Erick de Sousa Almeida - BI3008444

Filtragem Espacial

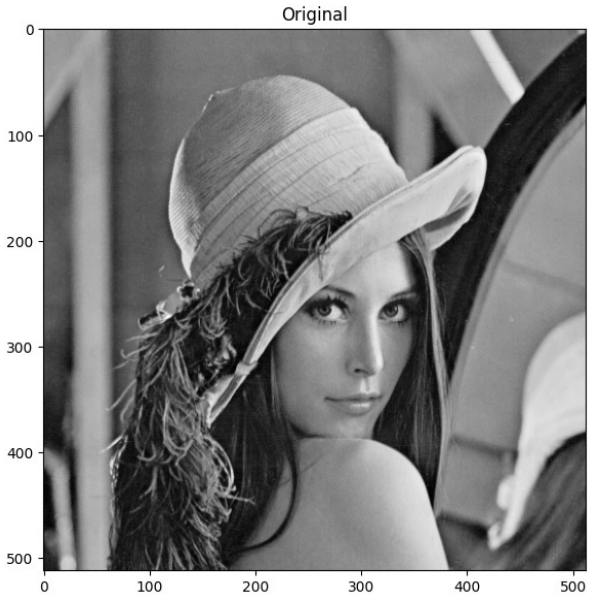
1. Implementar a operação de convolução.

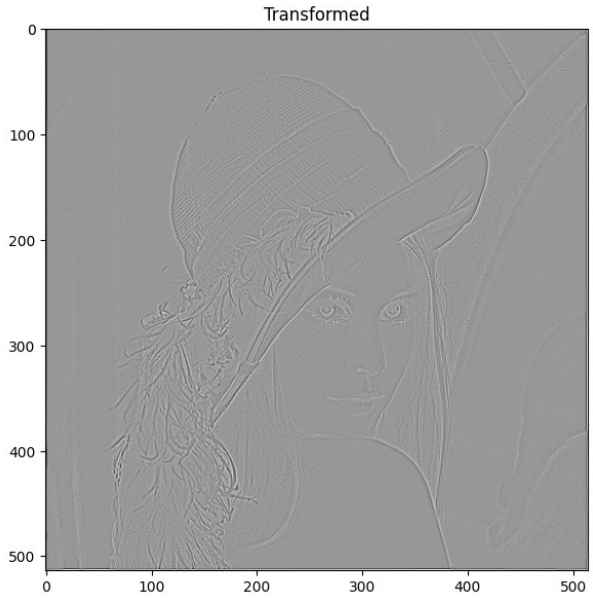
```
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image

images = [np.array(Image.open(img)) for img in ['lena_gray_512.tif',
'cameraman.tif', 'biel.png']]


def plot_result(original, transformed, *args):
    fig, axs = plt.subplots(1, 2, figsize=(15, 15))
    axs[0].imshow(original, cmap='gray')
    axs[0].set_title('Original')
    axs[1].imshow(transformed, cmap='gray')
    axs[1].set_title('Transformed')


def process(img, transformation, *args):
    transformed = transformation(img, *args)
    plot_result(img, transformed, *args)
```

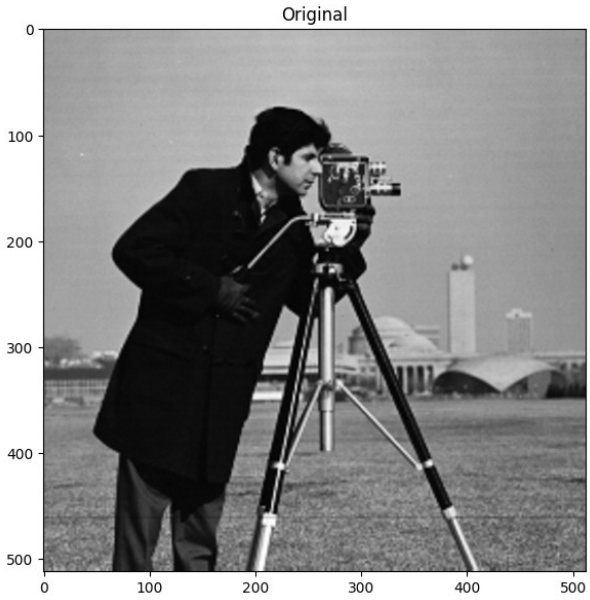
Convolução manual

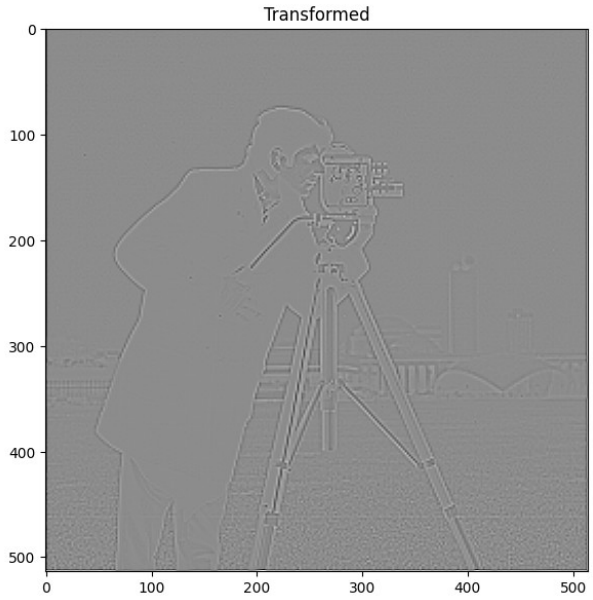
```
def convolve(signal, kernel):
    output_rows = signal.shape[0] + kernel.shape[0] - 1
    output_cols = signal.shape[1] + kernel.shape[1] - 1
    output = [[0 for j in range(output_cols)] for i in
range(output_rows)]
    kernel = kernel[::-1, ::-1]
    padded_signal = [[0 for j in range(signal.shape[1] + 2 *
(kernel.shape[1] - 1))] for i in
                     range(signal.shape[0] + 2 * (kernel.shape[0] -
1))]
    for i in range(signal.shape[0]):
        for j in range(signal.shape[1]):
            padded_signal[i + kernel.shape[0] - 1][j + kernel.shape[1]
- 1] = signal[i][j]
    for i in range(output_rows):
        for j in range(output_cols):
            for k in range(kernel.shape[0]):
                for l in range(kernel.shape[1]):
                    output[i][j] += padded_signal[i + k][j + l] *
kernel[k][l]
```

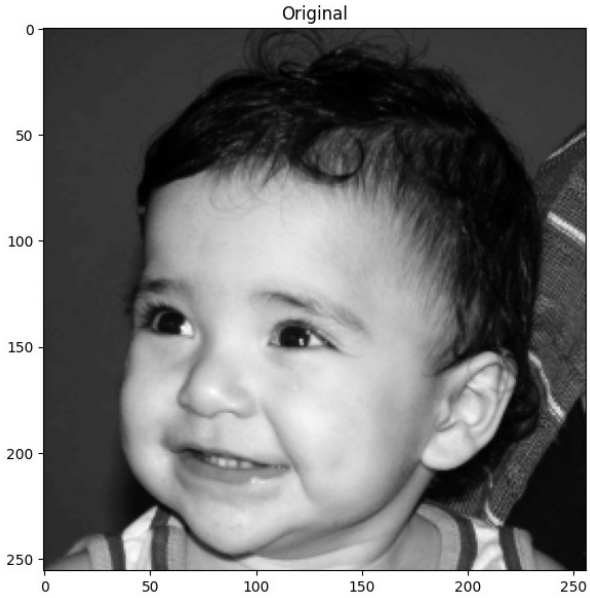
```
    return output


for img in images:
    process(img, convolve, np.array([[0, 1, 0],
                                     [1, -4, 1],
                                     [0, 1, 0]]))
```

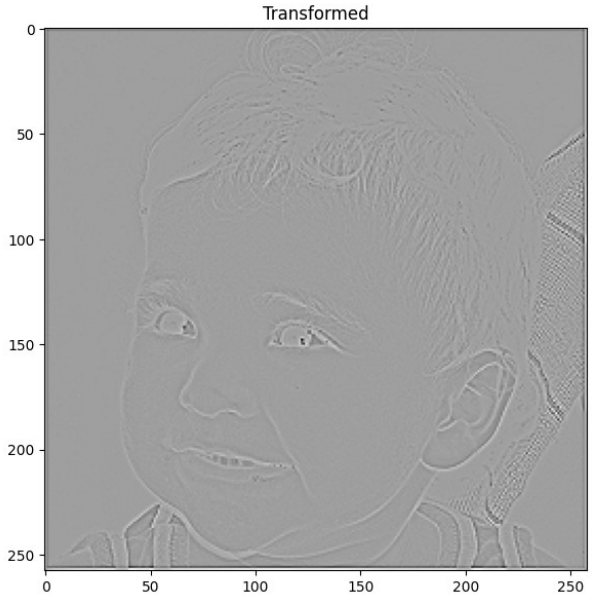






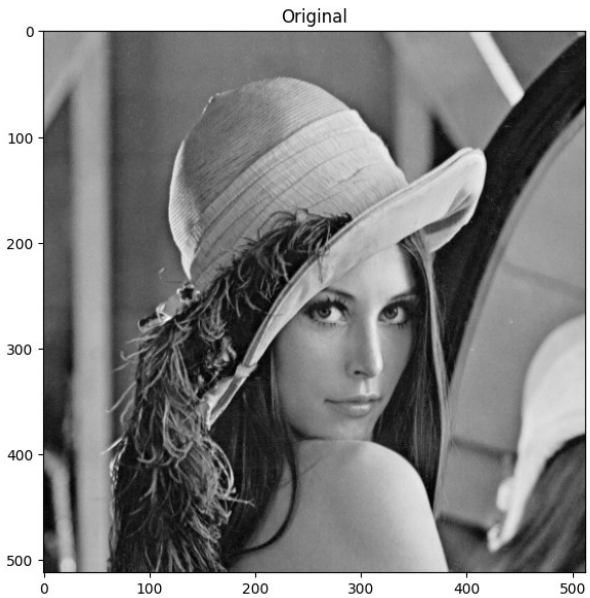


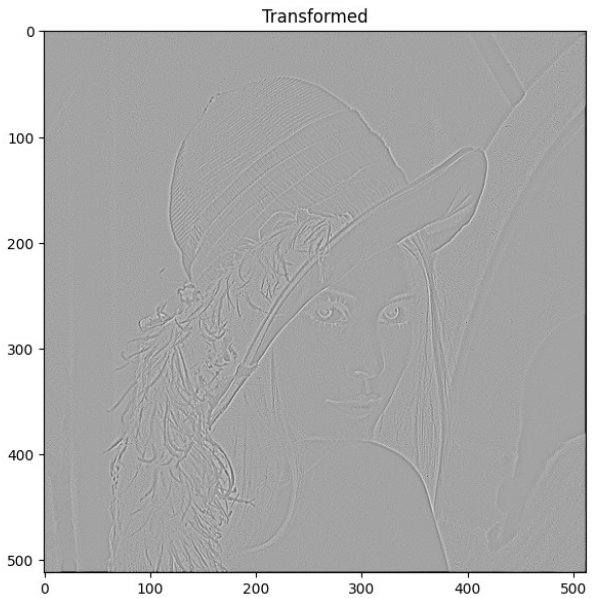


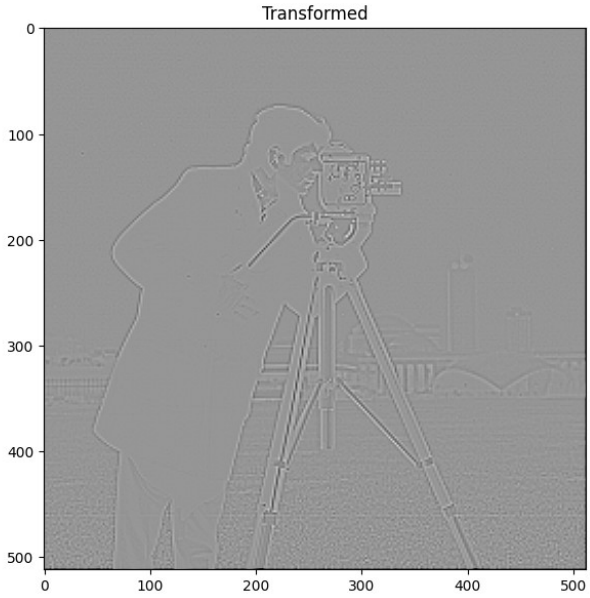


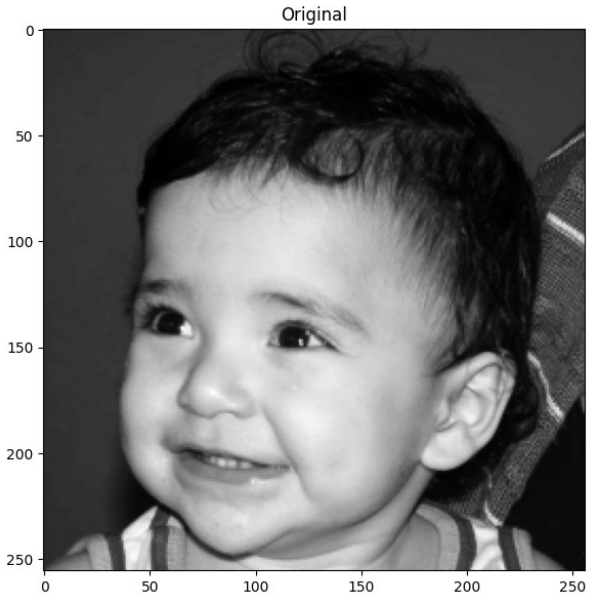
Utilizando scipy

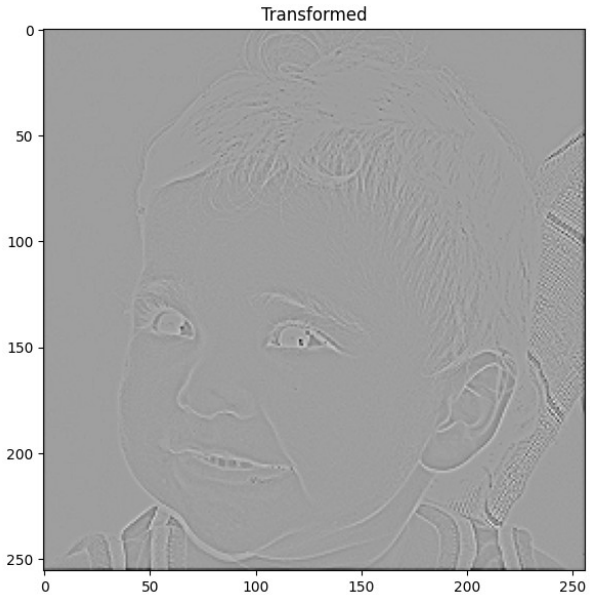
```
def scipy_convolve(img, kernel):
    from scipy.signal import convolve2d
    return convolve2d(img, kernel, mode='same', boundary='fill',
fillvalue=0)


for img in images:
    process(img, scipy_convolve, np.array([[0, 1, 0],
                                           [1, -4, 1],
                                           [0, 1, 0]]))
```





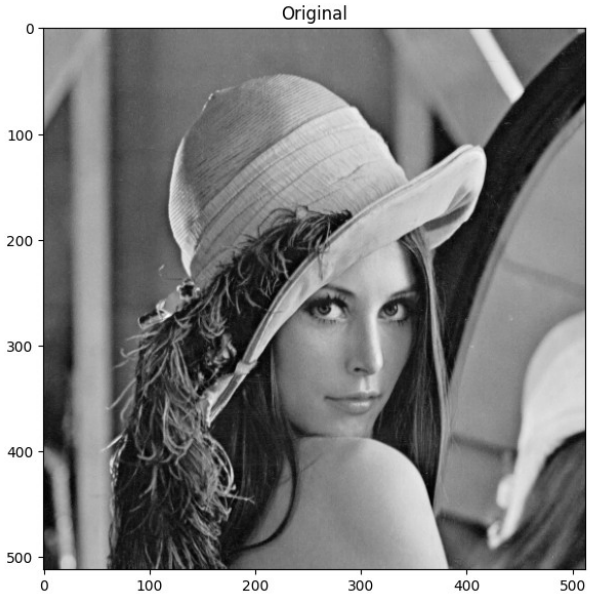


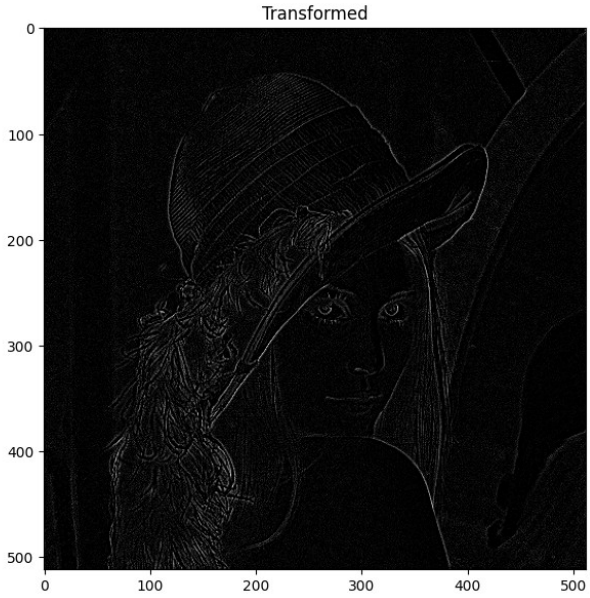


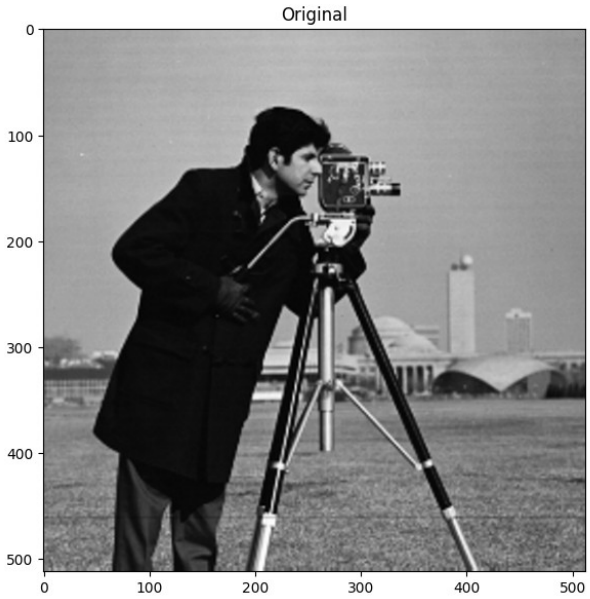


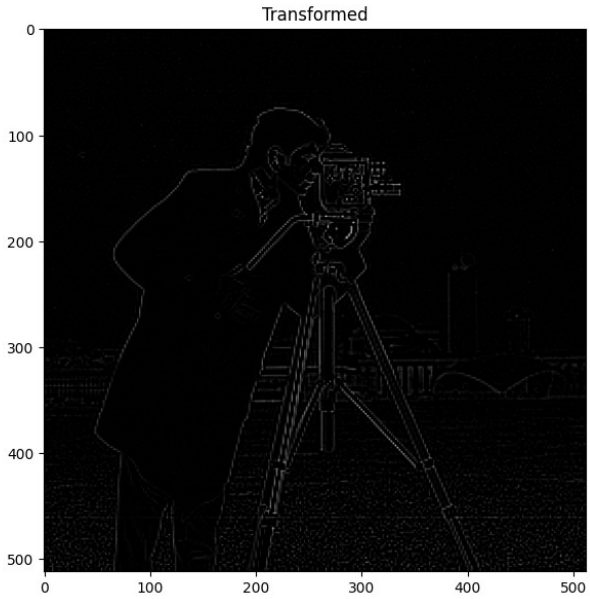
Utilizando openCV

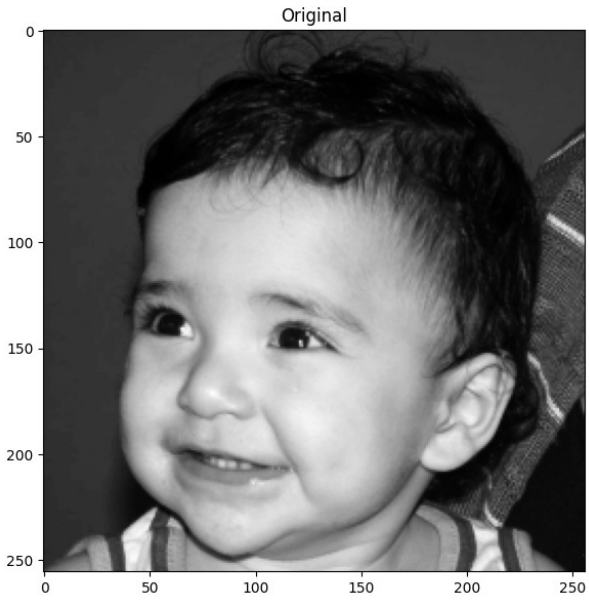
```
def opencv_convolve(img, kernel):
    return cv2.filter2D(img, -1, kernel)


for image in images:
    process(image, opencv_convolve, np.array([[0, 1, 0],
                                               [1, -4, 1],
                                               [0, 1, 0]]))
```

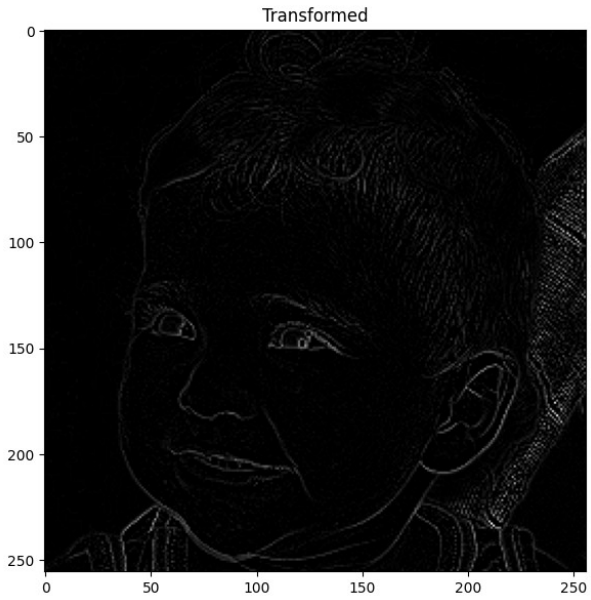

Original
0
100
200
300
400
500
0
100
200
300
400
500


Transformed
0
100
200
300
400
500
0
100
200
300
400
500


Original
0
100
200
300
400
500
0
100
200
300
400
500


Transformed
0
100
200
300
400
500
0
100
200
300
400
500





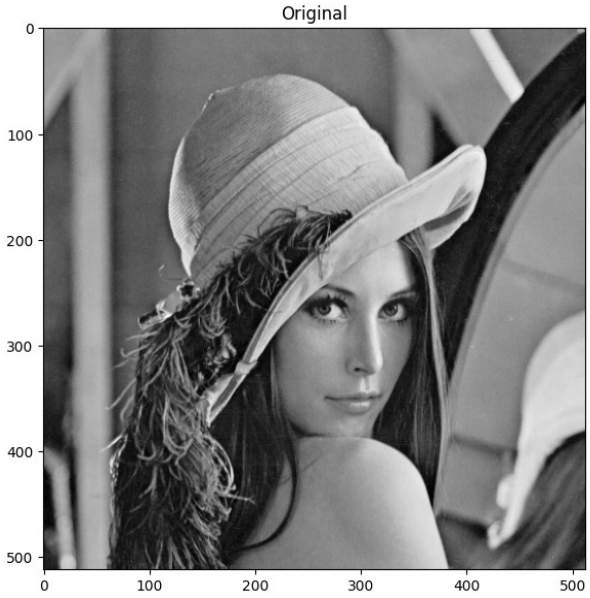
1. Utilizando OPENCV, scipy função convolve e implementação manual. Implementar seguintes máscaras:

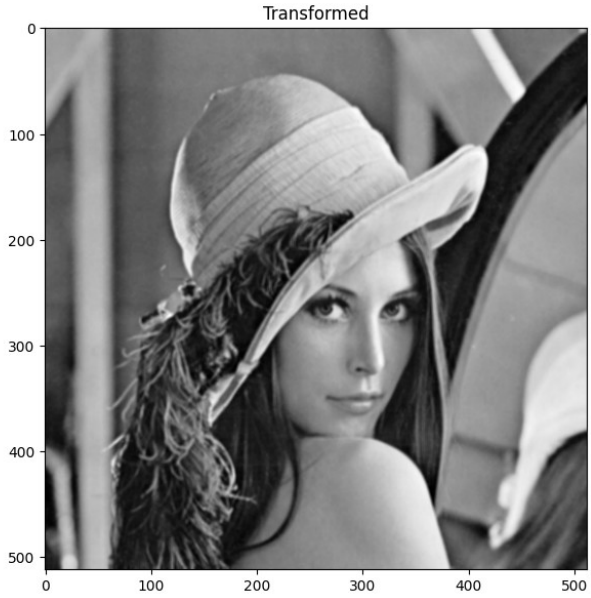
- Média
- Guassiano
- Laplaciano
- Sobel X
- Sobel Y
- Gradiente (Sobel X + Sobel Y)
- Laplaciano somado a imagem original

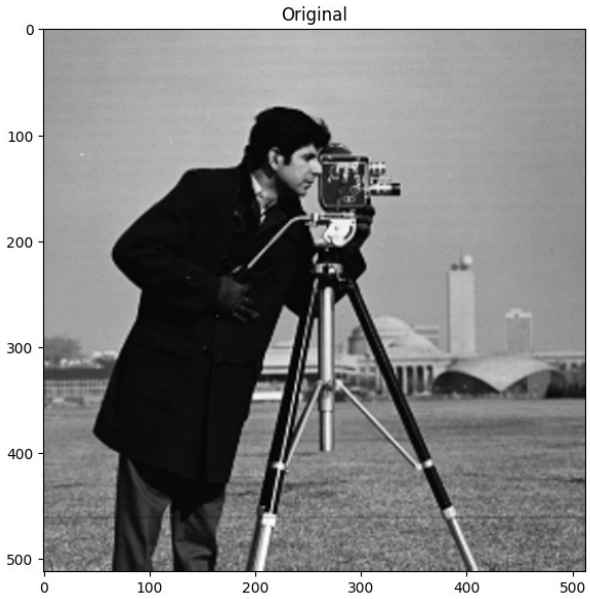
Utilizando o a mascara da média com openCV

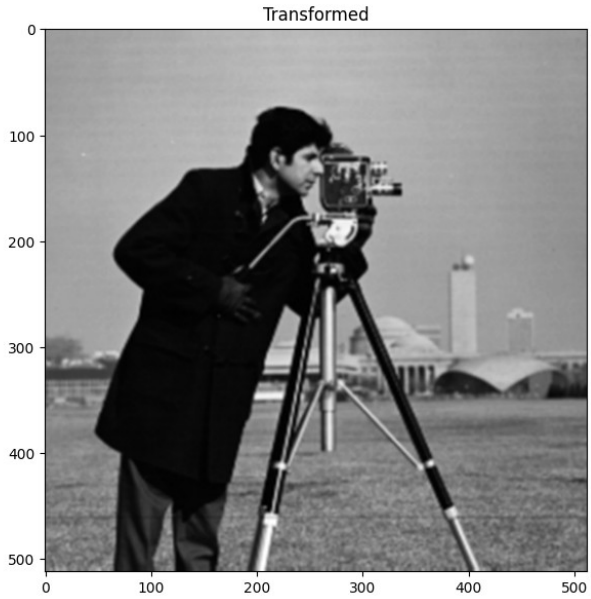
```
def get_identity_kernel():
    return np.array((
        [0, 0, 0],
        [0, 1, 0],
        [0, 0, 0]), dtype="int")


def get_mean_kernel():
    return np.array((
        [0.1111, 0.1111, 0.1111],
        [0.1111, 0.1111, 0.1111],
        [0.1111, 0.1111, 0.1111]), dtype="float")


def get_gaussian_kernel():
    return np.array((
        [0.0625, 0.125, 0.0625],
        [0.1250, 0.250, 0.1250],
```

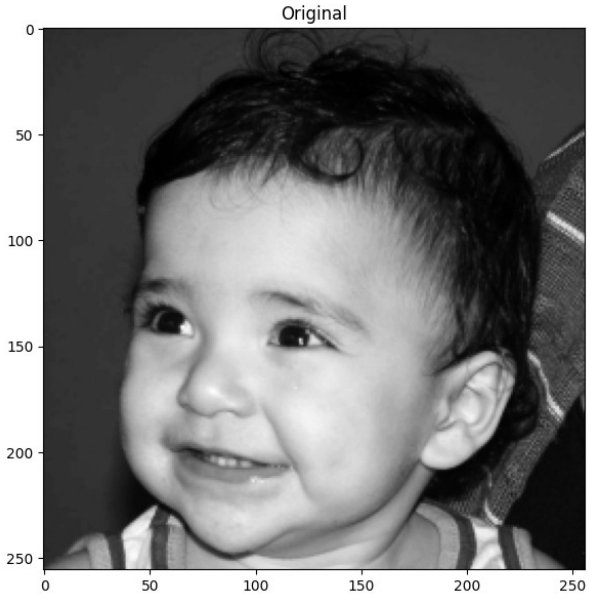
```
        [0.0625, 0.125, 0.0625]), dtype="float")


def get_laplacian_kernel():
    return np.array((
        [0, 1, 0],
        [1, -4, 1],
        [0, 1, 0]), dtype="int")


def get_sobel_x_kernel():
    return np.array((
        [-1, 0, 1],
        [-2, 0, 2],
        [-1, 0, 1]), dtype="int")


def get_sobel_y_kernel():
    return np.array((
        [-1, -2, -1],
        [0, 0, 0],
        [1, 2, 1]), dtype="int")


def get_laplacian_sum_kernel(img):
    return np.array((
        [0, 1, 0],
        [1, -4, 1],
        [0, 1, 0]), dtype="int")

import cv2

for img in images:
    process(img, opencv_convolve, get_mean_kernel())
```

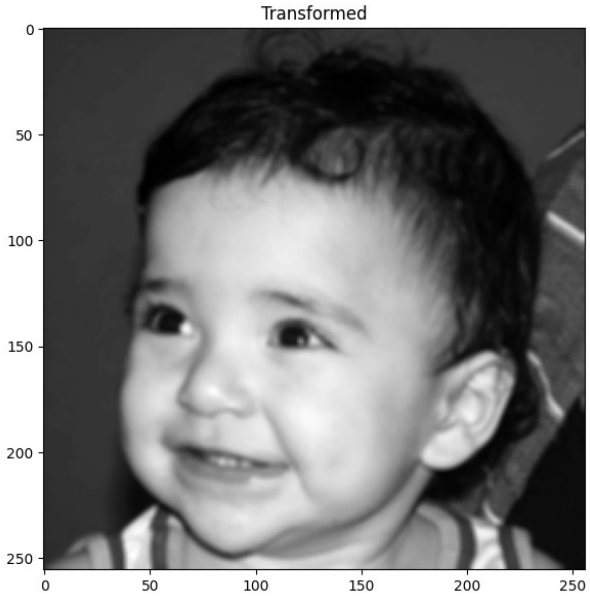

Original
0
100
200
300
400
500
0
100
200
300
400
500


Transformed
0
100
200
300
400
500
0
100
200
300
400
500


Original
0
100
200
300
400
500
0
100
200
300
400
500

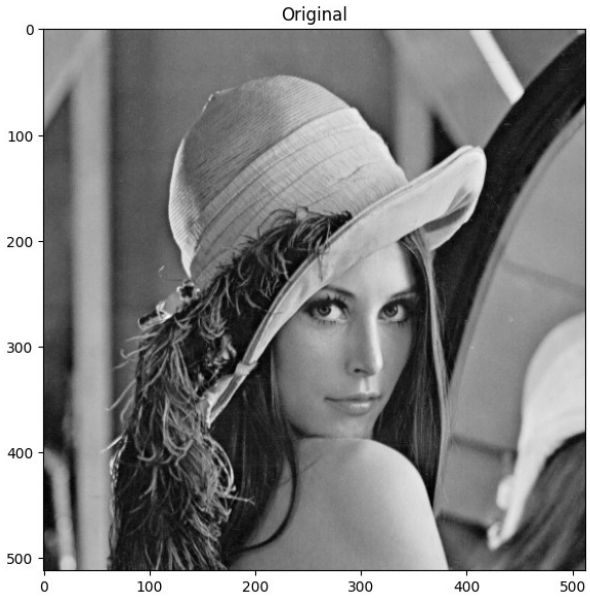

Transformed
0
100
200
300
400
500
0
100
200
300
400
500

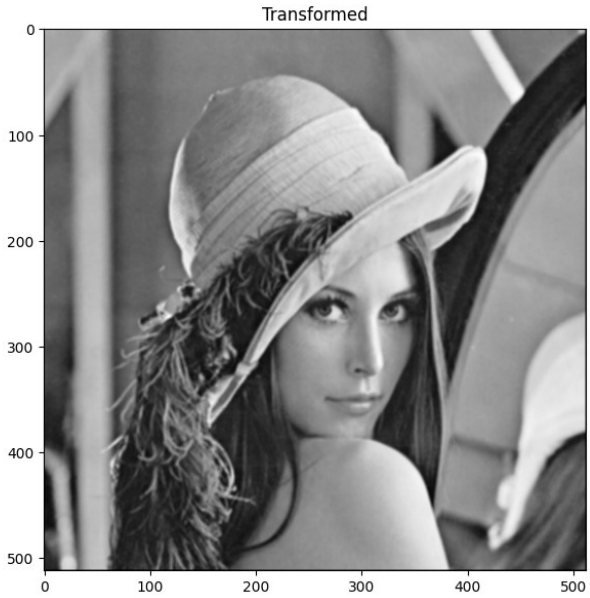


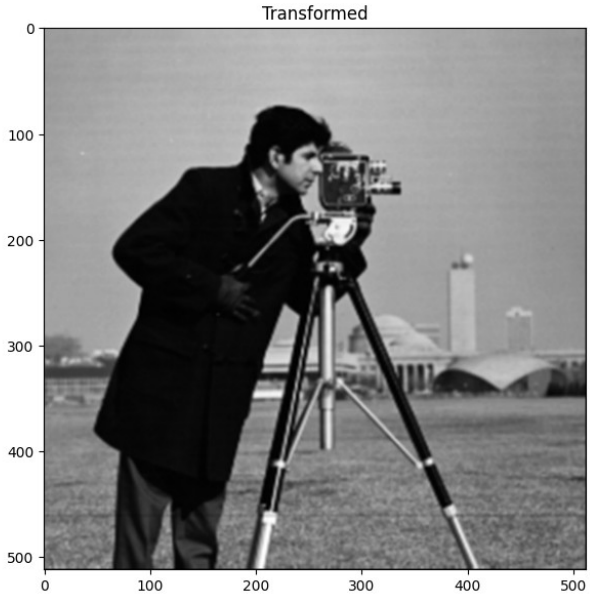


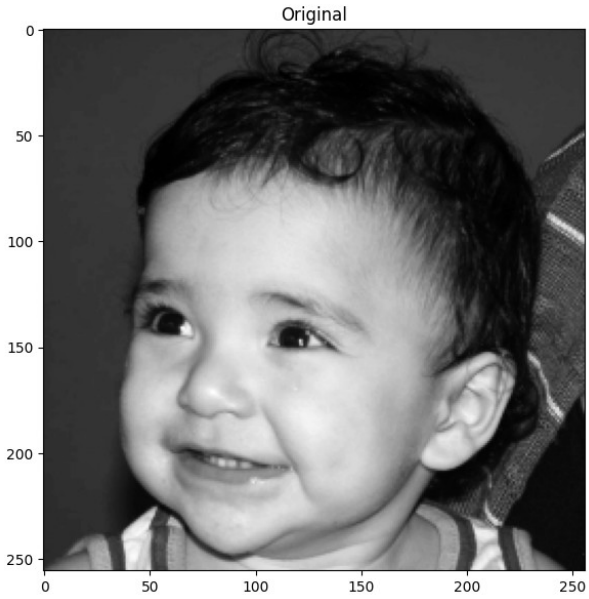
Utilizando o a mascara da média com scipy

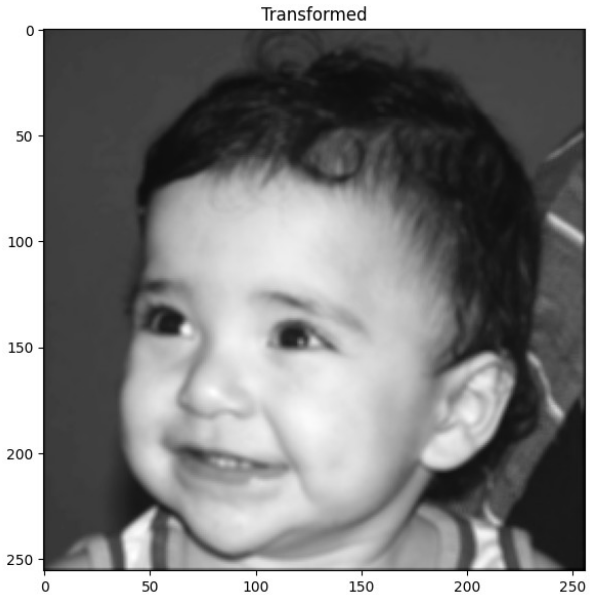
```
for img in images:
    process(img, scipy_convolve, get_mean_kernel())
```





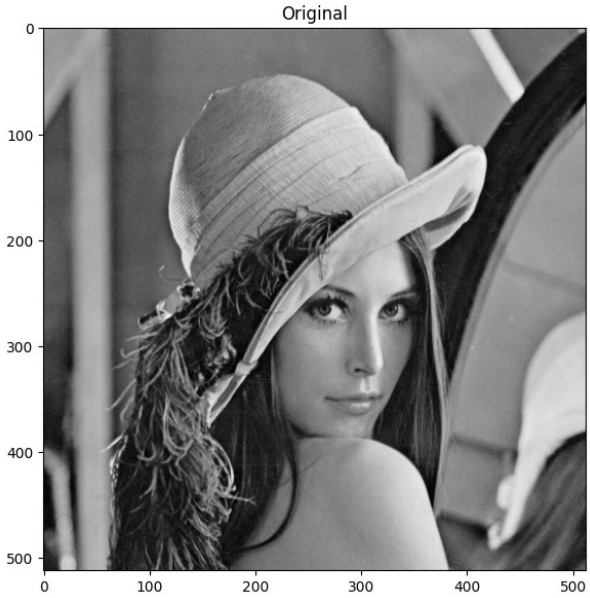


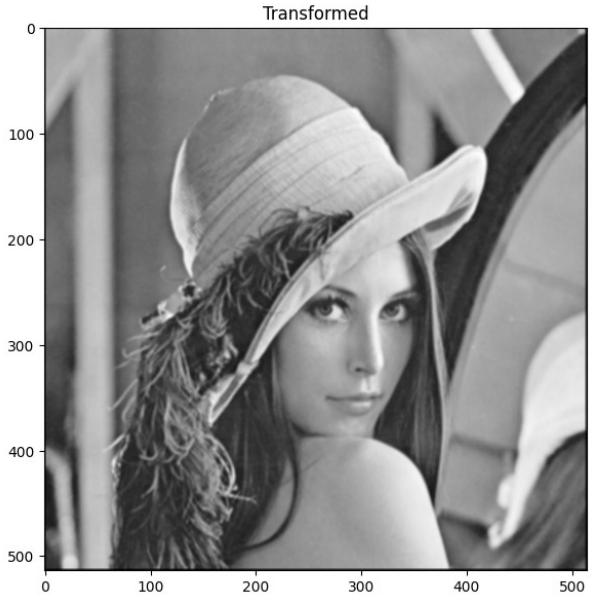




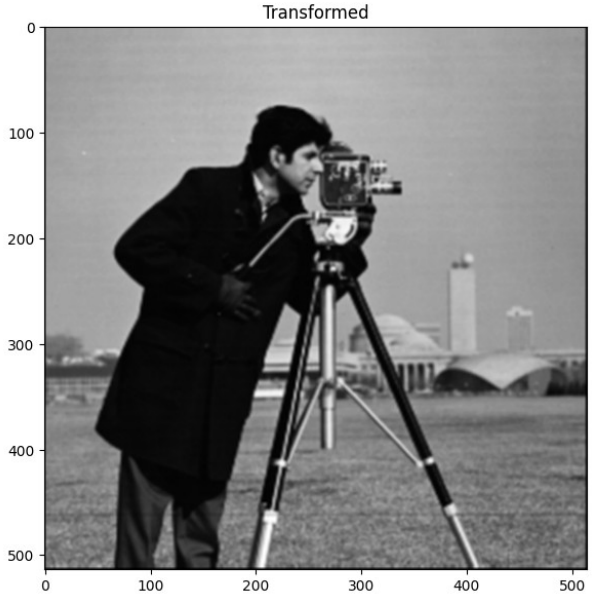
Utilizando o a mascara da média com implementação manual

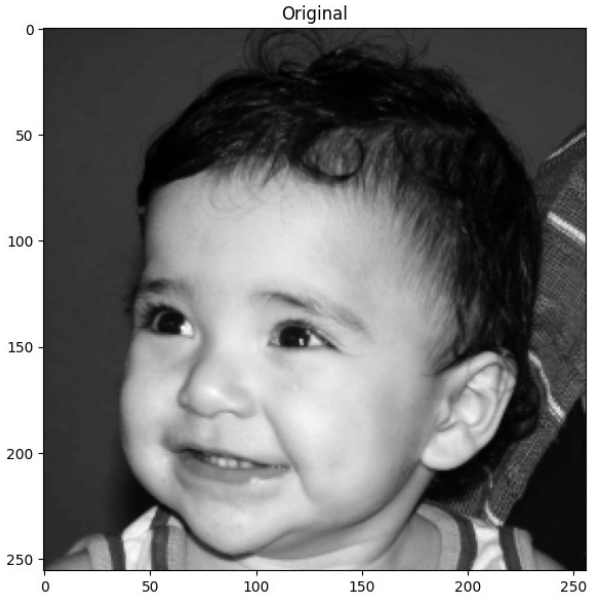
```
for img in images:
    process(img, convolve, get_mean_kernel())
```

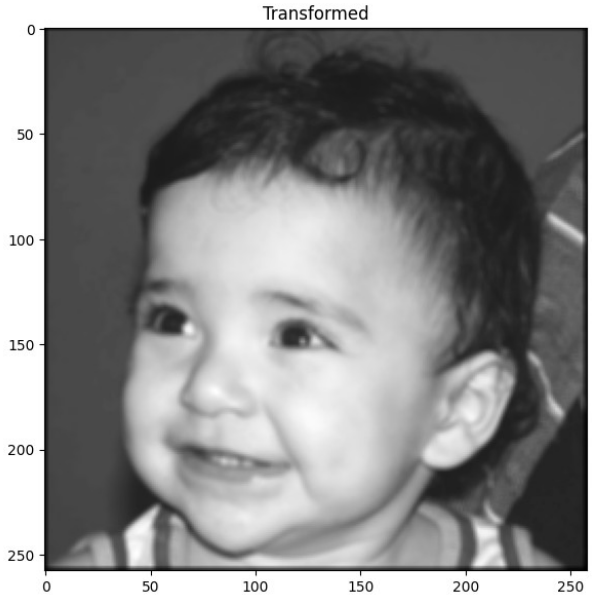

Original
0
100
200
300
400
500


Transformed
0
100
200
300
400
500


Original
0
100
200
300
400
500

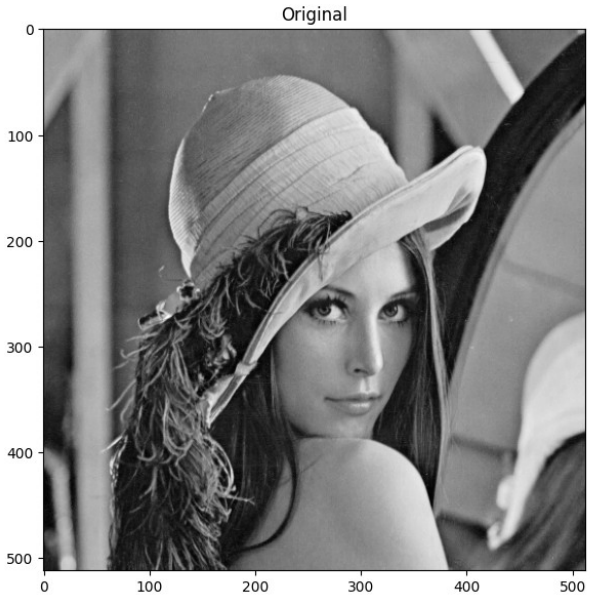

Transformed
0
100
200
300
400
500

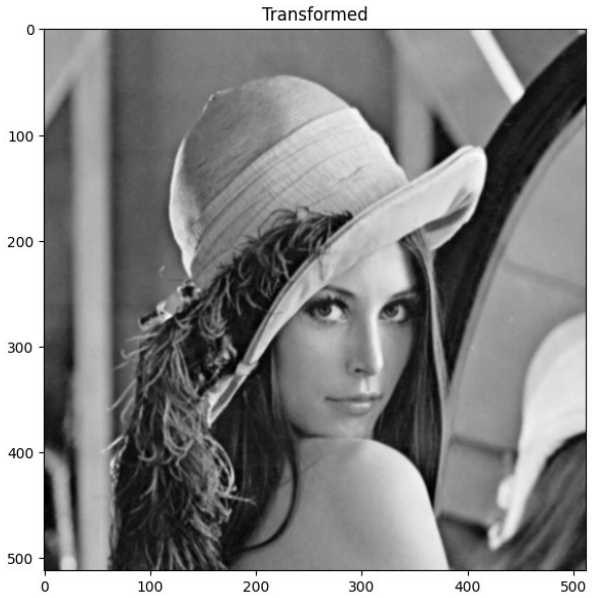


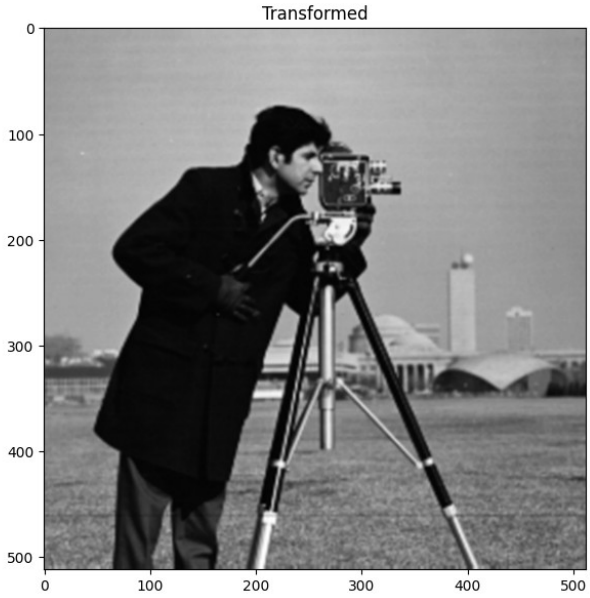


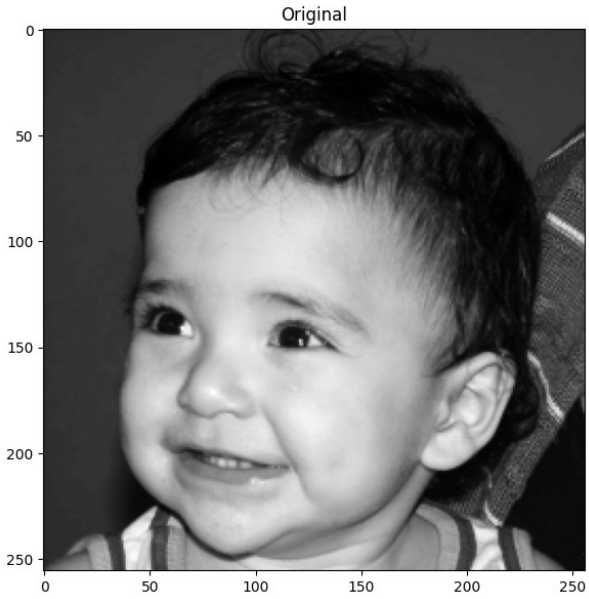
Utilizando o a mascara gaussiana com openCV

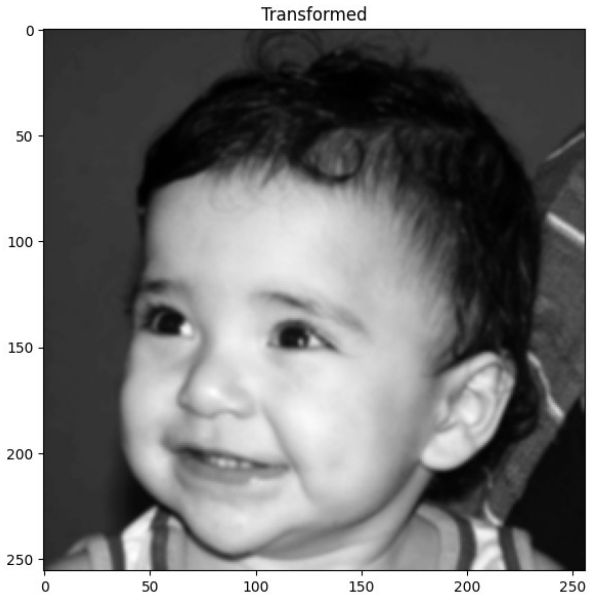
```
for img in images:
    process(img, opencv_convolve, get_gaussian_kernel())
```





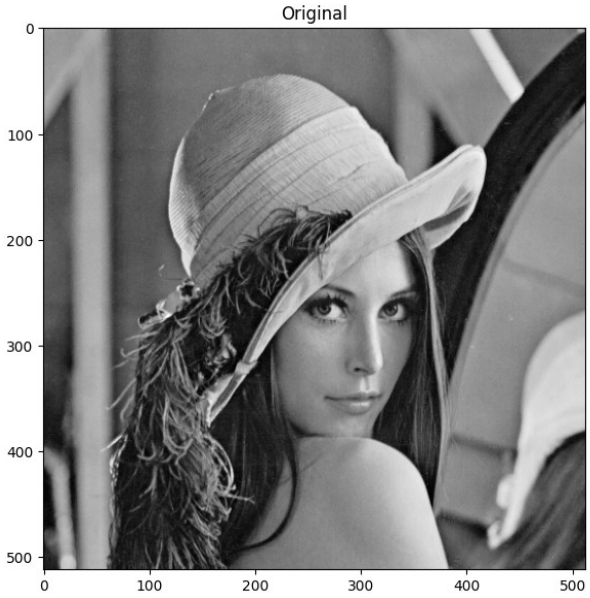


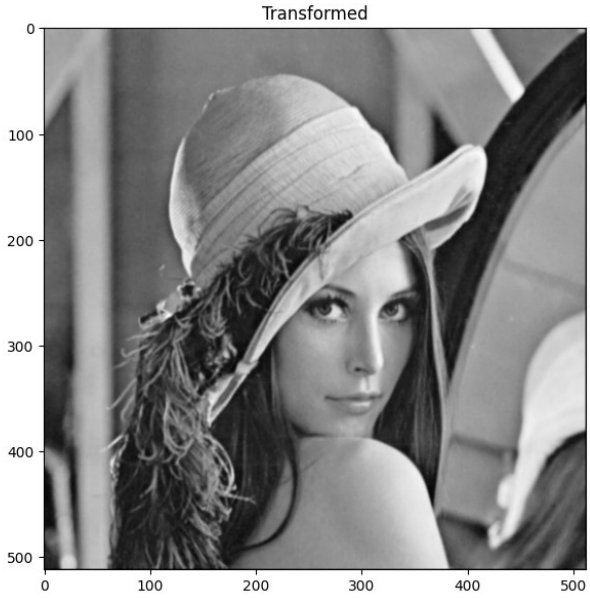


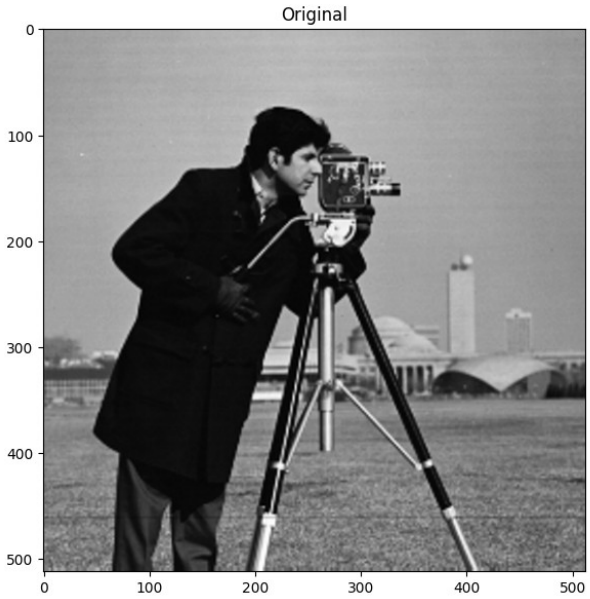


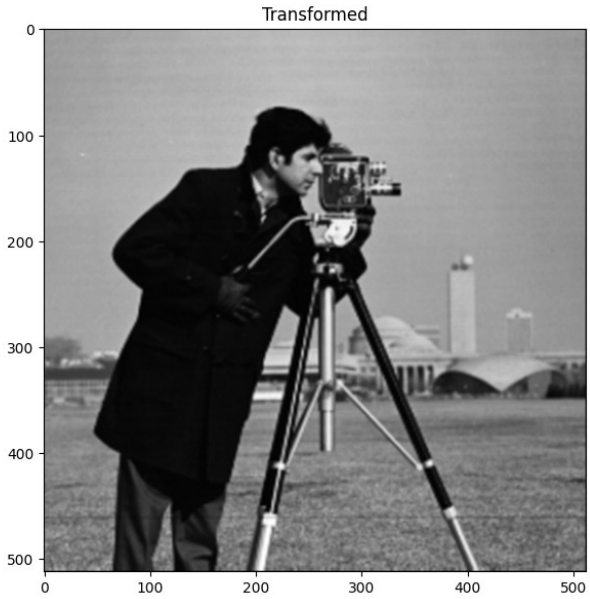
Utilizando o a mascara gaussiana com scipy

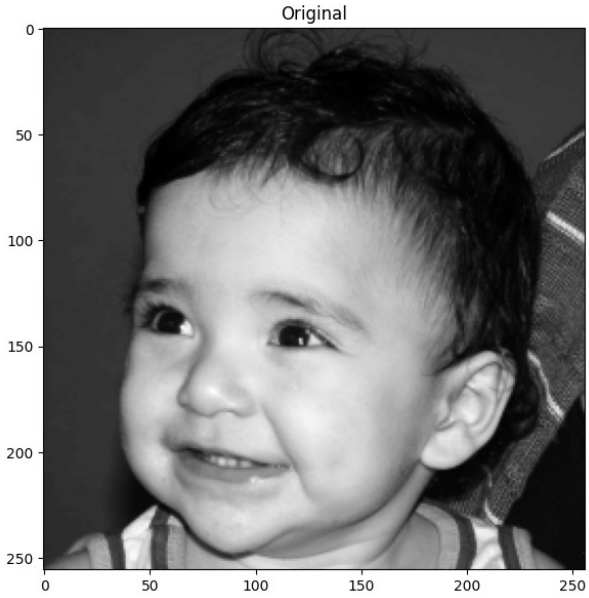
```
for img in images:
    process(img, scipy_convolve, get_gaussian_kernel())
```

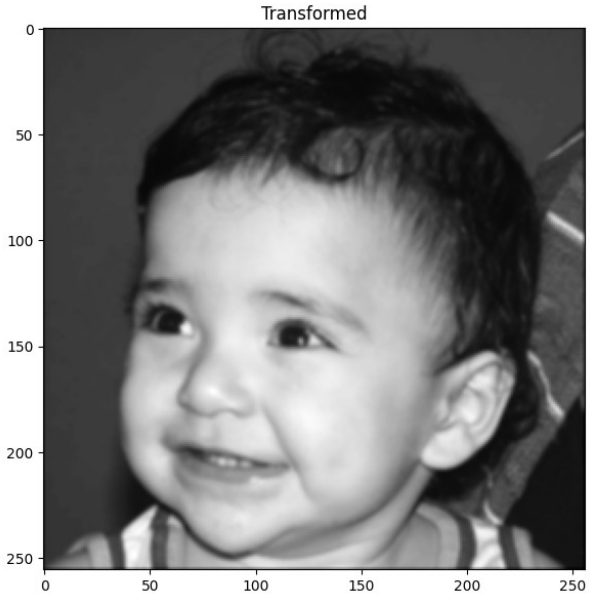

Original
0
100
200
300
400
500
0
100
200
300
400
500


Transformed
0
100
200
300
400
500
0
100
200
300
400
500


Original
0
100
200
300
400
500
0
100
200
300
400
500

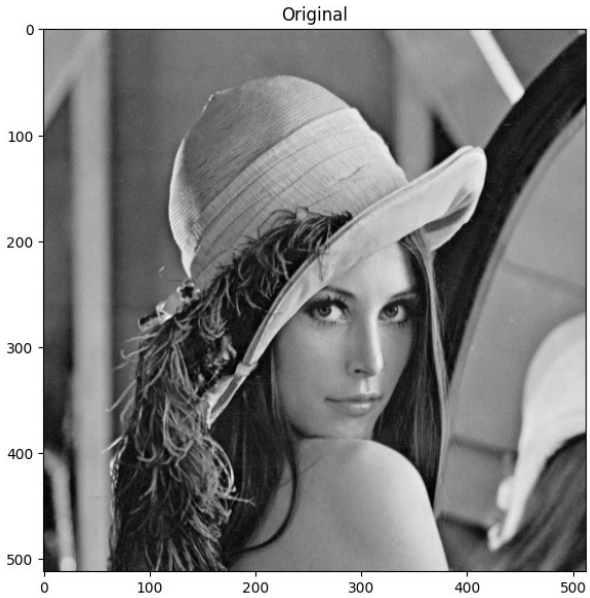

Transformed
0
100
200
300
400
500
0
100
200
300
400
500

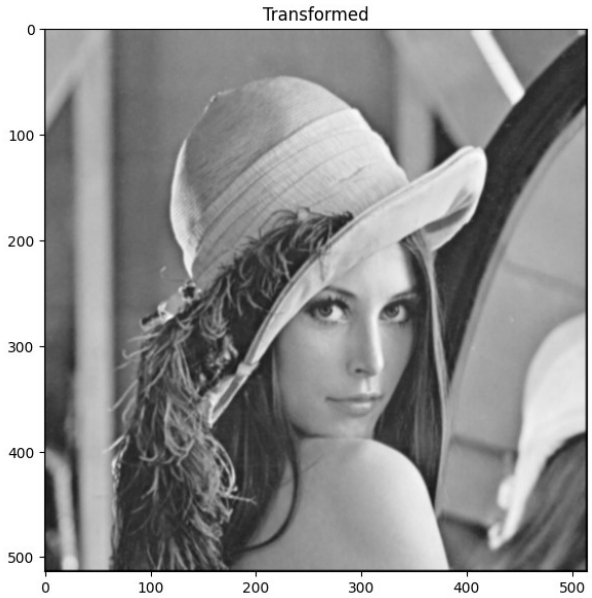


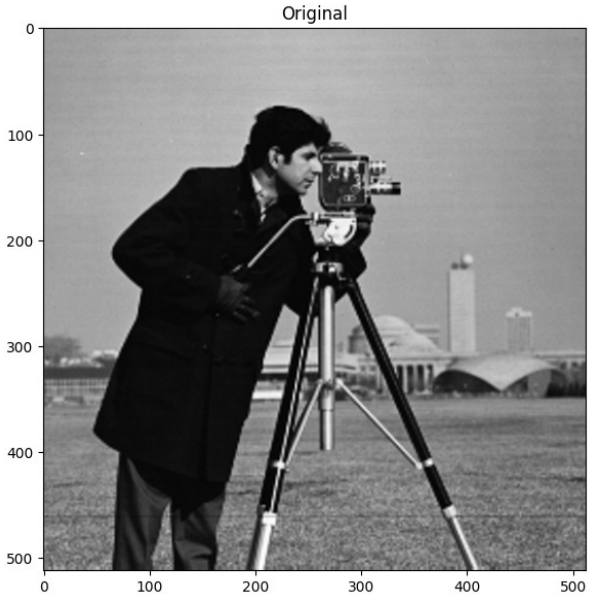


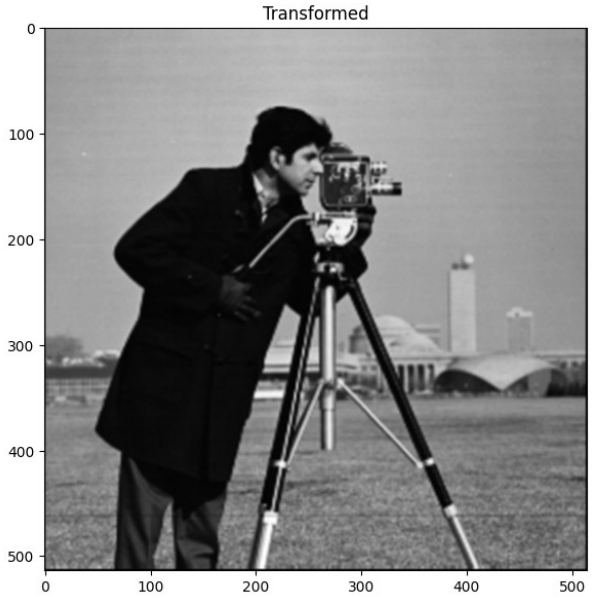
Utilizando o a mascara gaussiana com implementação manual

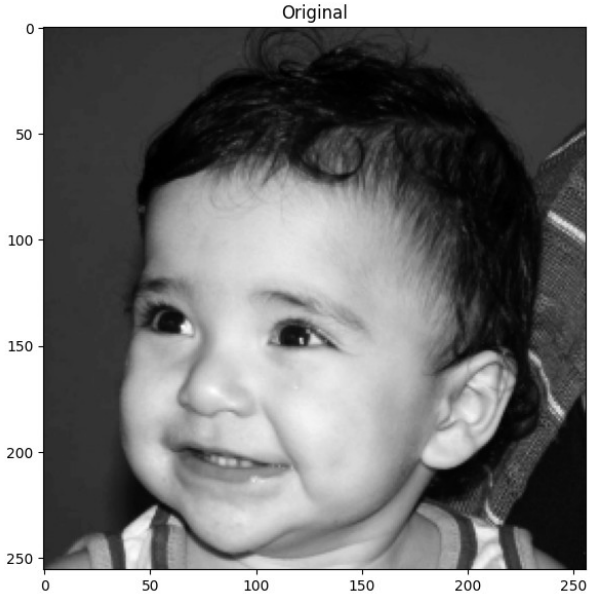
```
for img in images:
    process(img, convolve, get_gaussian_kernel())
```

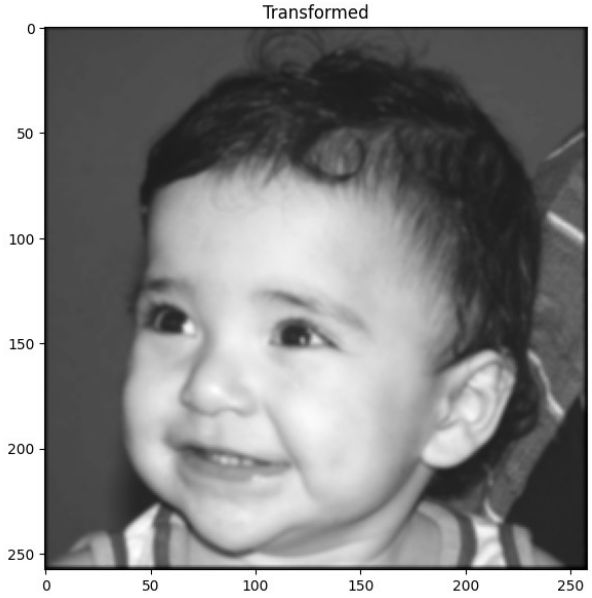






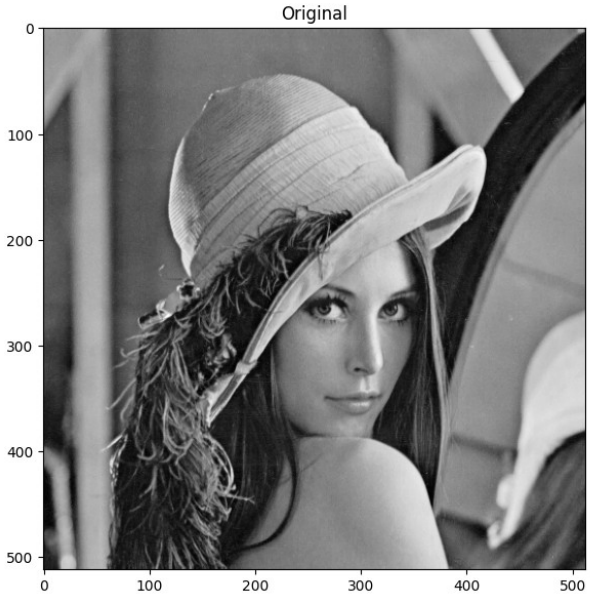


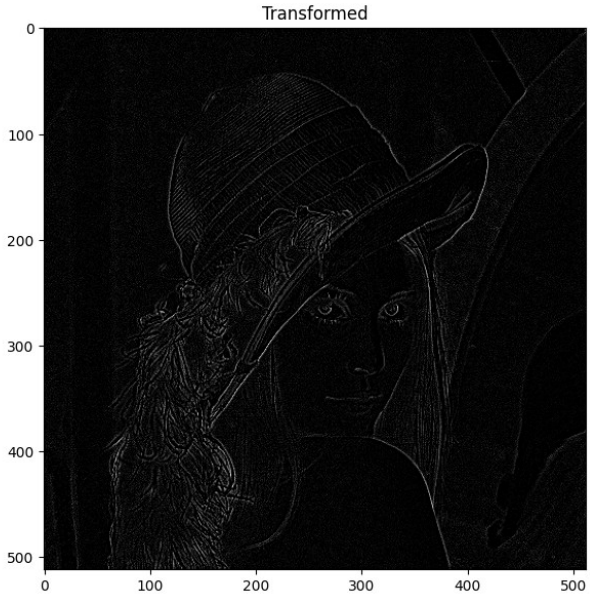


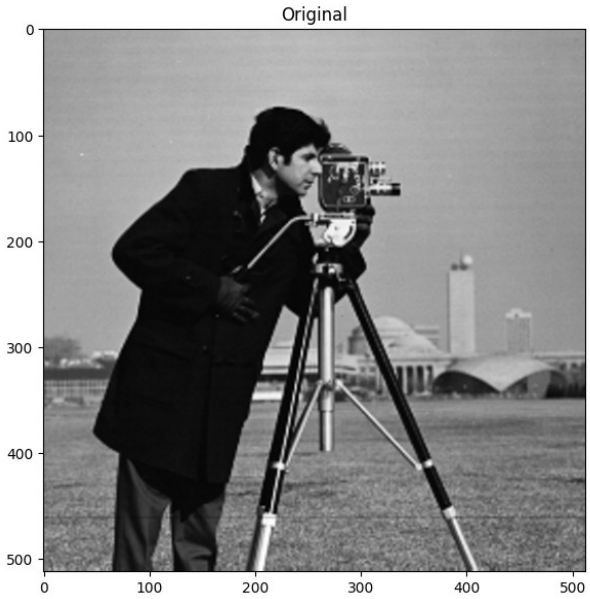


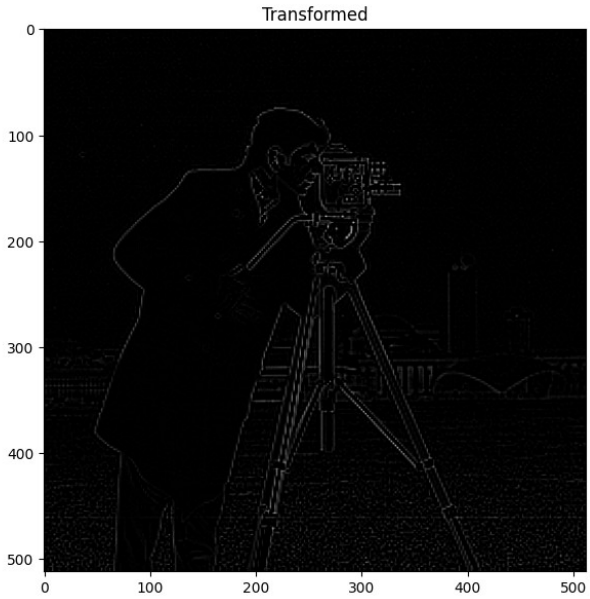
Utilizando o a mascara laplaciana com openCV

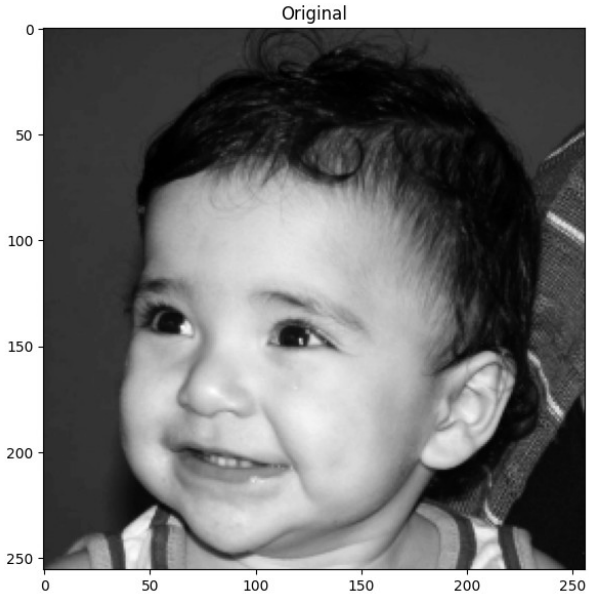
```
for img in images:
    process(img, opencv_convolve, get_laplacian_kernel())
```

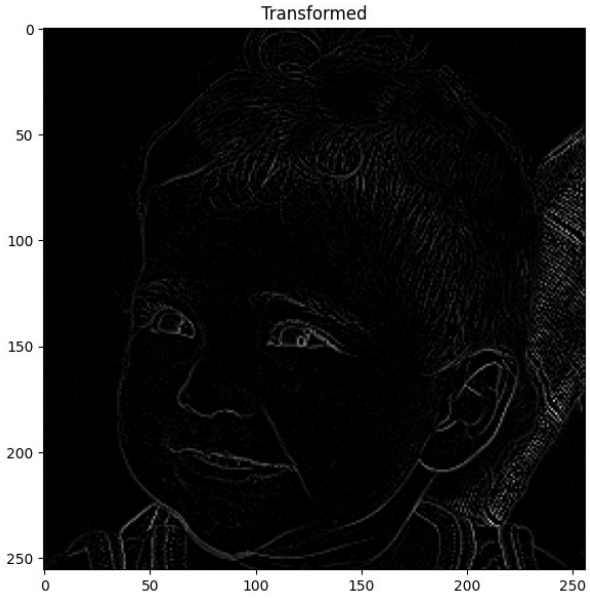

Original
0
100
200
300
400
500
0
100
200
300
400
500


Transformed
0
100
200
300
400
500
0
100
200
300
400
500


Original
0
100
200
300
400
500
0
100
200
300
400
500

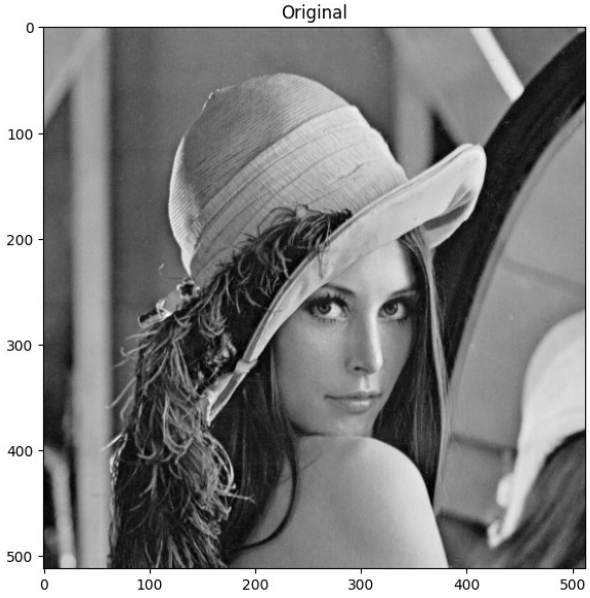

Transformed
0
100
200
300
400
500
0
100
200
300
400
500

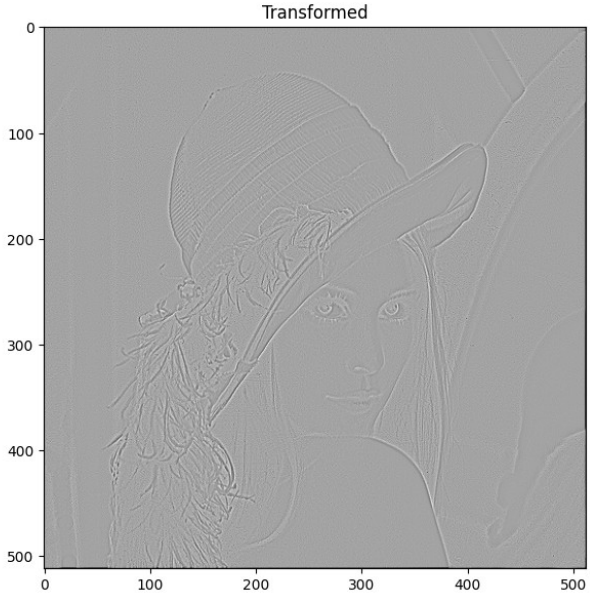


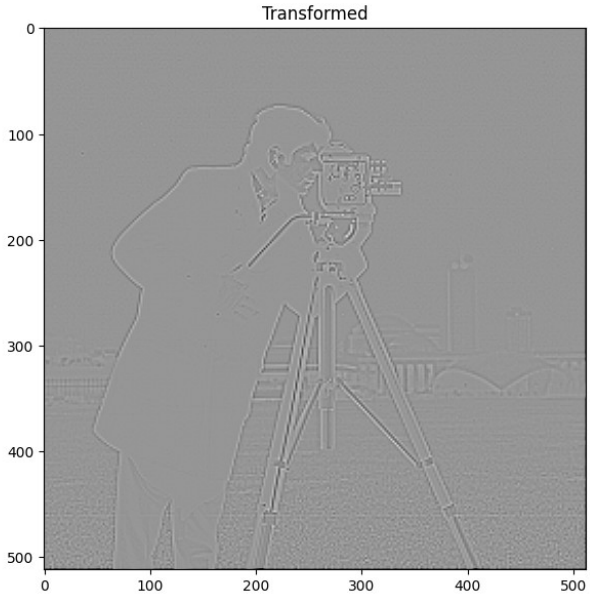


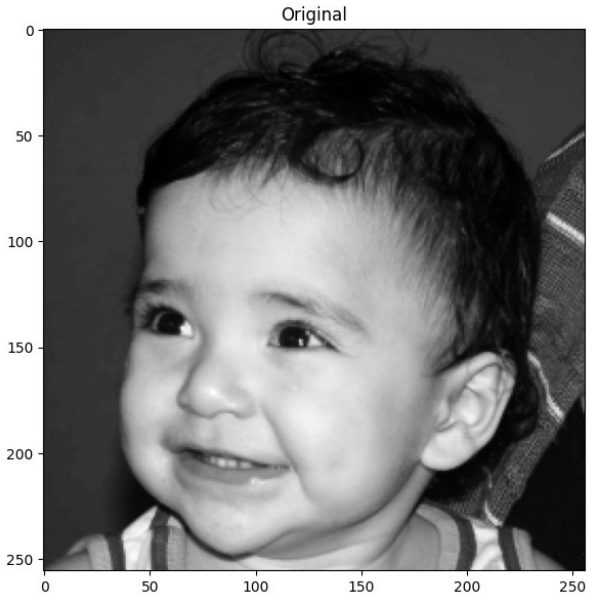
Utilizando o a mascara laplaciana com scipy

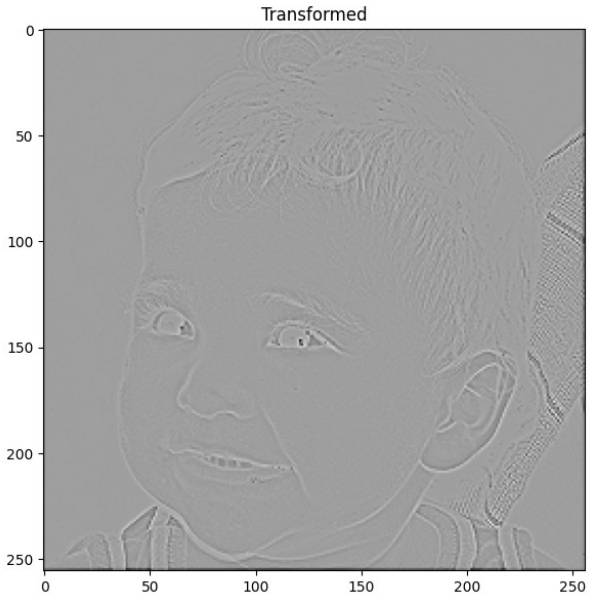
```
for img in images:
    process(img, scipy_convolve, get_laplacian_kernel())
```





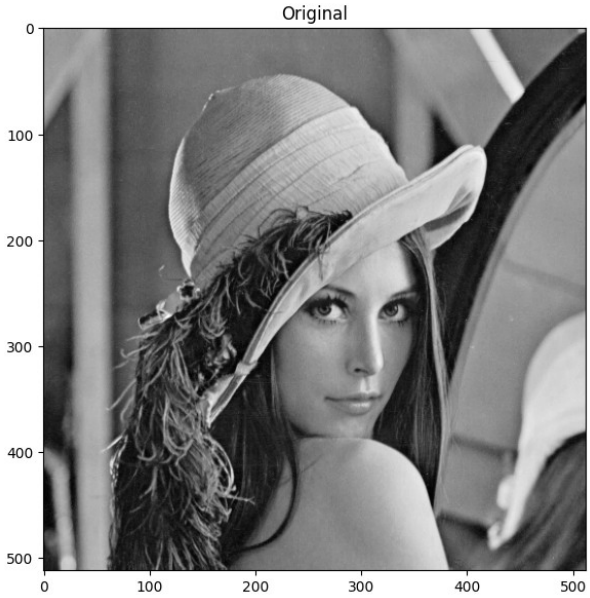


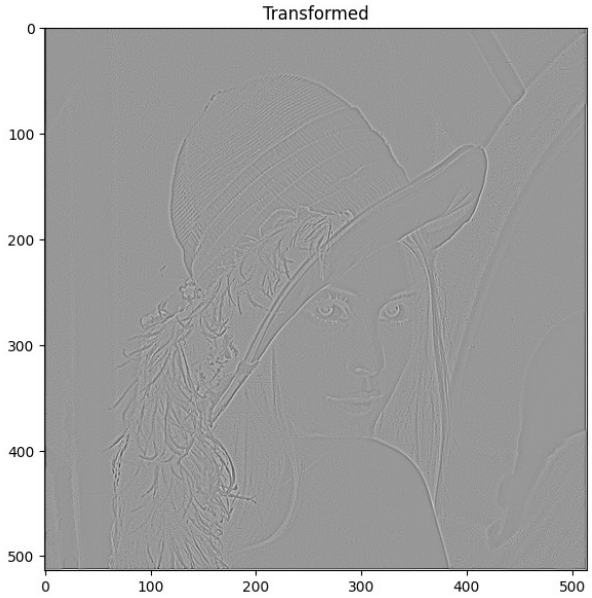




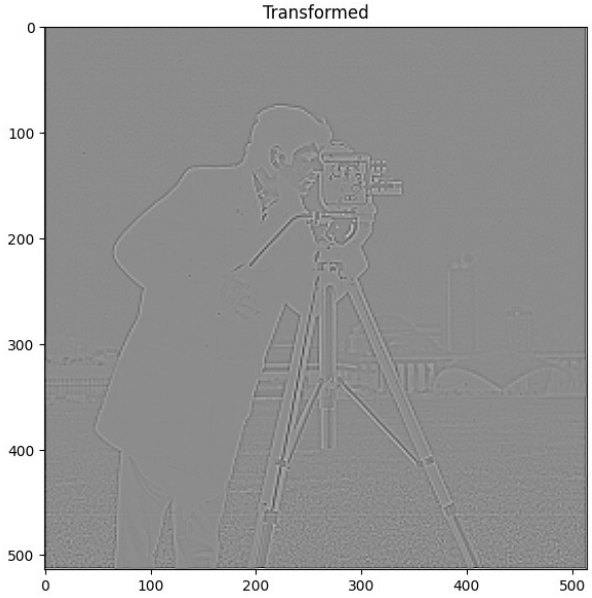
Utilizando o a mascara laplaciana com implementação manual

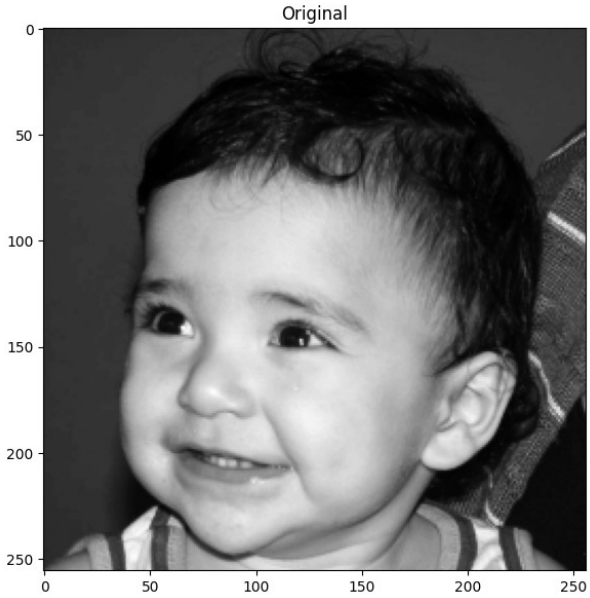
```
for img in images:
    process(img, convolve, get_laplacian_kernel())
```

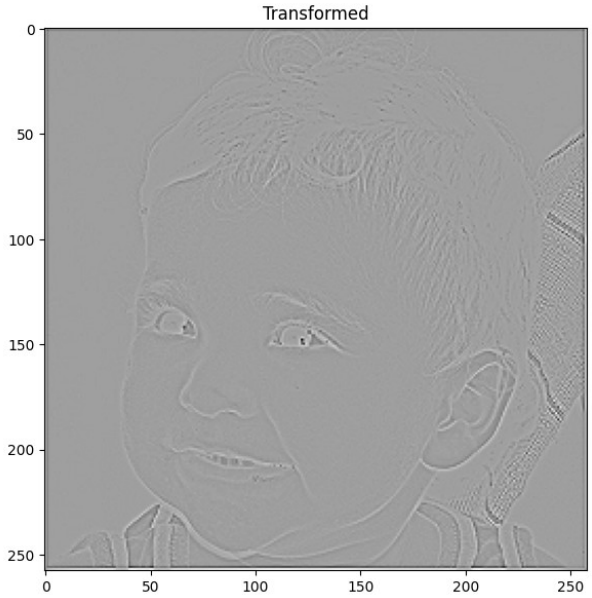

Original
0
100
200
300
400
500
0
100
200
300
400
500


Transformed
0
100
200
300
400
500
0
100
200
300
400
500


Original
0
100
200
300
400
500
0
100
200
300
400
500

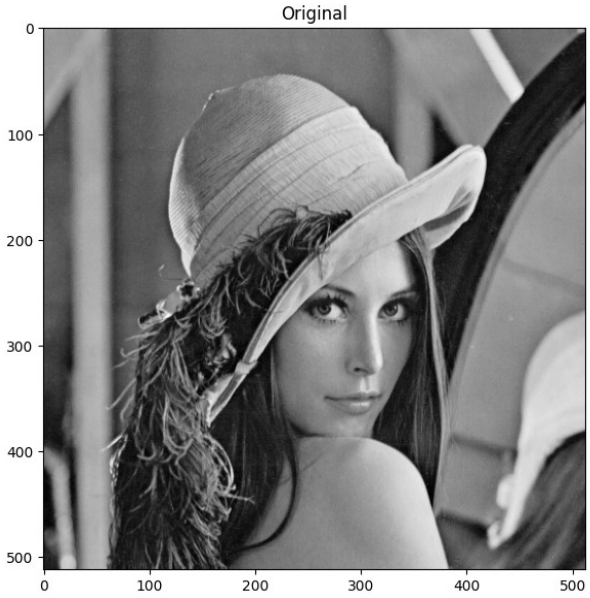

Transformed
0
100
200
300
400
500
0
100
200
300
400
500

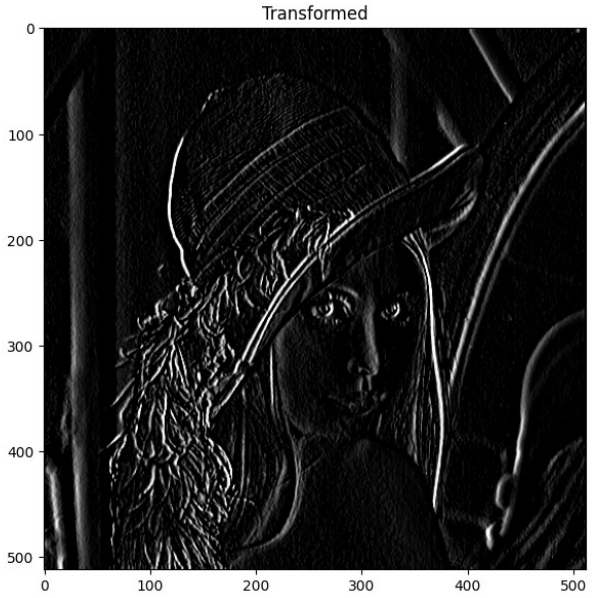


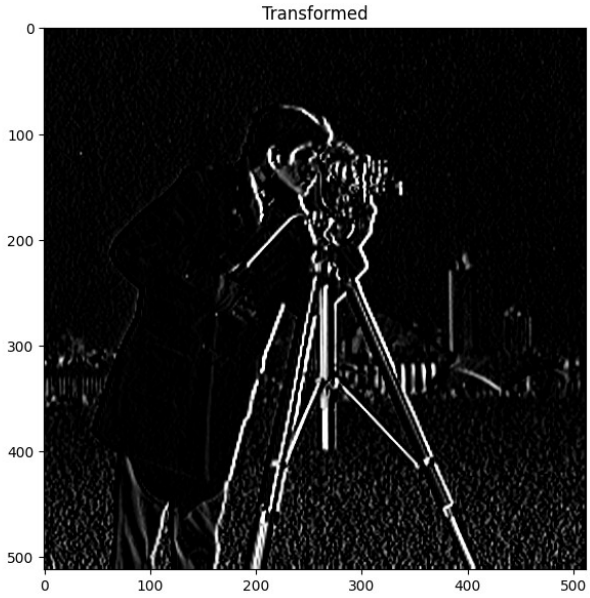


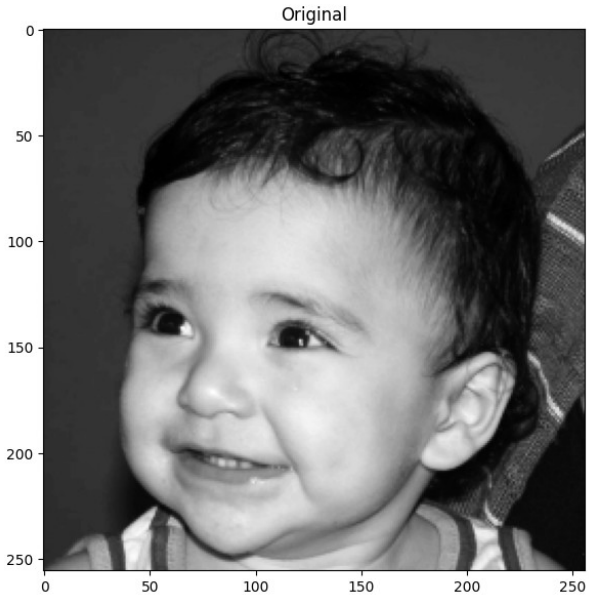
Utilizando o a mascara sobel x com openCV

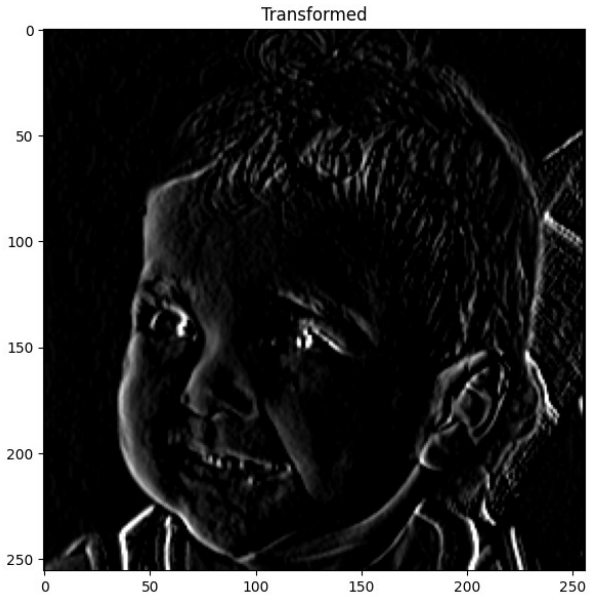
```
for img in images:
    process(img, opencv_convolve, get_sobel_x_kernel())
```





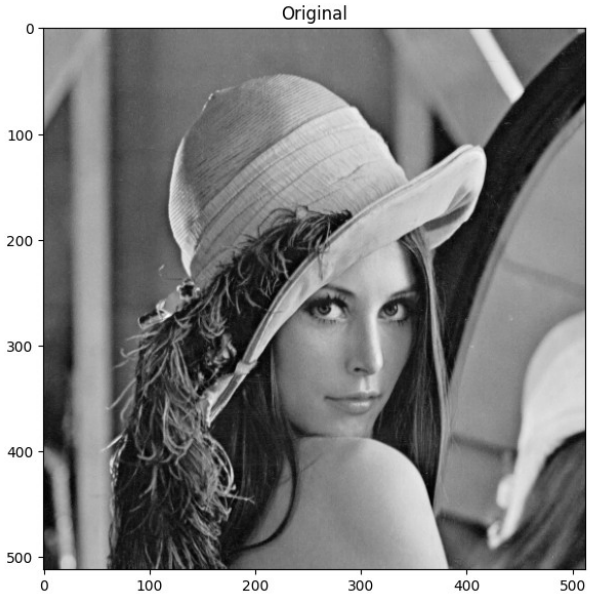


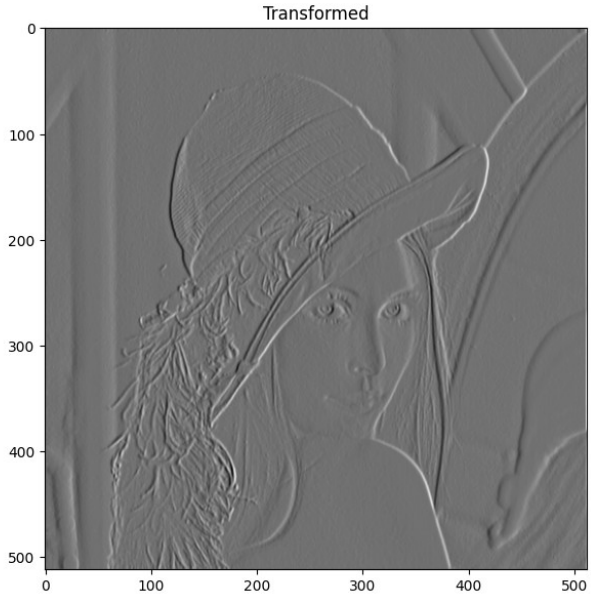


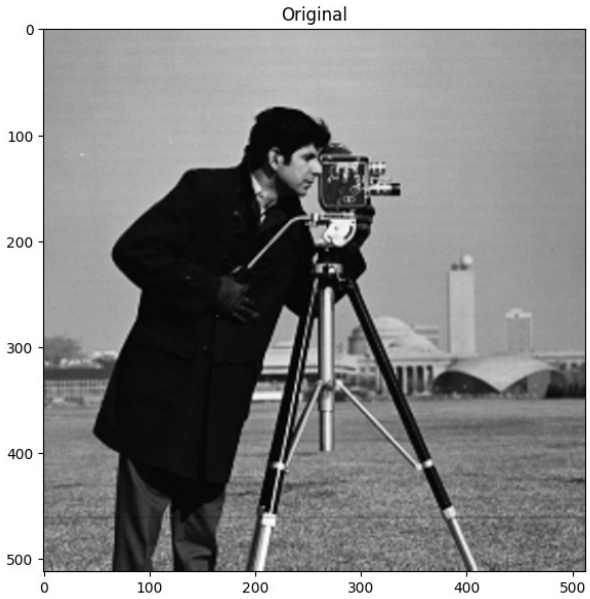


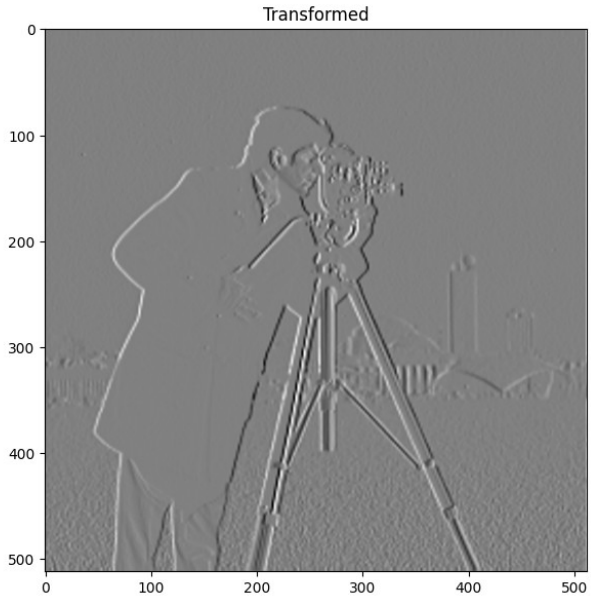
Utilizando o a mascara sobel x com scipy

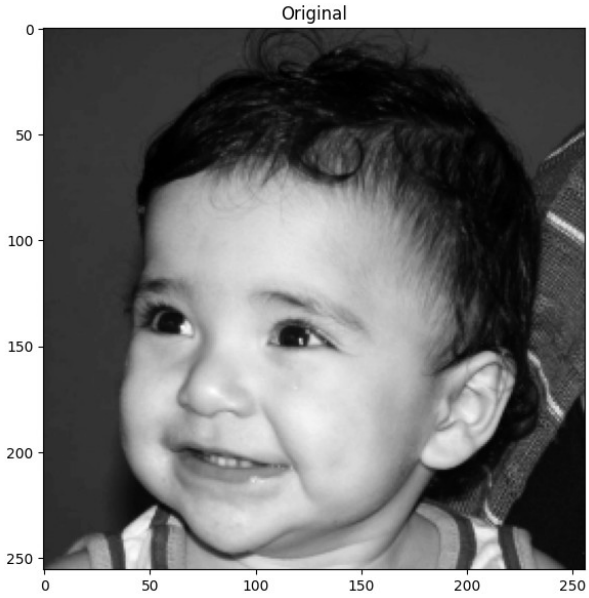
```
for img in images:
    process(img, scipy_convolve, get_sobel_x_kernel())
```

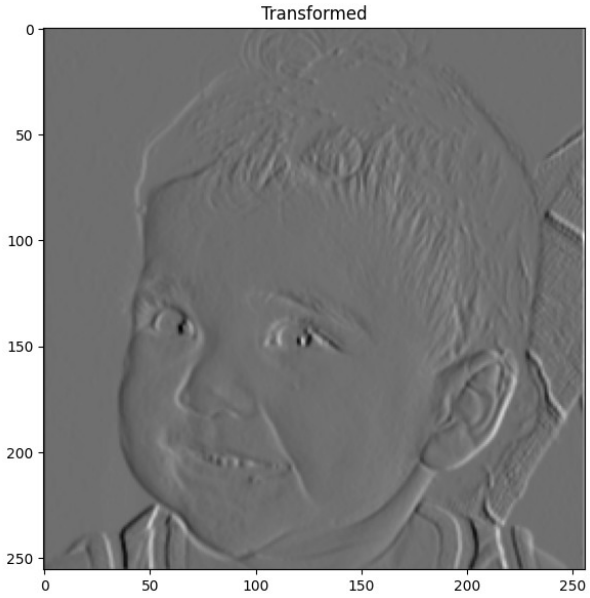

Original
0
100
200
300
400
500
0
100
200
300
400
500


Transformed
0
100
200
300
400
500
0
100
200
300
400
500


Original
0
100
200
300
400
500
0
100
200
300
400
500

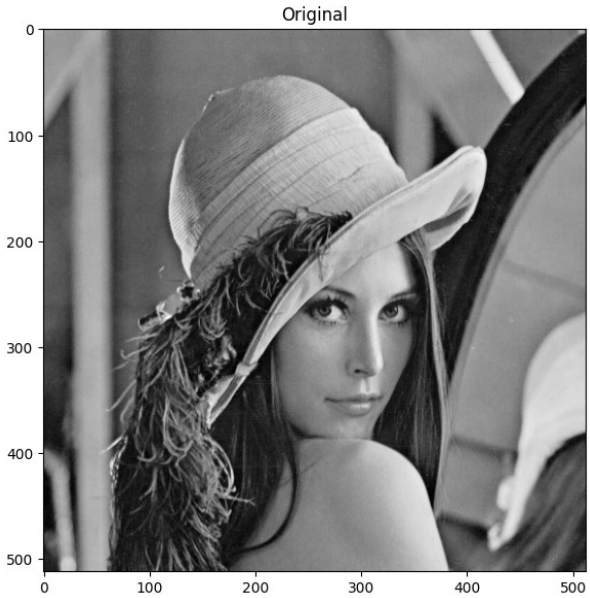

Transformed
0
100
200
300
400
500
0
100
200
300
400
500

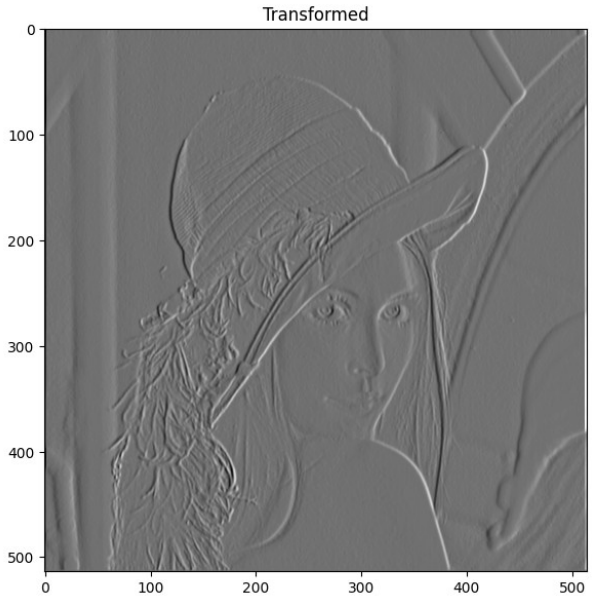


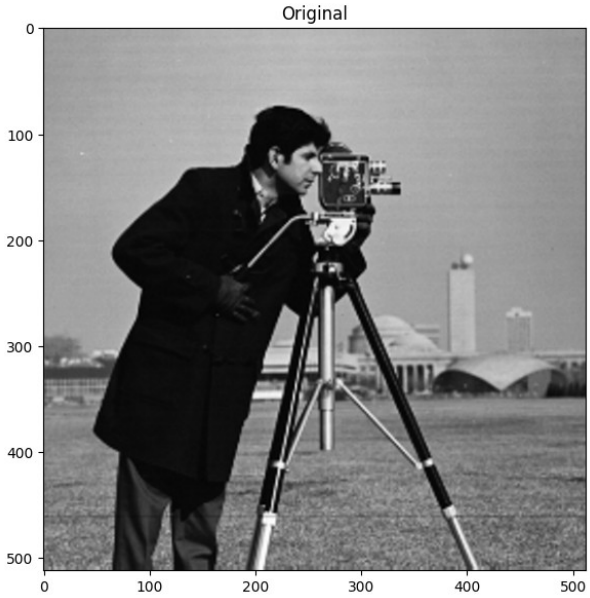


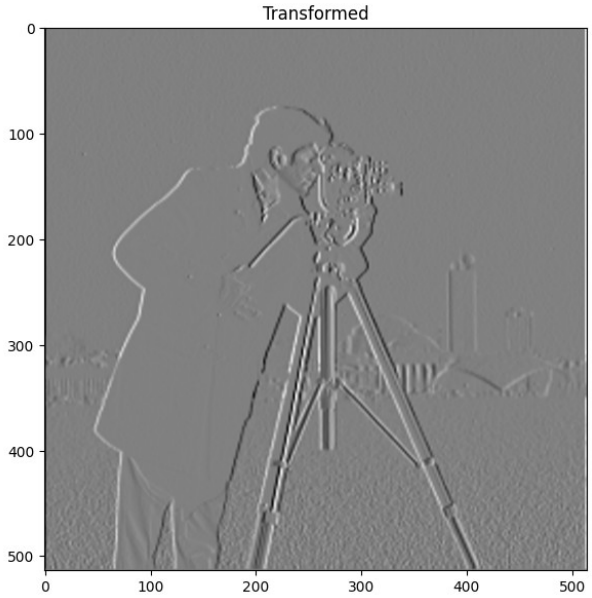
Utilizando o a mascara sobel x com implementação manual

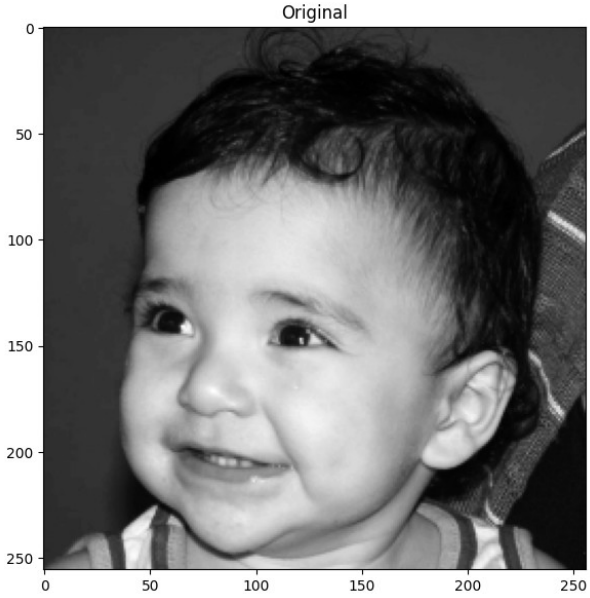
```
for img in images:
    process(img, convolve, get_sobel_x_kernel())
```

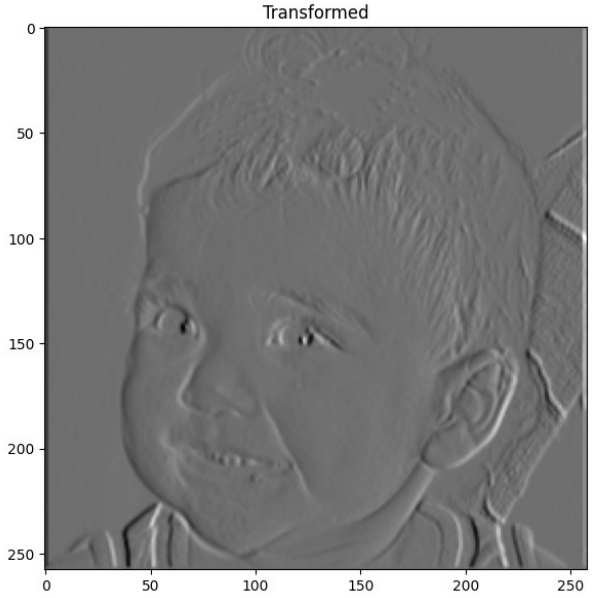






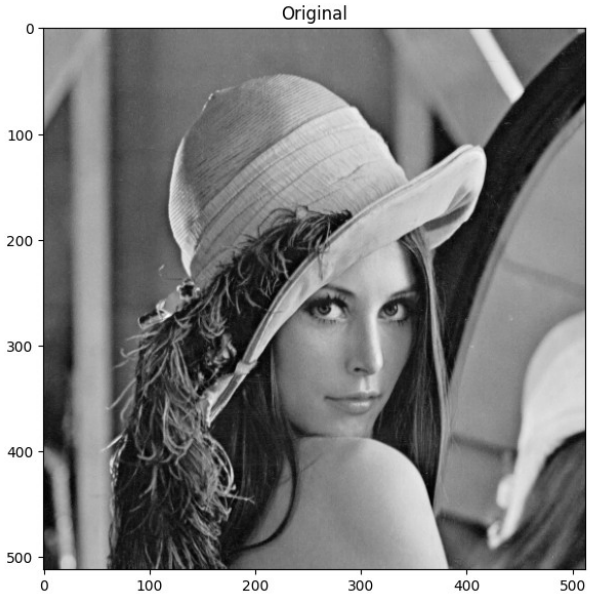


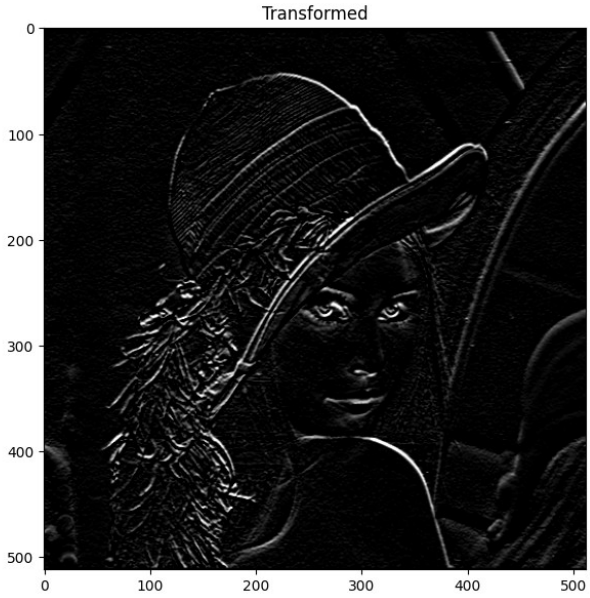


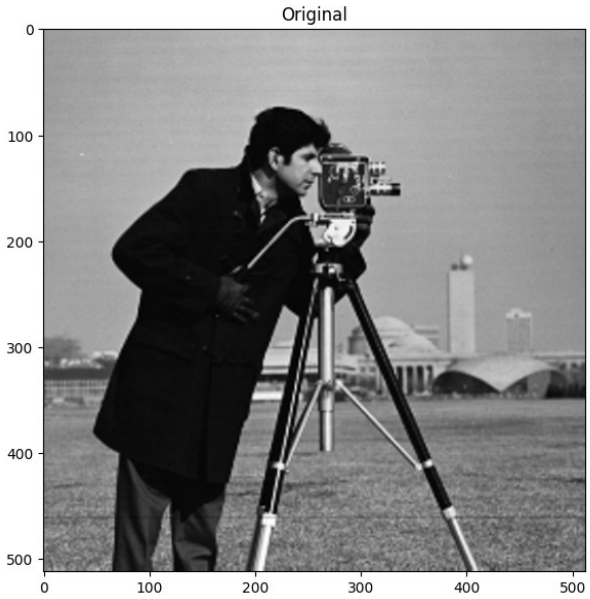


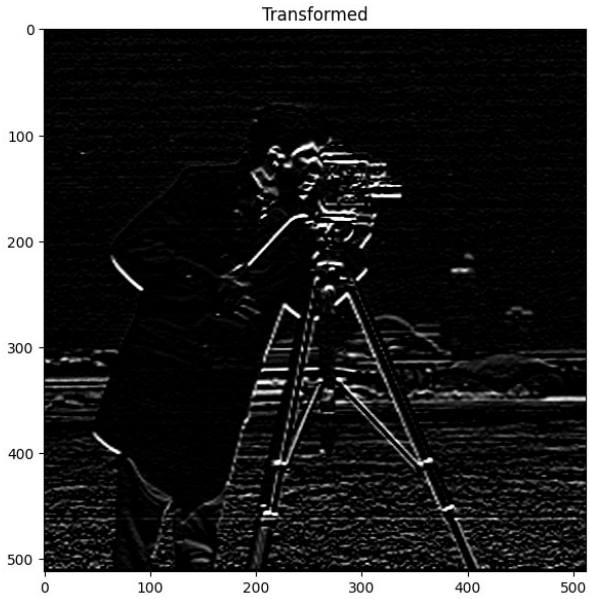
Utilizando o a mascara sobel y com openCV

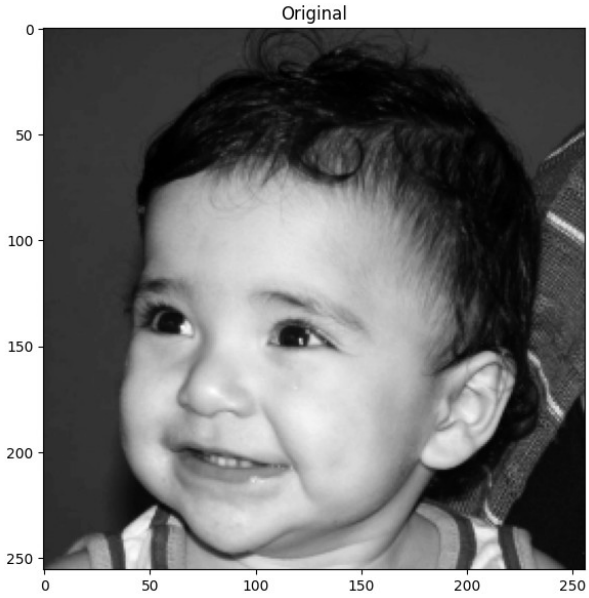
```
for img in images:
    process(img, opencv_convolve, get_sobel_y_kernel())
```

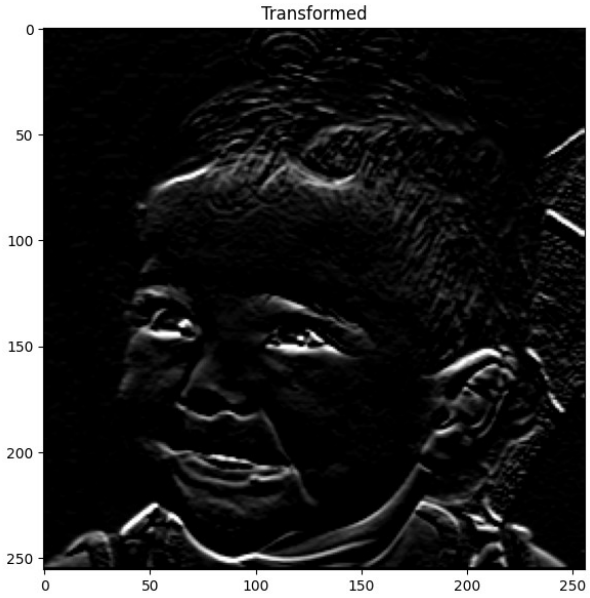

Original
0
100
200
300
400
500


Transformed
0
100
200
300
400
500


Original
0
100
200
300
400
500

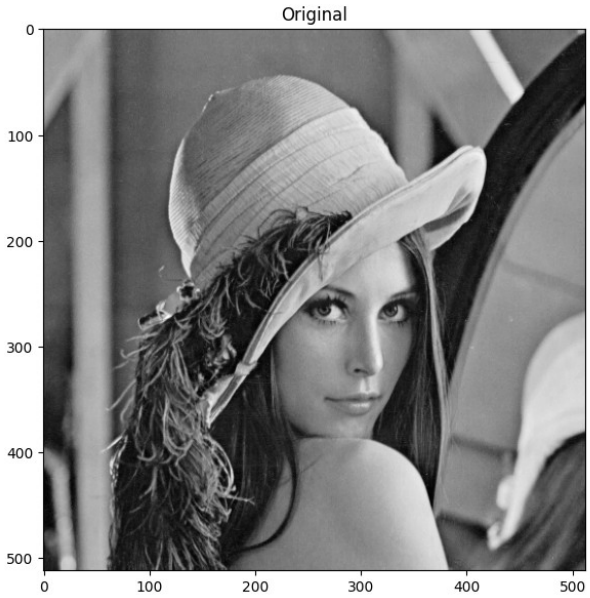

Transformed
0
100
200
300
400
500

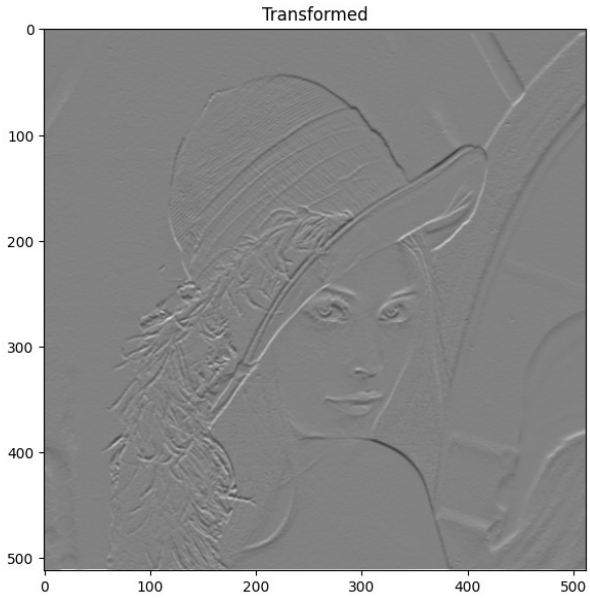


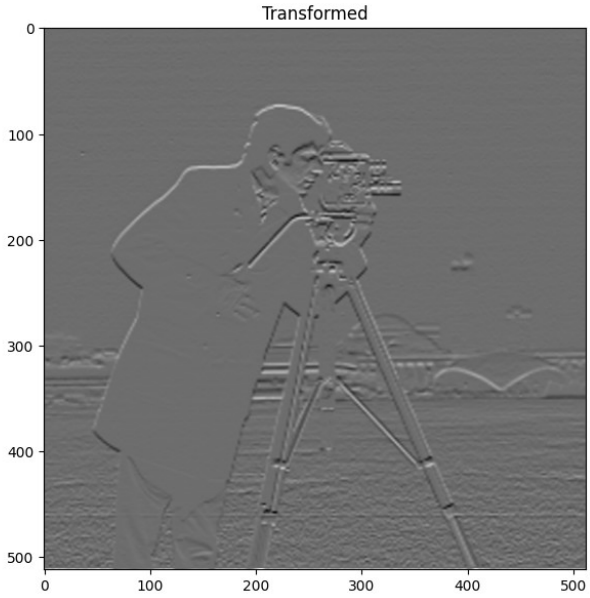


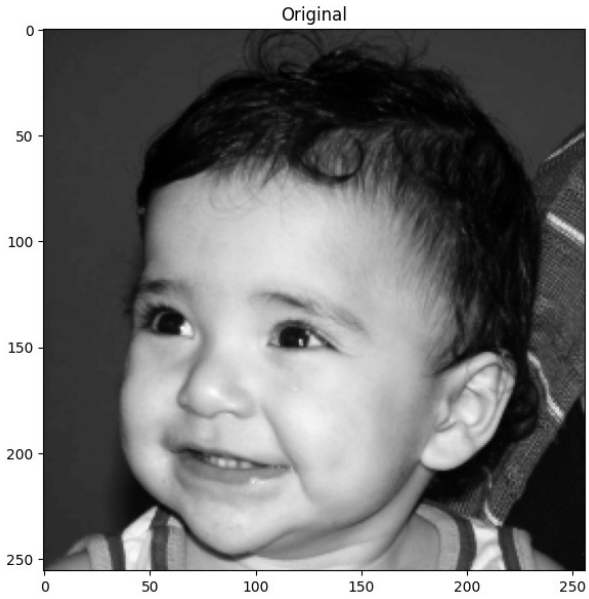
Utilizando o a mascara sobel y com scipy

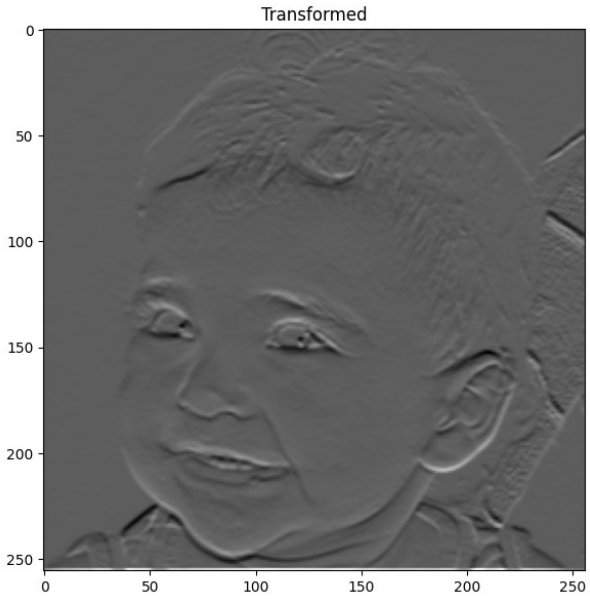
```
for img in images:
    process(img, scipy_convolve, get_sobel_y_kernel())
```





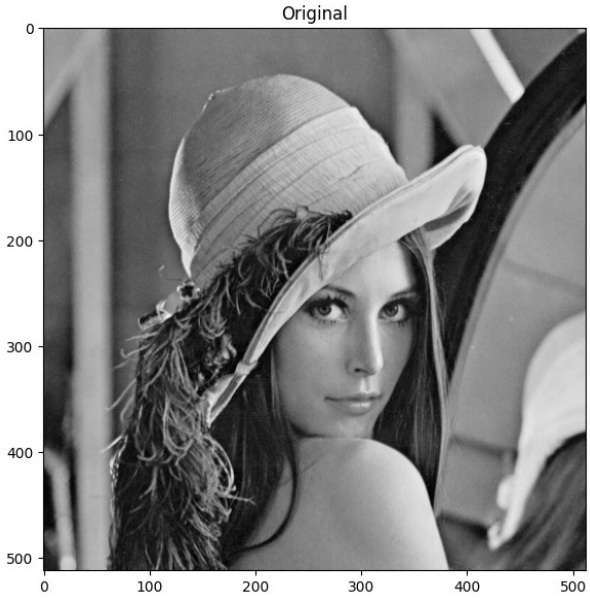


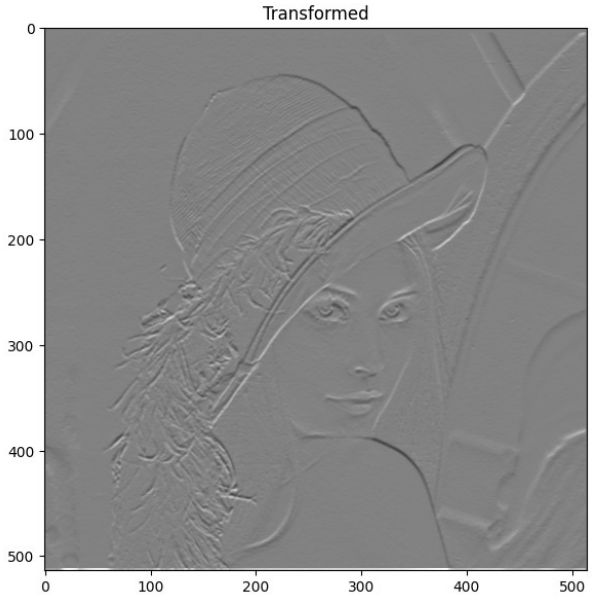




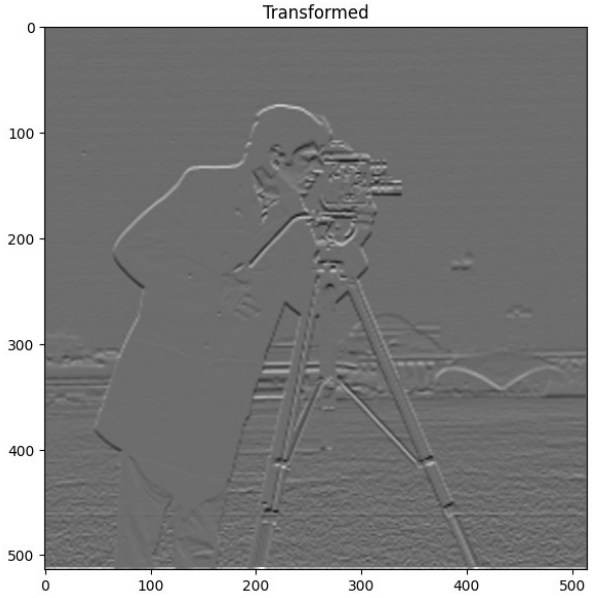
Utilizando o a mascara sobel y com implementação manual

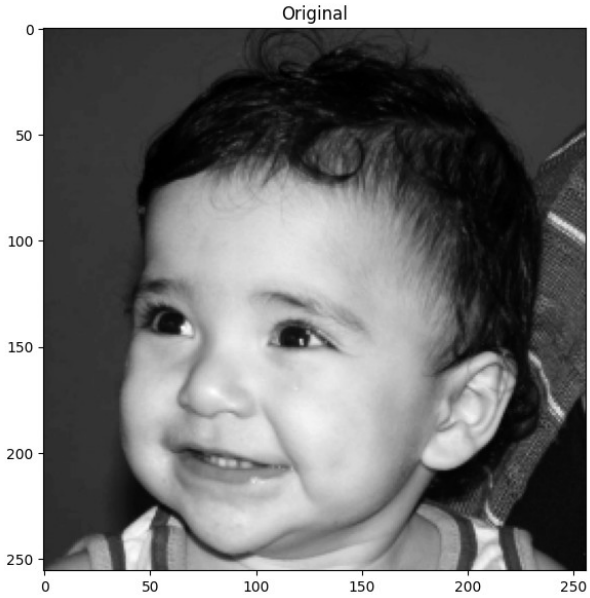
```
for img in images:
    process(img, convolve, get_sobel_y_kernel())
```

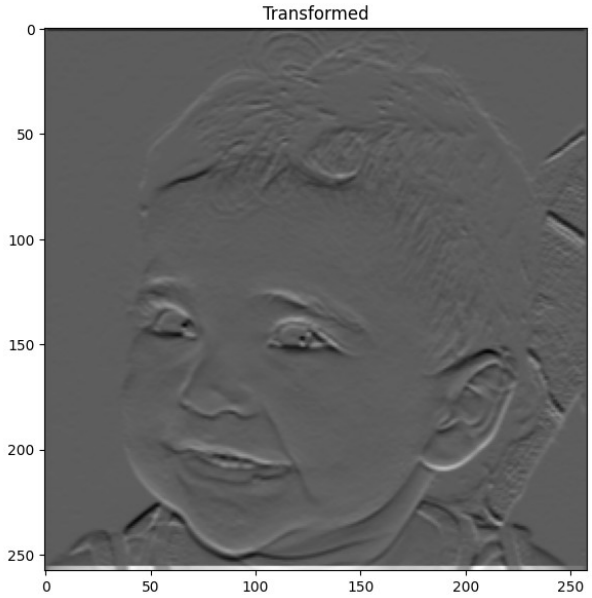

Original
0
100
200
300
400
500
0
100
200
300
400
500


Transformed
0
100
200
300
400
500
0
100
200
300
400
500


Original
0
100
200
300
400
500
0
100
200
300
400
500

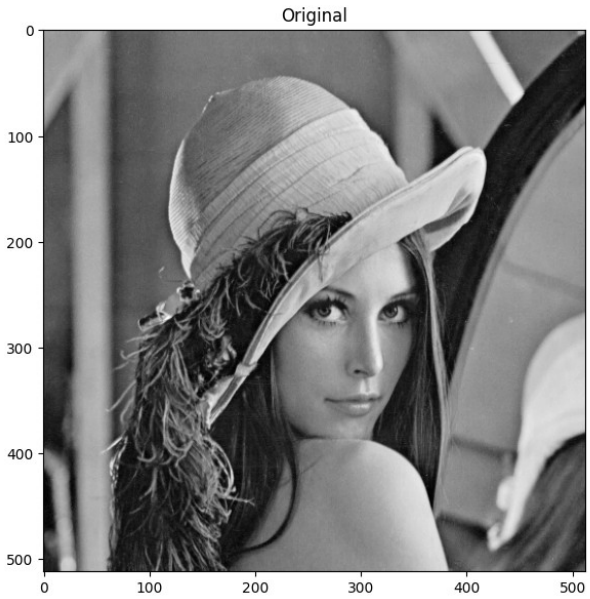

Transformed
0
100
200
300
400
500
0
100
200
300
400
500

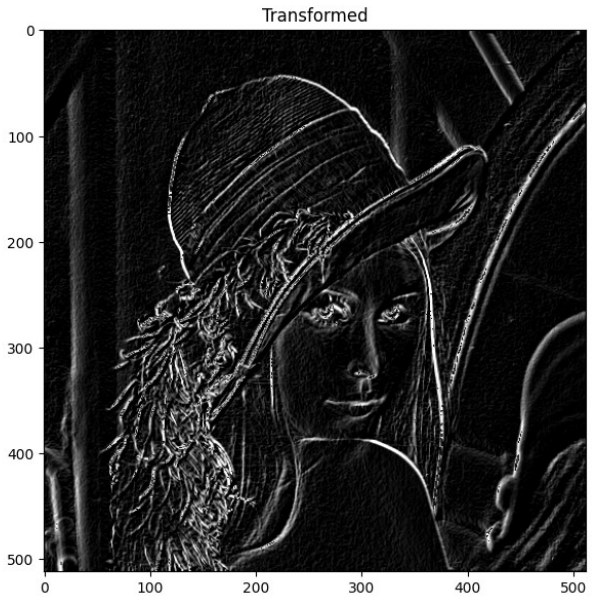


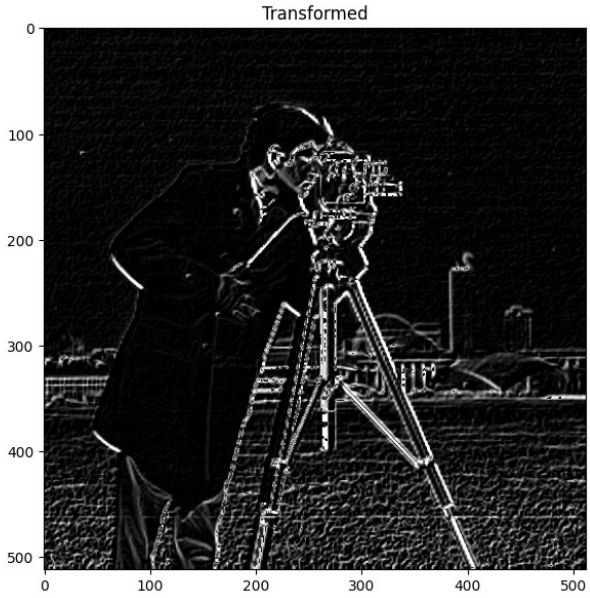


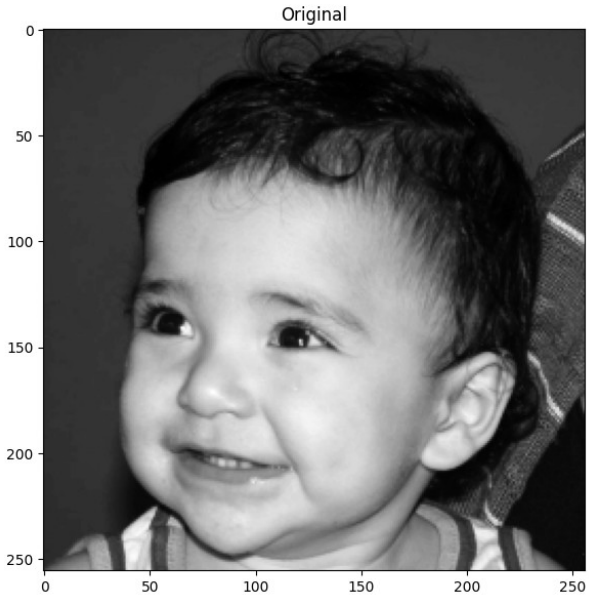
Utilizando o a mascara gradiente com openCV

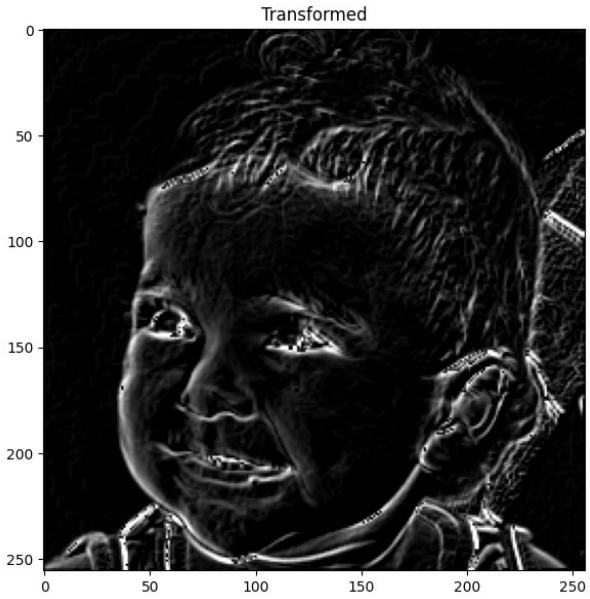
```
for img in images:
    process(img,
            lambda img:
            opencv_convolve(img, get_sobel_x_kernel()) +
            opencv_convolve(img, get_sobel_y_kernel())
            )
```





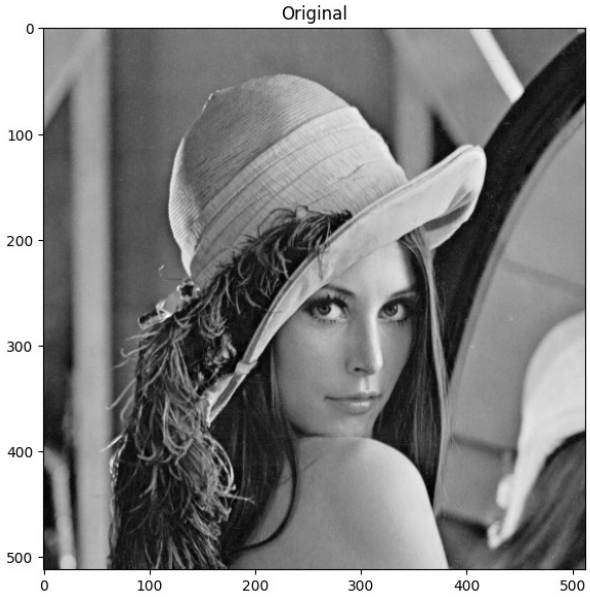


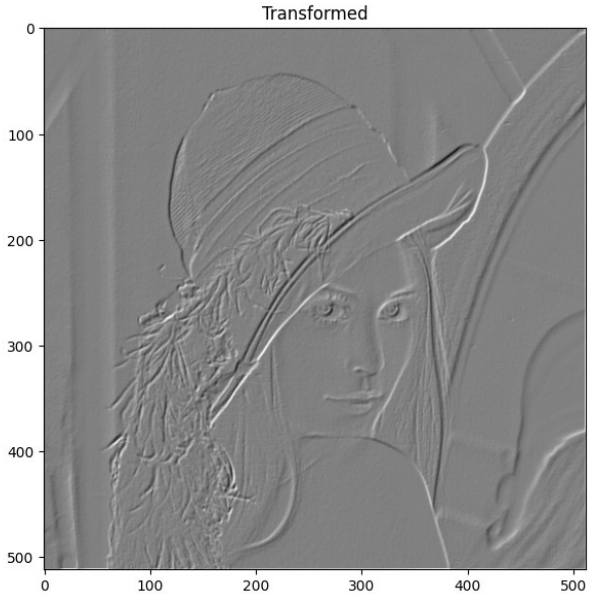


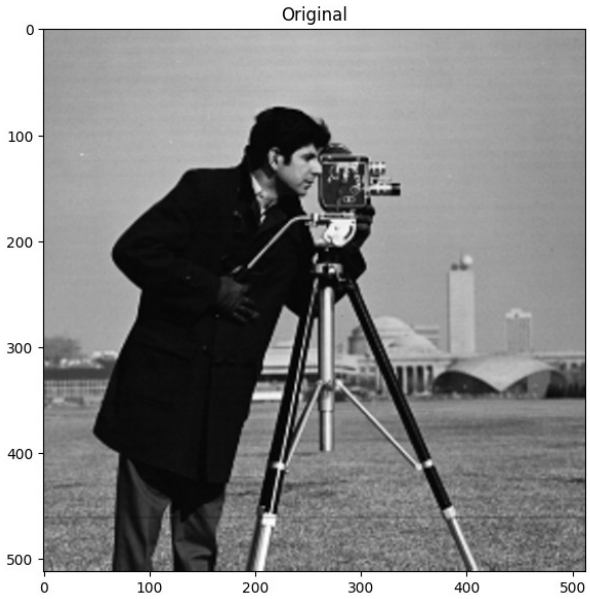


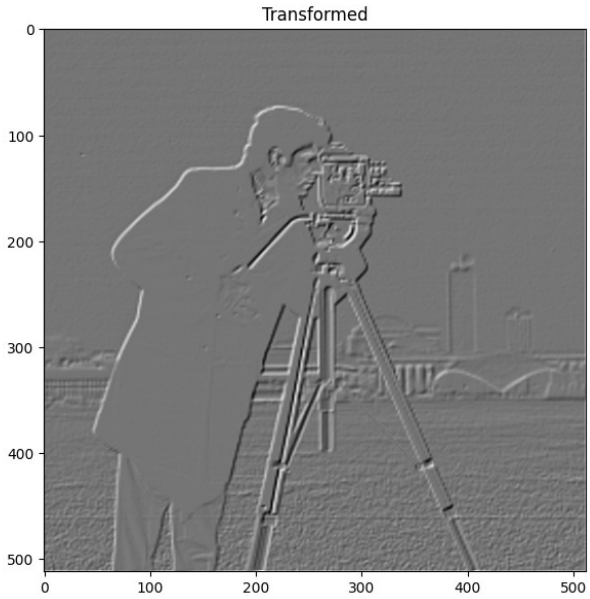
Utilizando a mascara gradiente com scipy

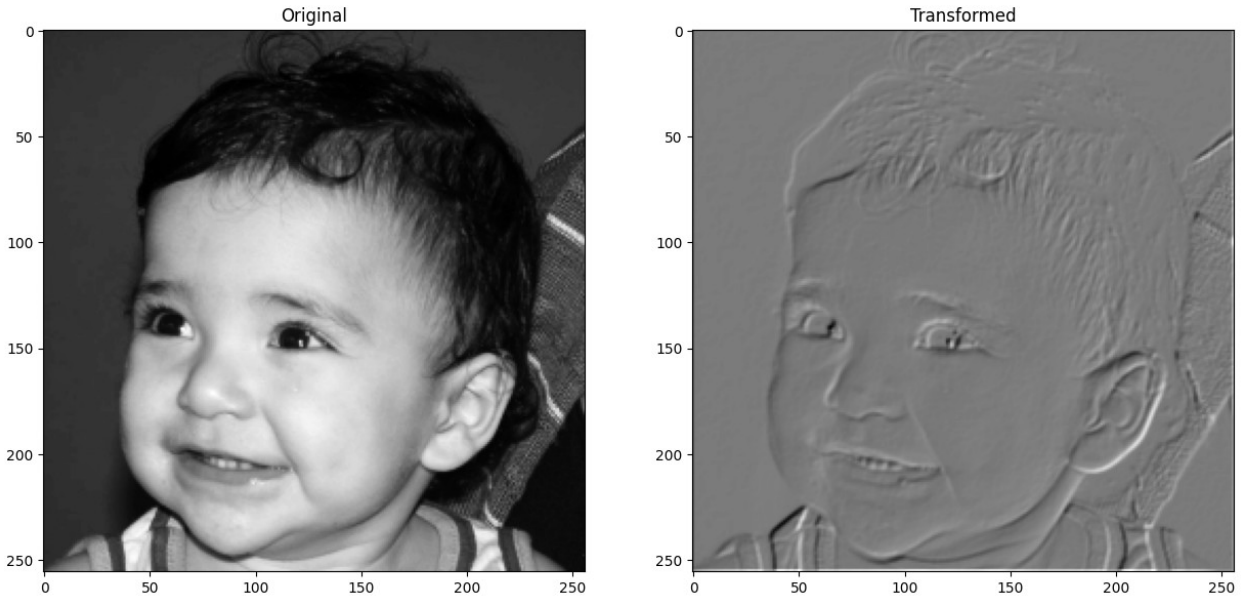
```
for img in images:
    process(img,
            lambda img:
            scipy_convolve(img, get_sobel_x_kernel()) +
            scipy_convolve(img, get_sobel_y_kernel())
            )
```


Original
0
100
200
300
400
500
0
100
200
300
400
500


Transformed
0
100
200
300
400
500
0
100
200
300
400
500

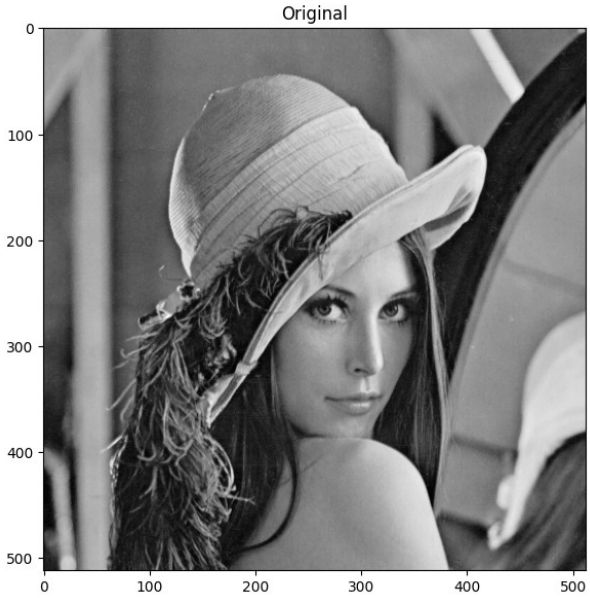

Original
0
100
200
300
400
500
0
100
200
300
400
500

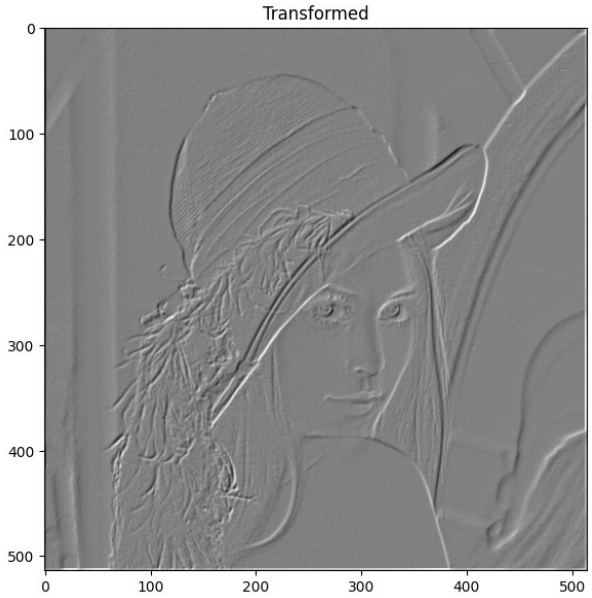

Transformed
0
100
200
300
400
500
0
100
200
300
400
500



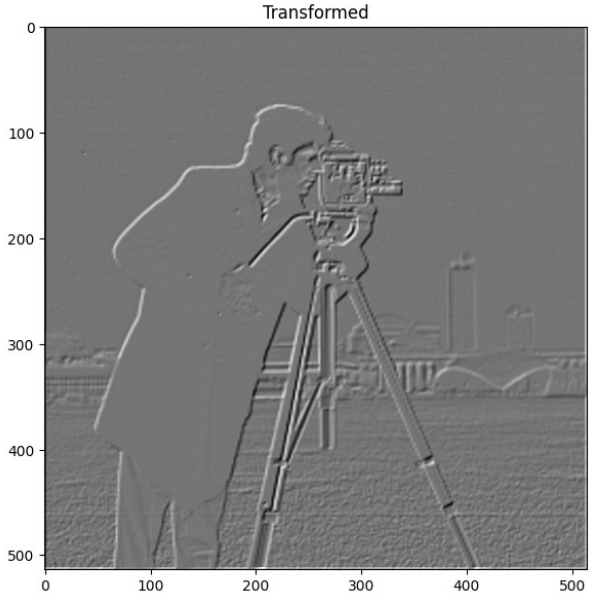
Utilizando o a mascara gradiente com implementação manual

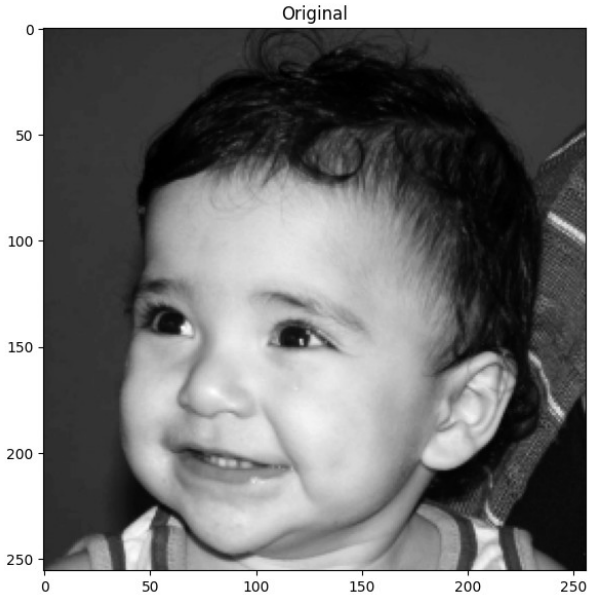
```
def gradiente_convolve(img):
    img1 = np.array(convolve(img, get_sobel_x_kernel()))
    img2 = np.array(convolve(img, get_sobel_y_kernel()))

    return img1 + img2


for img in images:
    process(img, gradiente_convolve)
```


Original
0
100
200
300
400
500
0
100
200
300
400
500


Transformed
0
100
200
300
400
500
0
100
200
300
400
500

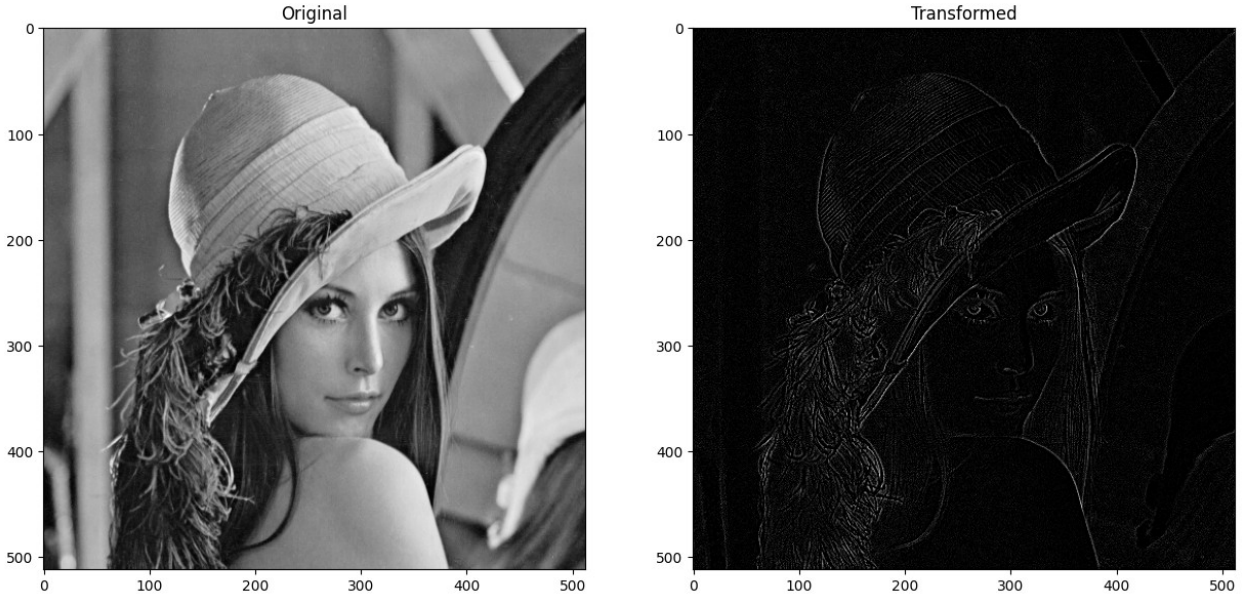

Original
0
100
200
300
400
500
0
100
200
300
400
500

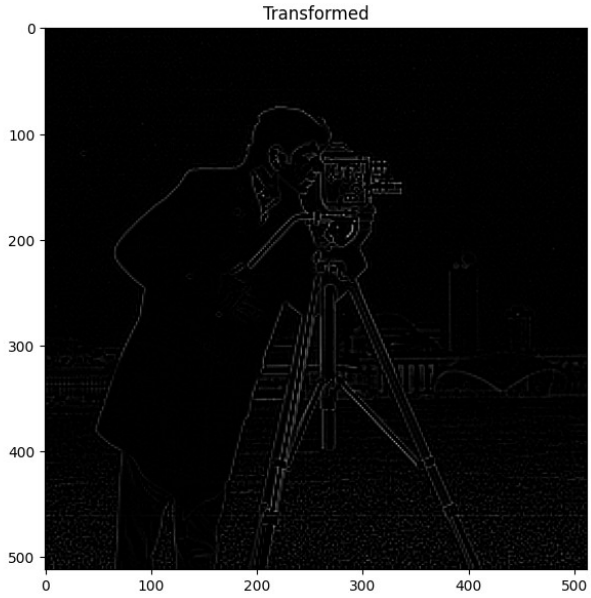

Transformed
0
100
200
300
400
500
0
100
200
300
400
500

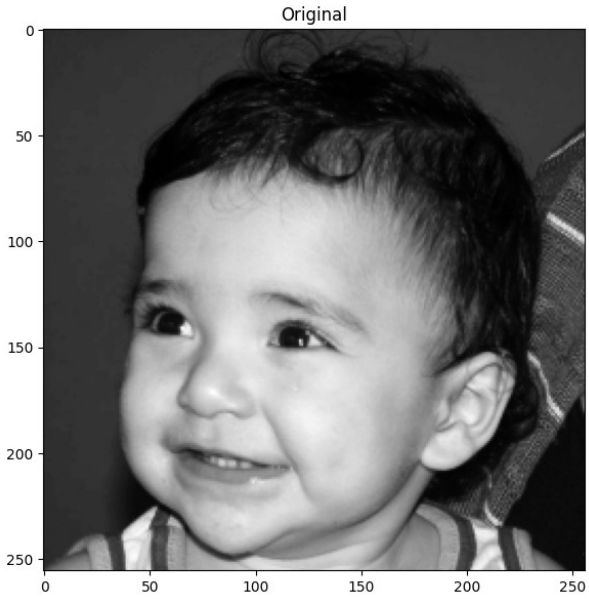


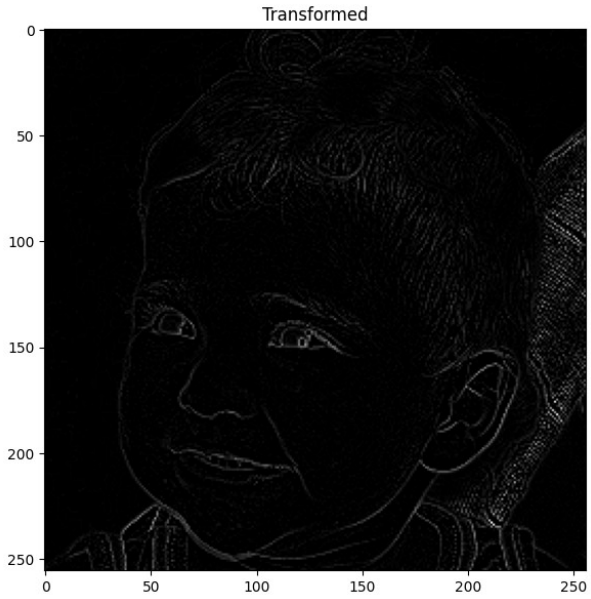
Utilizando o a mascara laplaciana somada a imagem original com openCV

```
for img in images:
    process(img, opencv_convolve, get_laplacian_sum_kernel(img))
```



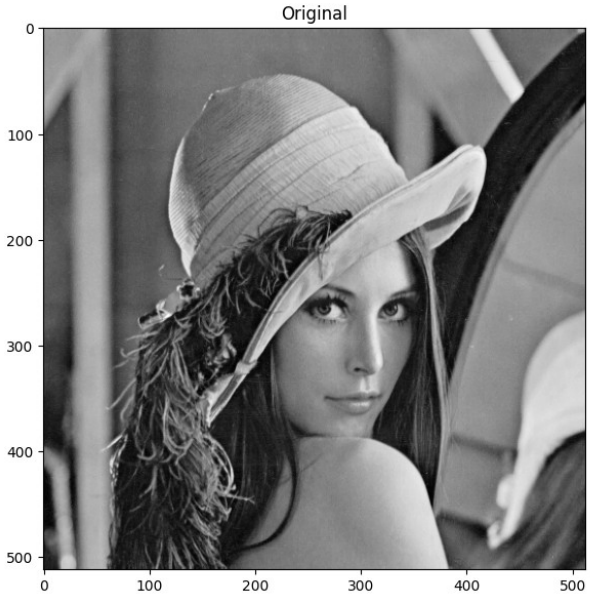


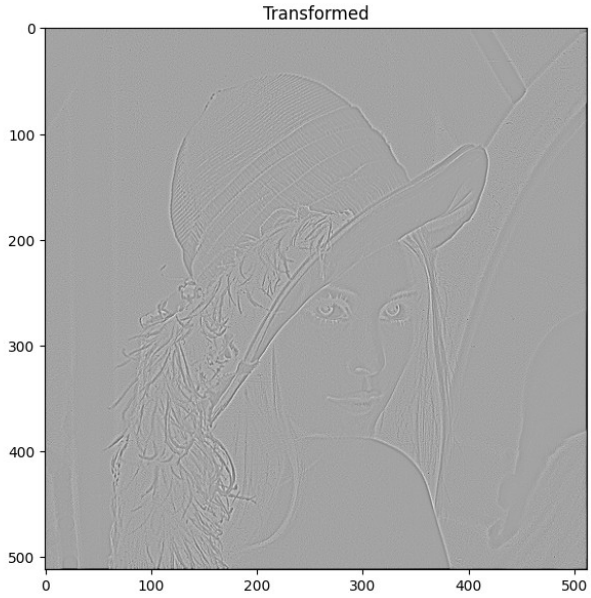


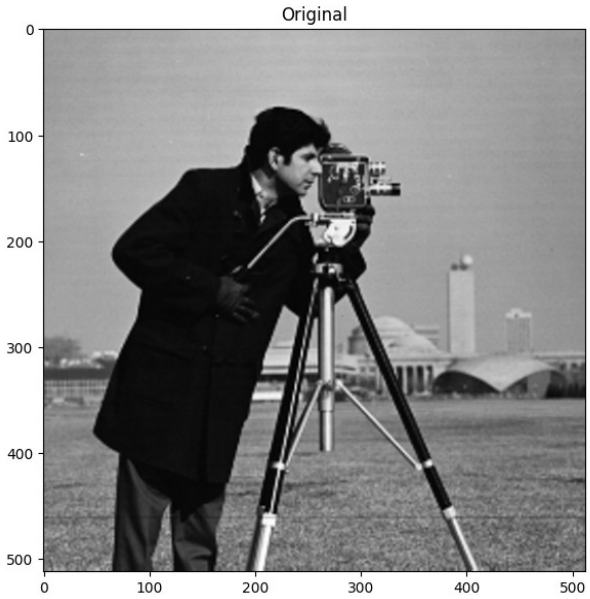


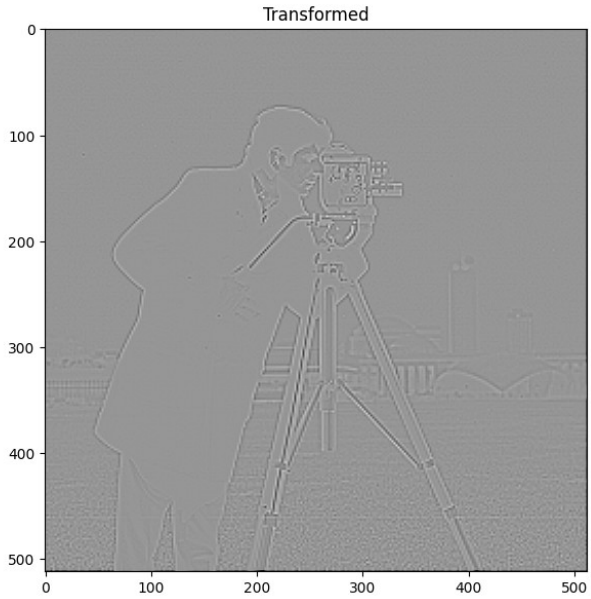
Utilizando o a mascara laplaciana somada a imagem original com scipy

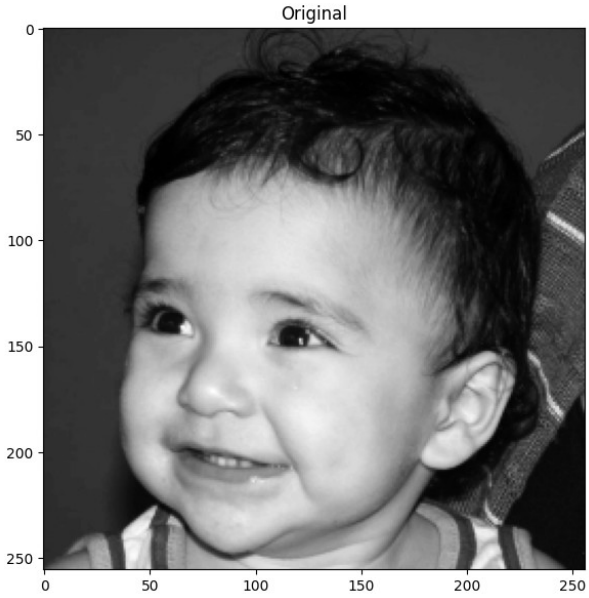
```
for img in images:
    process(img, scipy_convolve, get_laplacian_sum_kernel(img))
```

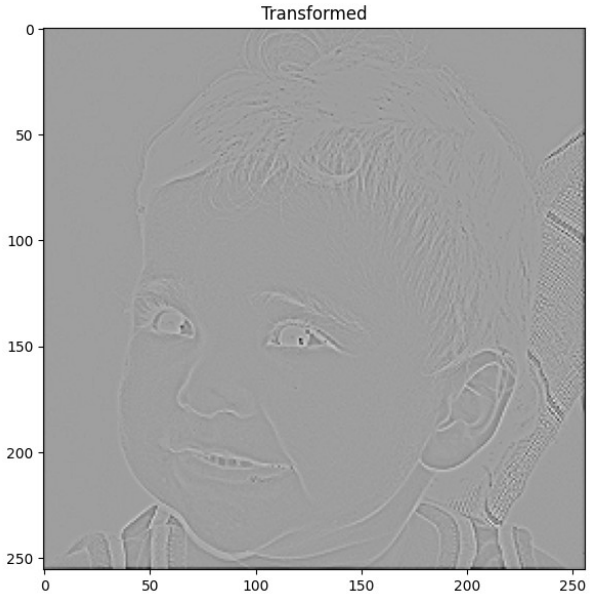

Original
0
100
200
300
400
500
0
100
200
300
400
500


Transformed
0
100
200
300
400
500
0
100
200
300
400
500


Original
0
100
200
300
400
500
0
100
200
300
400
500

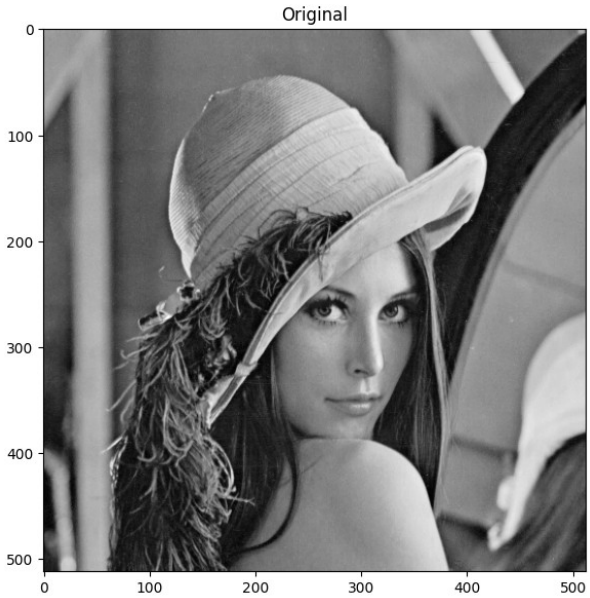

Transformed
0
100
200
300
400
500
0
100
200
300
400
500

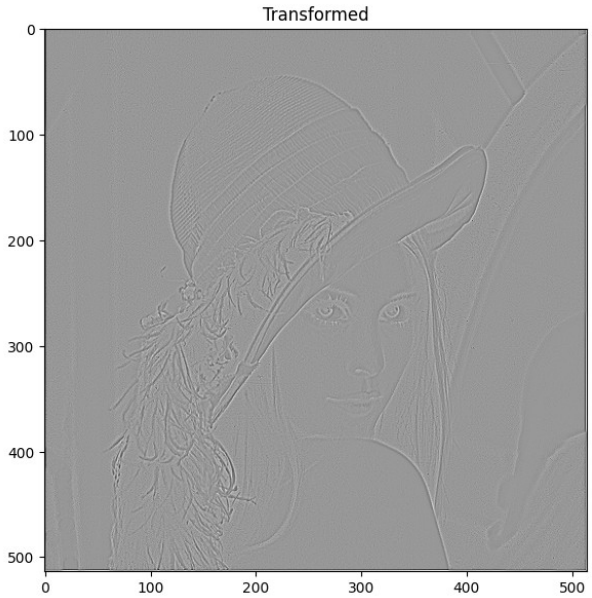


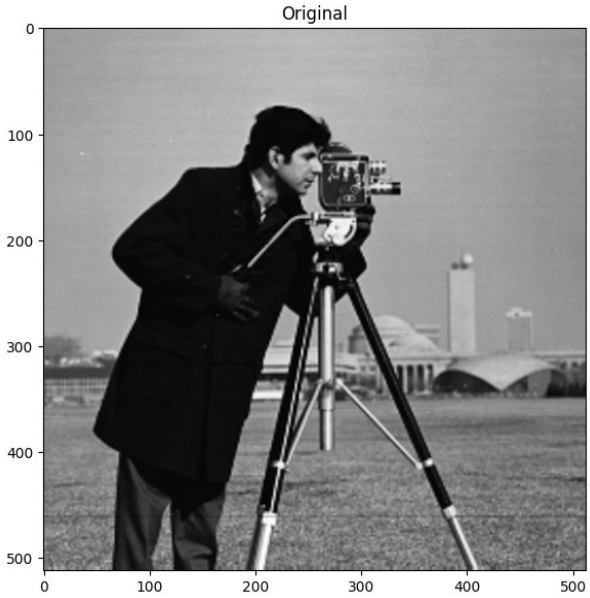


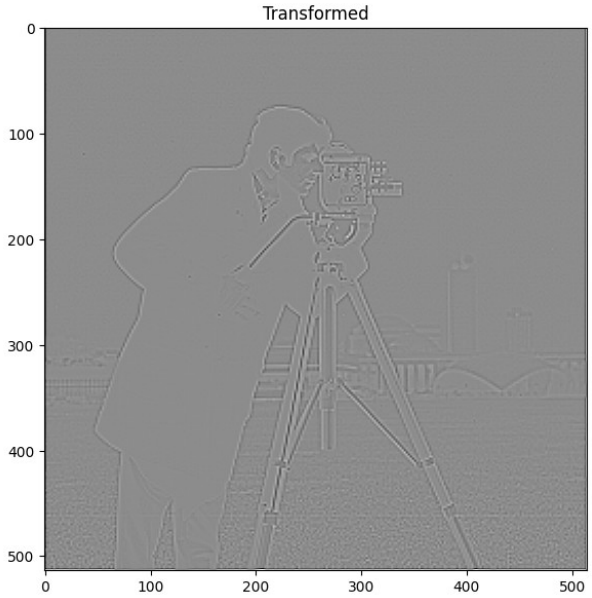
Utilizando o a mascara laplaciana somada a imagem original com implementação manual

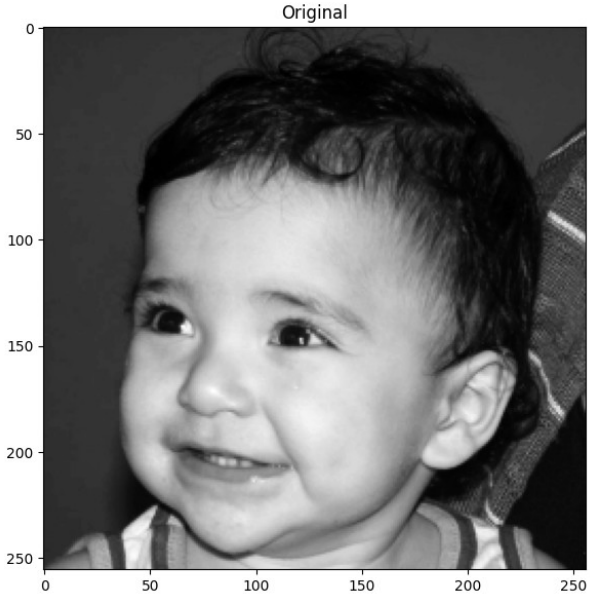
```
for img in images:
    process(img, convolve, get_laplacian_sum_kernel(img))
```

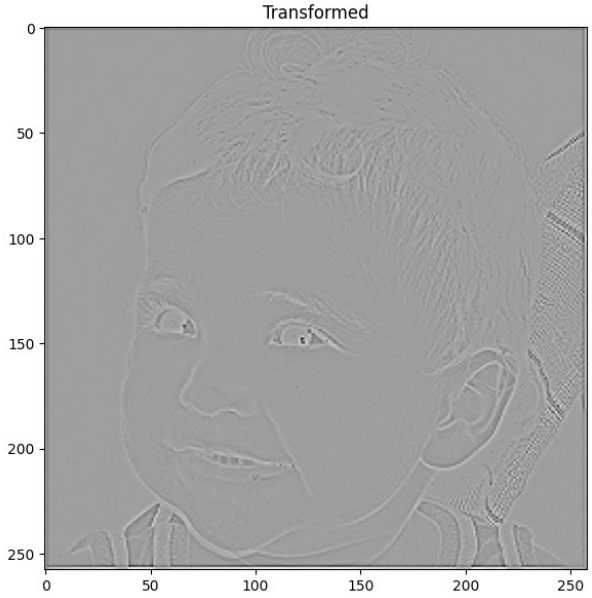





Original
0
100
200
300
400
500
0
100
200
300
400
500


Transformed
0
100
200
300
400
500
0
100
200
300
400
500


Original
0
50
100
150
200
250
0
50
100
150
200
250


Transformed
0
50
100
150
200
250
0
50
100
150
200
250