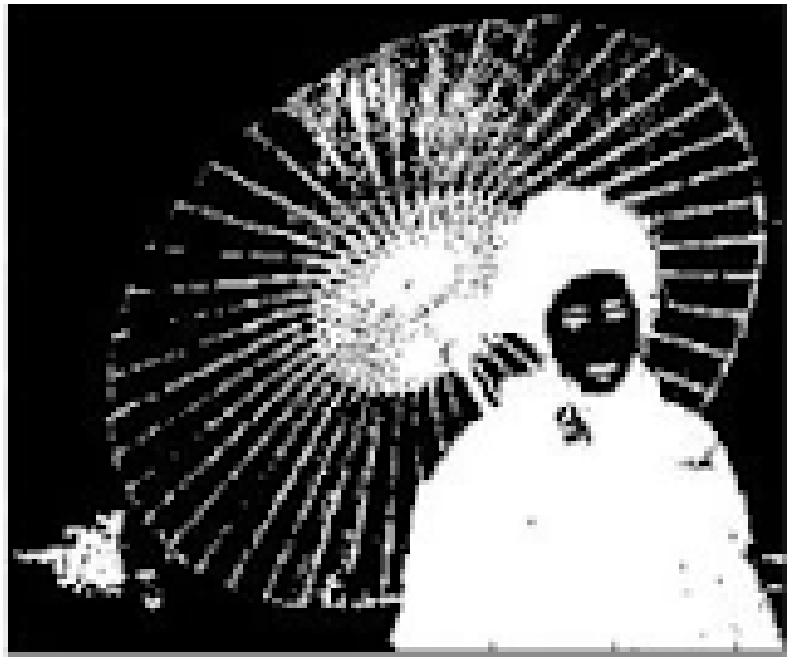
Quando uma imagem é submetida a esse procedimento, os pixels representados por valores maiores que o limiar são estabelecidos como o objeto de interesse, sendo redefinidos para a cor preta ou branca. Do contrário, os pixels representados por valores menores que o limiar são estabelecidos como o segundo plano, sendo redefinidos para a cor oposta à do objeto de interesse.



Segmentação de Objetos

Segmentação por Binarização

A segmentação por binarização resulta em uma imagem binária, na qual geralmente o objeto de interesse é representado pela cor branca e o segundo plano pela preta



Segmentação de Objetos

Segmentação por Binarização

Com a biblioteca OpenCV, essa função requer quatro parâmetros obrigatórios. O primeiro deles é a imagem em tons de cinza que receberá o tratamento, e o segundo é o valor do limiar, geralmente definido por tentativa e erro. O terceiro parâmetro define o valor da intensidade que receberá os pixels representados por valores superiores ao limiar e o quarto indica o método de aplicação do limiar.

Segmentação de Objetos

Segmentação por Binarização

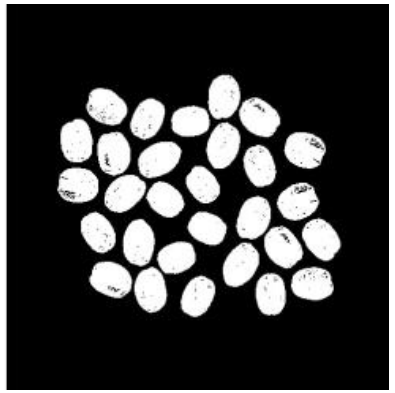
A tabela a seguir apresenta os dois métodos mais utilizados

Parâmetro	Descrição
THRESH_BINARY	Objeto de interesse na cor preta
THRESH_BINARY_INV	Objeto de interesse na cor branca

Segmentação de Objetos

Segmentação por Binarização

O valor 135 foi obtido através de tentativa e erro e definido como limiar.



Exemplo

```
import cv2
import numpy as np

imgOriginal = cv2.imread("graos-de-cafe.jpeg", 0)

metodo = cv2.THRESH_BINARY_INV
ret, imgBinarizada = cv2.threshold(imgOriginal, 135, 255, metodo)

cv2.imshow("Imagem Original", imgOriginal)
cv2.imshow("Imagem Tratada", imgBinarizada)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Segmentação de Objetos

Segmentação por Binarização

Nessa figura os grãos de café estão falhadas, essas falhas são provocadas pelo procedimento de binarização e podem ser facilmente tratadas com operações morfológicas.

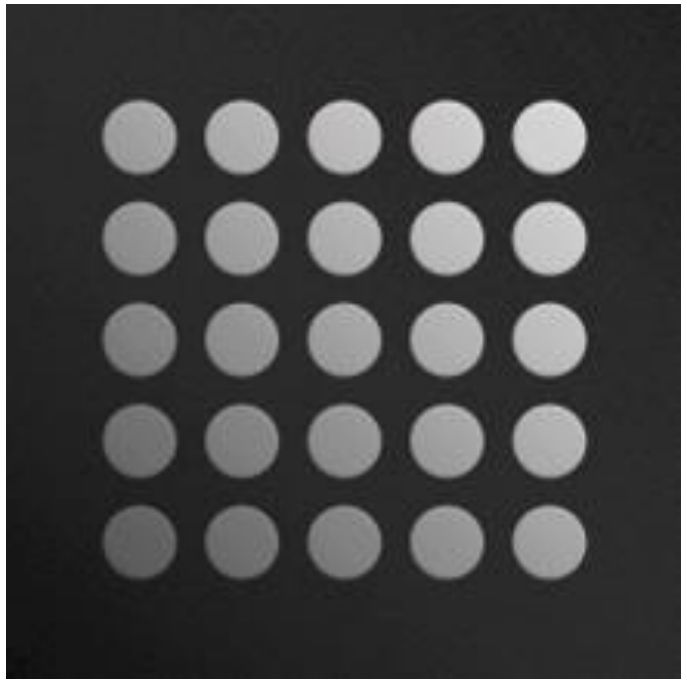
Exemplo de operação de morfologia

```
import cv2
import numpy as np

imagemOriginal = cv2.imread("rolamento-ruido-interno.bmp", 0)

elementoEstruturante = cv2.getStructuringElement(
  cv2.MORPH_ELLIPSE, (5,5)
)
imagemProcessada = cv2.morphologyEx(
  imagemOriginal, cv2.MORPH_CLOSE, elementoEstruturante
)

cv2.imshow("Original", imagemOriginal)
cv2.imshow("Resultado", imagemProcessada)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Segmentação de Objetos

Binarização Adaptativa

Na binarização da imagem dos grãos de café, um valor global foi definido para o limiar de binarização. Em outras palavras, um mesmo valor foi utilizado como parâmetro para tratar cada pixel da imagem. Esse procedimento apresenta bons resultados quando a fotografia está uniformemente iluminada, o que não ocorre na imagem ao lado nessas situações , o algoritmo de “binarização adaptativa” apresenta bons resultados.



Segmentação de Objetos

Binarização Adaptativa

A biblioteca OpenCV possui a função adaptiveThreshold , que nos permite aplicar um algoritmo de binarização adaptativa a uma imagem. Para utilizar essa função, seis parâmetros devem ser informados conforme indicados na tabela.

Parâmetro	Descrição
src	Matriz referente à imagem.
maxValue	Valor de intensidade máxima do pixel.
adaptiveMethod	Algoritmo de binarização adaptativa.
thresholdType	Tipo de binarização
blockSize	Tamanho da máscara.
C	Constante de subtração da média ou da média ponderada.

Segmentação de Objetos

Binarização Adaptativa

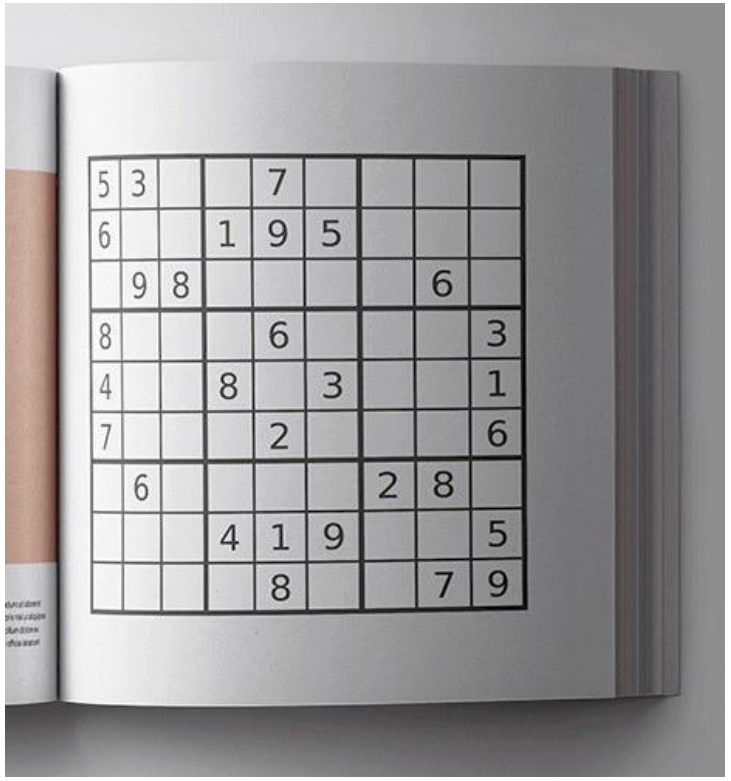
Com tantos parâmetros e uma infinidade de combinações possíveis para defini-los, ficar em dúvida sobre qual a melhor combinação de valores a ser utilizada é uma complicação frequente. Mesmo assim, não há regras ou dicas que possam ser aplicadas. Esses valores devem ser definidos, geralmente por tentativa e erro, para solucionar cada problema específico.

```
import cv2
import numpy as np

imgOriginal = cv2.imread("comprimidos.jpeg", 0)
imgTratada = cv2.medianBlur(imgOriginal, 7)

imgBinarizada = cv2.adaptiveThreshold(
  imgTratada, 255,
  cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
  cv2.THRESH_BINARY_INV, 11, 5
)

cv2.imshow("Imagem Original", imgOriginal)
cv2.imshow("Imagem Tratada", imgBinarizada)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Segmentação de Objetos

Binarização Adaptativa

Uma característica desfavorável desse processo, é que ele pode gerar imagens onde os objetos de interesse nem sempre estão devidamente preenchidos. A binarização adaptativa apresenta bons resultados, por exemplo, quando aplicada a imagens com texto.



Professor Ewerson Eduardo Martins

Segmentação de Objetos

Binarização de Nobuyuki Otsu

Acabamos de estudar a função threshold da bibliteca OpenCV (para binarização global de imagens). Diferente da adaptiveThreshold , a função threshold binariza cada pixel de uma imagem a partir de um limiar predefinido, sem considerar os pixels vizinhos ao pixel-alvo. O grande contratempo dessa função é que um limiar precisa ser definido pelo usuário. A fim de automatizar esse procedimento, um algoritmo desenvolvido por **Nobuyuki Otsu** define um limiar baseado no histograma da imagem.

```
import cv2
import numpy as np

imgOriginal = cv2.imread("graos-de-cafe.jpeg", 0)

tipo = cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU
limiar, imgBinarizada = cv2.threshold(imgOriginal, 0, 255, tipo)

print(limiar)

cv2.imshow("Imagem Original", imgOriginal)
cv2.imshow("Imagem Tratada", imgBinarizada)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Segmentação de Objetos

Segmentação por Cor

O procedimento de segmentação por cor nos permite separar objetos de interesse representados por uma cor específica na imagem. Para facilitar essa tarefa, o espaço de cor HSV é quase sempre utilizado, justamente por representar as informações referentes à cor (matiz) em um único canal.

Segmentação de Objetos

Segmentação por Cor

O método inRange da biblioteca OpenCV pode ser usado para executar esse procedimento. Ele requer três parâmetros obrigatórios: o primeiro é a imagem representada no espaço HSV; o segundo, um vetor contendo os valores referentes ao limite inferior da cor desejada, representada também no espaço HSV; e o terceiro, assim como o segundo, também é um vetor, entretanto, contém os valores referentes ao limite superior da cor desejada.

Segmentação de Objetos

Segmentação por Cor

Cor	Limite inferior	Limite superior
Amarelo	10, 100, 100	50, 255, 255
Azul	100, 100, 100	140, 255, 255
Verde	40, 100, 100	80, 255, 255
Vermelho	160, 100, 100	200, 255, 255

O limite inferior corresponde à tonalidade mais clara da cor do objeto de interesse, já o limite superior corresponde à mais forte ou escura. Para obter esses valores, a tabela a seguir pode ser consultada, ou, então, basta converter uma determinada cor do espaço RGB para o HSV.

Segmentação de Objetos

Segmentação por Cor

Para obter os valores de uma cor do espaço RGB no espaço HSV, podemos executar o procedimento exemplificado pelo código ao lado.

```
import cv2
import numpy as np

verdeRGB = np.uint8([[[0,255,0]]])
verdeHSV = cv2.cvtColor(verdeRGB, cv2.COLOR_BGR2HSV)

print verdeHSV
```

Resultado:

```
[[[60 255 255]]
```

Segmentação de Objetos

Segmentação por Cor

O vetor retornado pela execução do código apresenta os valores que representam respectivamente os canais de matiz, a saturação e o valor da cor no espaço HSV. Para obter o limite superior e inferior, basta somar +20 e subtrair -20 ao valor do primeiro canal.

```
# Limite inferior:
# 60 - 20 = 40
tomClaro = np.array([40 100 100])

# Limite superior:
# 60 + 20 = 80
tomEscuro = np.array([80 255 255])
```

Segmentação de Objetos

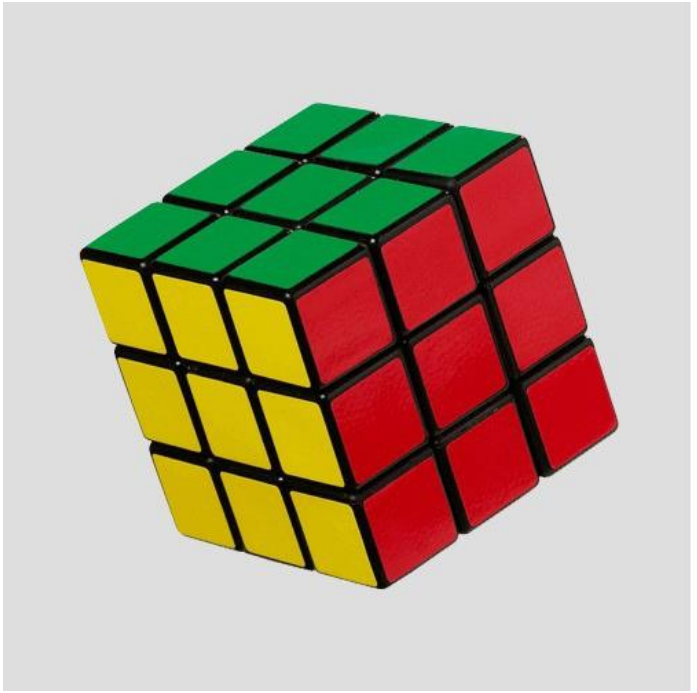
Segmentação por Cor

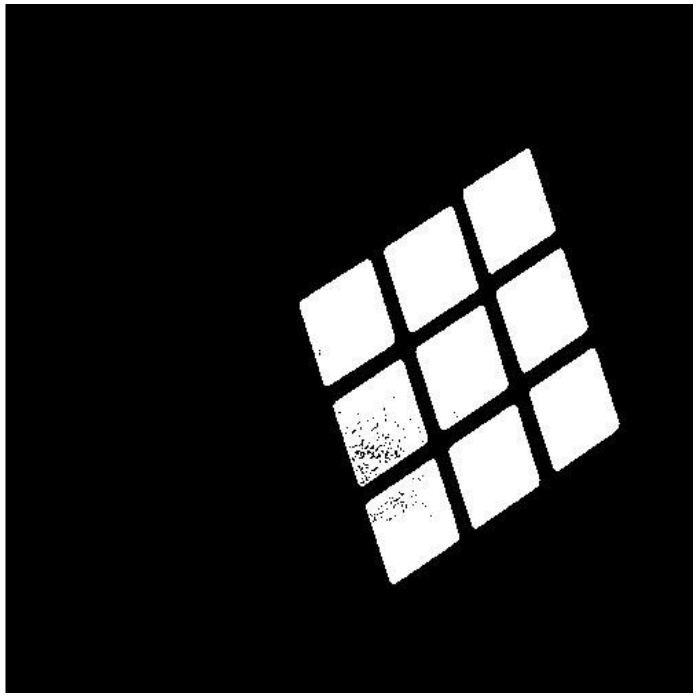
O código exemplifica como segmentar objetos coloridos em uma imagem com o método inRange . Nesse exemplo, as faces vermelhas de um cubo mágico são segmentadas em uma imagem binária.

```
import cv2
import numpy as np

imgRGB = cv2.imread("cubo-magico.jpeg")
imgHSV = cv2.cvtColor(imgRGB, cv2.COLOR_BGR2HSV)

tomClaro = np.array([160, 100, 100])
tomEscuro = np.array([200, 255, 255])

imgSegmentada = cv2.inRange(imgHSV, tomClaro, tomEscuro)

cv2.imshow("Original", imgRGB)
cv2.imshow("Segmentada",  imgSegmentada)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Segmentação de Objetos

Segmentação por Cor

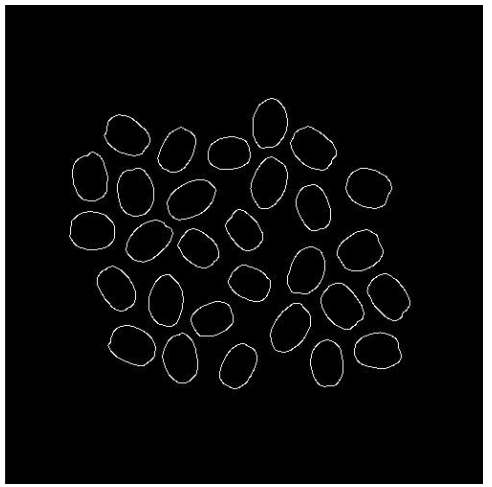




Segmentação de Objetos

Segmentação por Bordas

Nos slides anteriores, vimos alguns métodos para realçar contornos em imagens. Eles podem ser usados também para procedimentos de segmentação de objetos por borda. O mais eficiente desses métodos é o detector de bordas de Canny.



Segmentação de Objetos

Segmentação por Bordas

O código seguinte exemplifica como a segmentação por borda dos grãos de café. O primeiro passo é converter a imagem original para tons de cinza. Em seguida, aplicamos o procedimento de segmentação por binarização, removendo as informações sobre o segundo plano. Após essa etapa, devem ser aplicadas as operações morfológicas de fechamento e erosão, para tratamento de ruído. Por último, usamos o método Canny para detectar as bordas dos grãos, representando em uma nova imagem somente as bordas referentes aos objetos de interesse.

Segmentação de Objetos

Segmentação por Bordas

```
import cv2
import numpy as np

imagem = cv2.imread("graos-de-cafe.jpeg", 0)

metodo = cv2.THRESH_BINARY_INV
ret, imgBinarizada = cv2.threshold(imagem, 135, 255, metodo)
e = np.ones((3, 3), np.uint8)
imgTratada = cv2.morphologyEx(imgBinarizada, cv2.MORPH_CLOSE, e)
imgTratada = cv2.erode(imgTratada, e, iterations = 2)

imgSegmentada = cv2.Canny(imgTratada, 100, 200)

cv2.imshow("Binarizada", imgBinarizada)
cv2.imshow("Tratada", imgTratada)
cv2.imshow("Segmentada",  imgSegmentada)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Segmentação de Objetos

Segmentação por Movimento

O procedimento de segmentação por movimento consiste em detectar objetos que se moveram entre uma captura e outra, a partir de uma câmera fixa. O método mais eficaz para realizar essa tarefa é o subtract da biblioteca OpenCV,

Segmentação de Objetos

Segmentação por Movimento

A subtração de uma imagem por outra nos fornece a diferença entre elas, que vai representar o objeto de interesse. Os demais pixels que não sofreram mudanças são considerados o segundo plano da imagem, e todos serão apagados ou definidos como pixels pretos após esse procedimento.

```
import cv2
import numpy as np

cap1 = cv2.imread("captura-1.jpeg")
cap2 = cv2.imread("captura-2.jpeg")

imgRGB = cv2.subtract(cap1, cap2)
imgHSV = cv2.cvtColor(imgRGB, cv2.COLOR_BGR2HSV)

tomClaro = np.array([0, 120, 120])
tomEscuro = np.array([180, 255, 255])

imgSegmentada = cv2.inRange(imgHSV, tomClaro, tomEscuro)
cv2.imshow("Segmentada", imgSegmentada)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Segmentação de Objetos

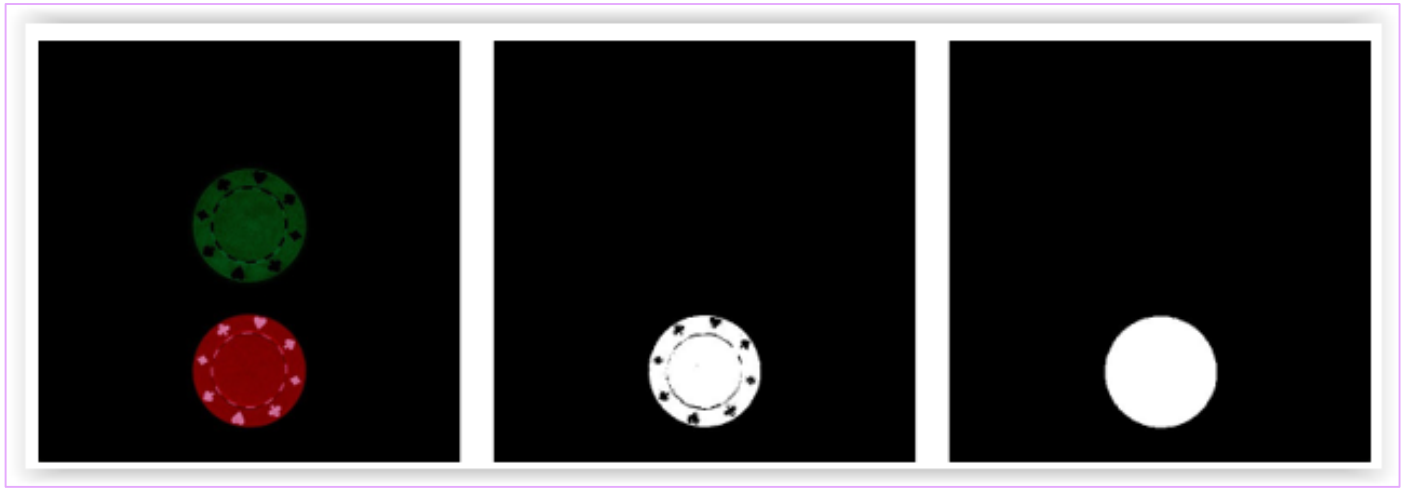
Segmentação por Movimento

Exemplo

Segmentação de Objetos

Segmentação por Movimento

Para segmentá-lo somente após a movimentação, uma solução é aplicar a segmentação por cor, uma vez que estão representados por cores diferentes. Em seguida, aplicaremos operações morfológicas para preencher a imagem ou tratar os ruídos.



Segmentação de Objetos

Referências

BARELLI, Felipe. Introdução à Visão Computacional: uma abordagem prática com Python e OpenCV. São Paulo: Casa do Código, 2018.

https://www.google.com.br/imghp