Curso de Especialização em Manufatura Avançada (indústria 4.0)
Processamento Digital de Sinais
Prof. Alexsandro Cristovão Bonatto

Atividade Individual - Detecção de objetos (APNP de 09/10/2024)

A detecção de objetos em imagens e sua contagem pode ser feita através de uma sequência de etapas de processamento de imagens, que resultam na detecção de bordas e contagem. A detecção de bordas é um processo essencial na análise de imagens, permitindo identificar as transições de intensidade que ocorrem entre regiões de uma imagem. A detecção de bordas é amplamente utilizada em áreas como visão computacional, reconhecimento de padrões e análise de imagem. Ao identificar as bordas em uma imagem, podemos destacar os objetos e as estruturas presentes, tornando mais fácil a identificação e o reconhecimento de elementos específicos.

Para este trabalho, escolha uma imagem que contenha um certo número de objetos sobre um fundo uniforme com pouca informação, como os exemplos abaixo:



Os passos necessários para realizar a detecção de bordas são:

1. Carregar uma imagem e fazer a sua conversão para tons de cinza.

Para fazer a conversão para tons de cinza (grayscale) pode-se utilizar a função

```
imGray = cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)
```

2. Aplicar o efeito de suavização (*blur*) para retirar o ruído e facilitar a identificação das bordas.

Para aplicar o efeito de suavização, pode-se utilizar o filtro de janela retangular, janela mediana, Gaussiano ou Bilateral. A função blur() é utilizada para suavização utilizando kernel de janela retangular.

```
kernel = (11,11)
imSuave = cv2.blur(imGray, kernel);
```

3. Aplicar uma binarização na imagem resultando em pixels só brancos e pretos.

A binarização é últil para remover detalhes indesejados da imagem facilitando o processo de detecção de bordas. Para uma binarização bem sucedida, é preciso ajustar o limiar que irá distinguir pixels pretos e brancos. A função cv2.threshold() permite a binarização.

```
limiar = 200
imBin = cv2.threshold(imSuave, limiar, 255, cv2.THRESH_BINARY)
```

4. Aplicar um detector de bordas para identificar os objetos.

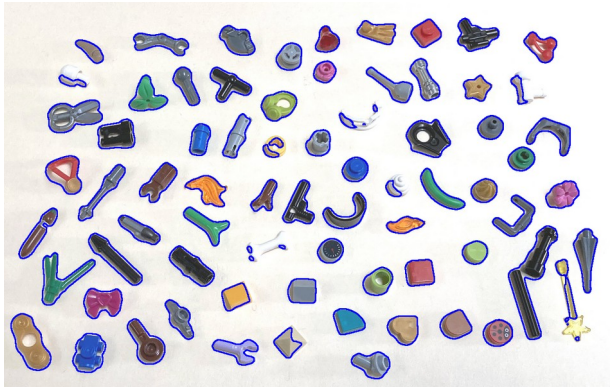
Existem diferentes métodos para detecção de bordas, disponíveis para uso na biblioteca OpenCV, mas o método *Canny*() é o mais utilizado por resultar em melhores contornos. Para que o método funcione adequadamente, é preciso ajustar dois limiares de histerese para detecção de bordas. O chamado limiar fraco é usado para indicar não bordas, quando abaixo deste limite. Já o limiar forte é usado para indicar bordas fortes quando acima deste limite.

```
limiarFraco = 70
limiarForte = 150
imBordas = cv2.Canny(imBin, limiarFraco, limiarForte)
```

5. Fazer a detecção de contornos da imagem com as bordas.

Com as bordas identificadas, vamos contar os contornos externos para achar a quantidade de dados presentes na imagem. A função cv2.*findContours*() é uma função da biblioteca OpenCV utilizada para encontrar os contornos em uma imagem. Os contornos são uma representação dos limites de objetos presentes em uma imagem, que podem ser utilizados para diversas finalidades, como detecção, segmentação e análise de formas.

```
objetos = cv2.findContours(imgBordas, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
objetos = objetos[0] if len(objetos) == 2 else objetos[1]
imgC2 = imgColorida.copy()
cv2_imshow(imgColorida)
cv2.drawContours(imgC2, objetos, -1, (255, 0, 0), 2)
cv2_imshow(imgC2)
cv2.waitKey(0)
print("Quantidade de objetos: "+str(len(objetos)))
```



Referências:

Arcanjo, Jonys. Contagem de Moedas com OpenCV: Detectando Contornos em Imagens. Disponível em: https://jonysarcanjo.medium.com/contagem-de-moedas-com-opencv-detectando-contornos-em-imagens-82b8f31e10b0

OpenCV. Contours : Getting Started. Disponível em: https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html