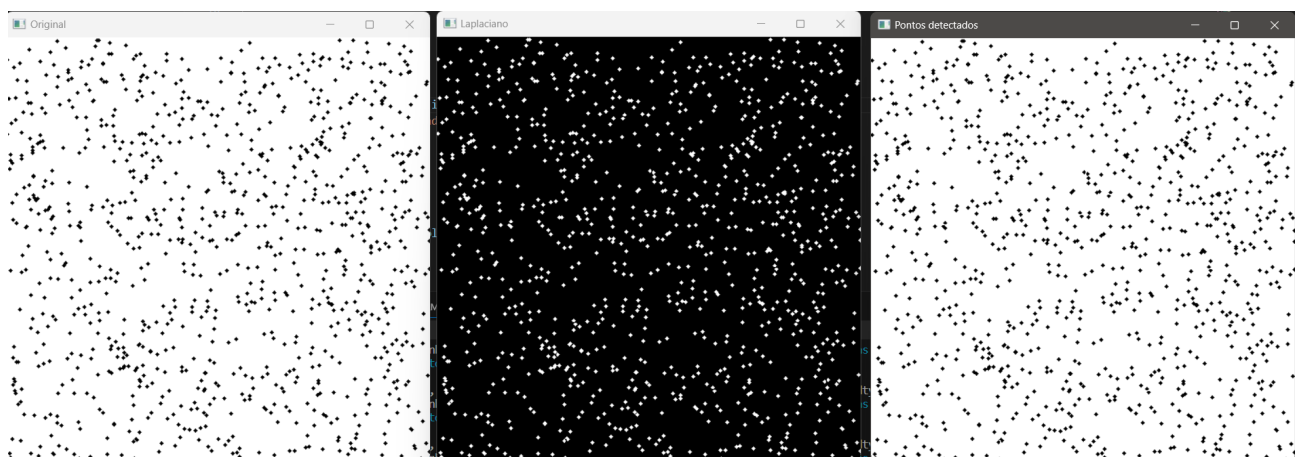
 <p>INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SÃO PAULO Campus Birigui</p>	<p>INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA Campus Birigui <b>Bacharelado em Engenharia de Computação</b></p>	
<p><b>Disciplina:</b> Processamento Digital de Imagens</p>	<p><b>Trabalho</b></p>	
<p><b>Professor:</b> Murilo Varges da Silva</p>	<p><b>Data:</b> 03/10/2021</p>	
<p><b>Nome do Aluno:</b> Leonardo Reneres dos Santos</p>	<p><b>Prontuário:</b> BI3009131</p>	

## Trabalho

### Exercícios:

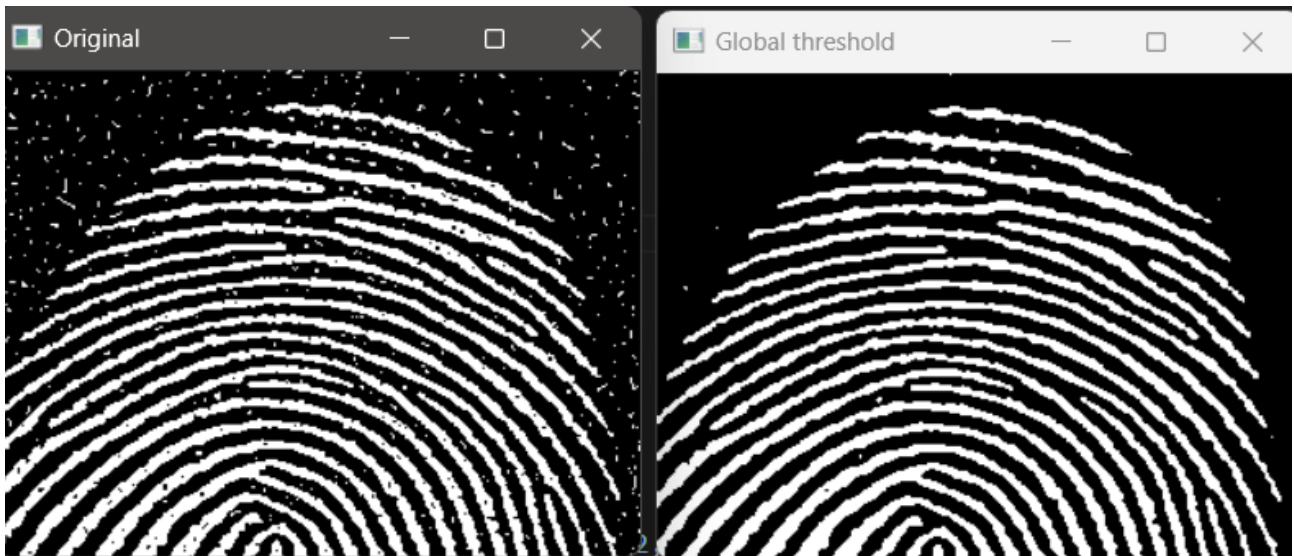
#### 1. Implementar detector de ponto conforme slide 17.



**1.1. Tirar uma foto de uma imagem com um fundo branco e fazer alguns pontos com caneta preta.**

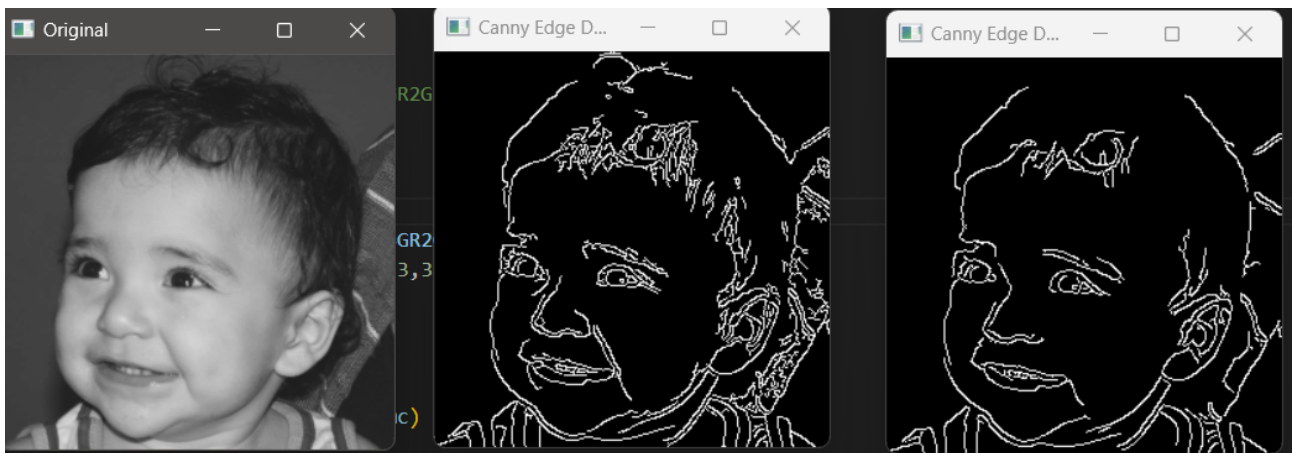
## 2. Implementar limiarização, definir o limiar.

### 2.1. Utilizar a imagem da impressão digital.



## 3. Implementar detector de bordas Canny.

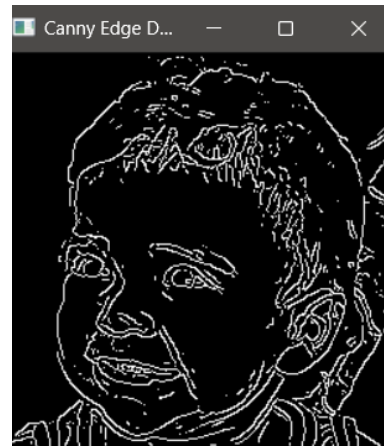
### 3.1. Aplicar o filtro de borramento (gaussiano) e verificar se o borramento melhora a detecção de bordas.



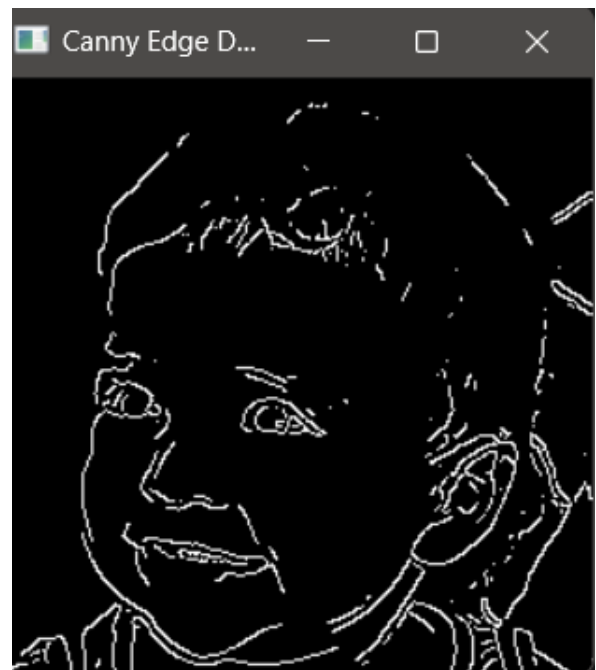
### 3.2. Mudar os parametros T1 e T2 e avaliar a qualidade das bordas detectadas.

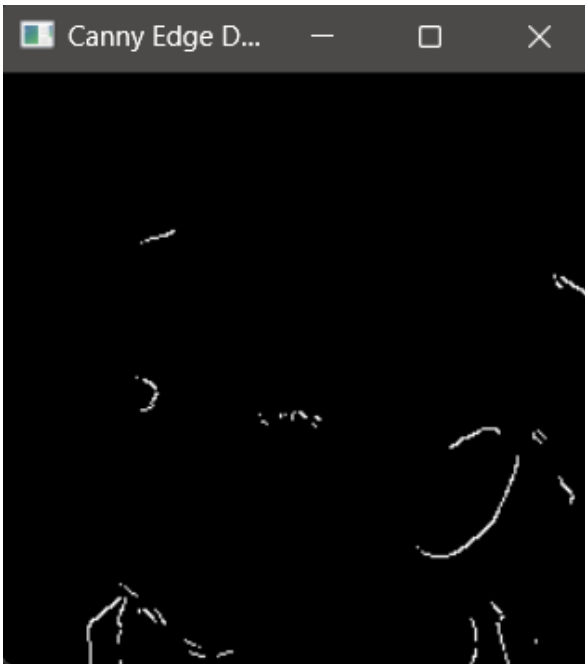

Valores	Imagens
---------	---------

t1=50  
t2 = 50



t1 = 100  
t2 = 100



<p>t1=300 t2=300</p>	 <p>A window titled "Canny Edge D..." showing a sparse edge detection result. The image is mostly black with a few thin, white line segments visible, representing the edges of the original image.</p>
<p>t1=20 t2=20</p>	 <p>A window titled "Canny Edge D..." showing a dense edge detection result. The image is mostly black with a high density of white line segments, representing the edges of the original image.</p>

### Código:

```
import cv2 as cv
import numpy as np
```

```

kernel_laplaciano = np.array([[1, 1, 1], [1, -4, 1], [1, 1, 1]],dtype="int")

kernel_laplacian_large = np.array([[1, 1, 1, 1, 1], [1, 1, 1, 1, 1], [1, 1, -24, 1, 1], [1, 1, 1, 1, 1], [1, 1, 1, 1, 1]],dtype="int")

def detectarpontos(im):

    iml = cv.Laplacian(im,cv.CV_8U,ksize=5)
    imlA = np.abs(iml)

    T = 0.9*np.max(imlA)

    print(T)

    pontos = np.where(imlA>0,0,255).astype('uint8')

    print(np.where(pontos>0))

    cv.imshow('Original', im)
    cv.imshow('Laplaciano ', iml)
    cv.imshow('Laplaciano Absoluto', imlA)
    cv.imshow('Pontos detectados', pontos)

    cv.waitKey(0)
    cv.destroyAllWindows()

def canny(im,t1,t2):
    imc = cv.Canny(im,threshold1=t1, threshold2=t2, apertureSize = 3)
    return imc

def limiarizacao(im):
    # global thresholding
    im_blur = cv.GaussianBlur(im, (5,5), 0)
    ret,th = cv.threshold(im_blur,127,255,cv.THRESH_BINARY)

    cv.imshow('Original', im)
    cv.imshow('Global threshold', th)
    cv.waitKey(0)

```

```

cv.destroyAllWindows()

def main():
    # Crie uma imagem em branco (255 é branco em escala de cinza)
    imagem = np.ones((500, 500), np.uint8) * 255

    # Defina a cor dos pontos como preto
    cor_ponto = 1

    # Crie pontos pretos na imagem
    for _ in range(1000):
        # Gere coordenadas aleatórias para os pontos
        x = np.random.randint(1, imagem.shape[1])
        y = np.random.randint(1, imagem.shape[1])

        # Desenhe o ponto na imagem
        cv.circle(imagem, (x, y), 2, cor_ponto, -1)

    # Exiba a image

    detectarPontos(imagem)

    # img = cv.imread('fingerprint.tif')
    # img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

    # limiarizacao(img)

    # im = cv.imread('biel.png')
    # im_gray = cv.cvtColor(im, cv.COLOR_BGR2GRAY)
    # im_blur = cv.GaussianBlur(im_gray, (3,3), 1)

    # imc = canny(im_gray, 50, 150)
    # cv.imshow('Canny Edge Detection', imc)

    # imc2 = canny(im_blur, 50, 150)
    # cv.imshow('Canny Edge Detection with Blur', imc2)
    # cv.waitKey(0)

    # cv.destroyAllWindows()

    # while(True):
    #     t1 = int(input("Digite o valor de t1: "))

```

```
#     t2 = int(input("Digite o valor de t2: "))

#     if(t1 == 0 or t2 == 0):
#         break
#     else:
#         imc3 = canny(im_blur,t1,t2)
#         cv.imshow('Canny Edge Detection with Blur and T1, T2', imc3)
#         cv.waitKey(0)
#         cv.destroyAllWindows()

if __name__ == "__main__":
    main()
```