edge_detection

October 6, 2024

1 Deteksi Tepi

Nama: Ronggo Widjoyo NIM: 220411100061 Kelas: PCD A

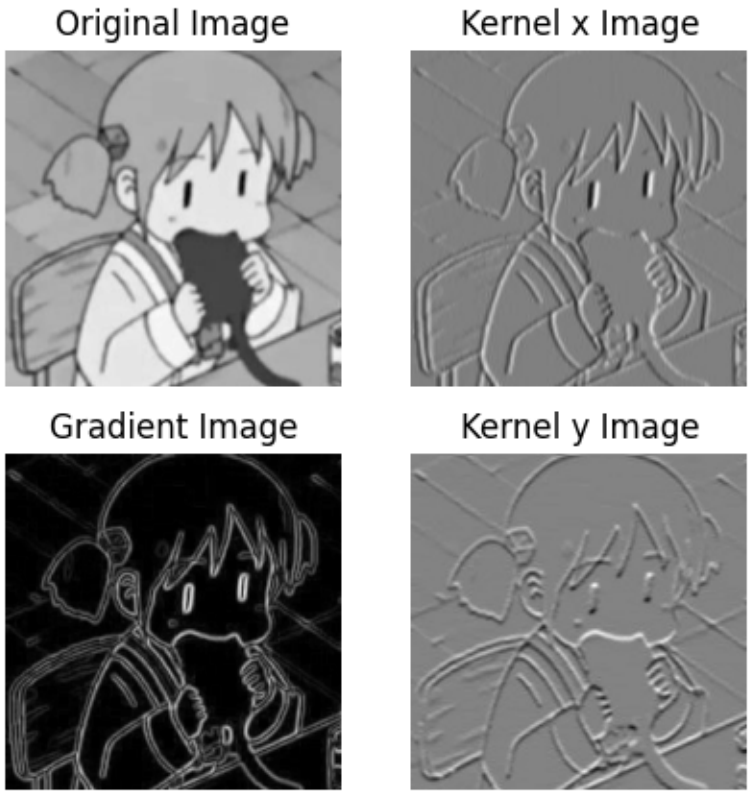
[1]:
```
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
```

1.1 Gradient

[126]:
```
img = cv.imread('test_cropped.jpg', cv.IMREAD_GRAYSCALE)

kernelx = np.array([
        [-1, 1],
        [-1, 1]
])
kernely = np.array([
        [1, 1],
        [-1, -1]
])


gx = cv.filter2D(img, cv.CV_16S, kernelx)
gy = cv.filter2D(img, cv.CV_16S, kernely)
gradient_img = np.sqrt(gx**2 + gy**2)

fig, ax = plt.subplots(2,2, figsize=(5,5))
ax[0,0].imshow(img, cmap='gray')
ax[0,0].set_axis_off()
ax[0,0].set_title('Original Image')

ax[0,1].imshow(gx, cmap='gray')
ax[0,1].set_axis_off()
ax[0,1].set_title('Kernel x Image')

ax[1,0].imshow(gradient_img, cmap='gray')
ax[1,0].set_axis_off()
ax[1,0].set_title('Gradient Image')
```

```
ax[1,1].imshow(gy, cmap='gray')
ax[1,1].set_axis_off()
ax[1,1].set_title('Kernel y Image')
```

[126]: `Text(0.5, 1.0, 'Kernel y Image')`

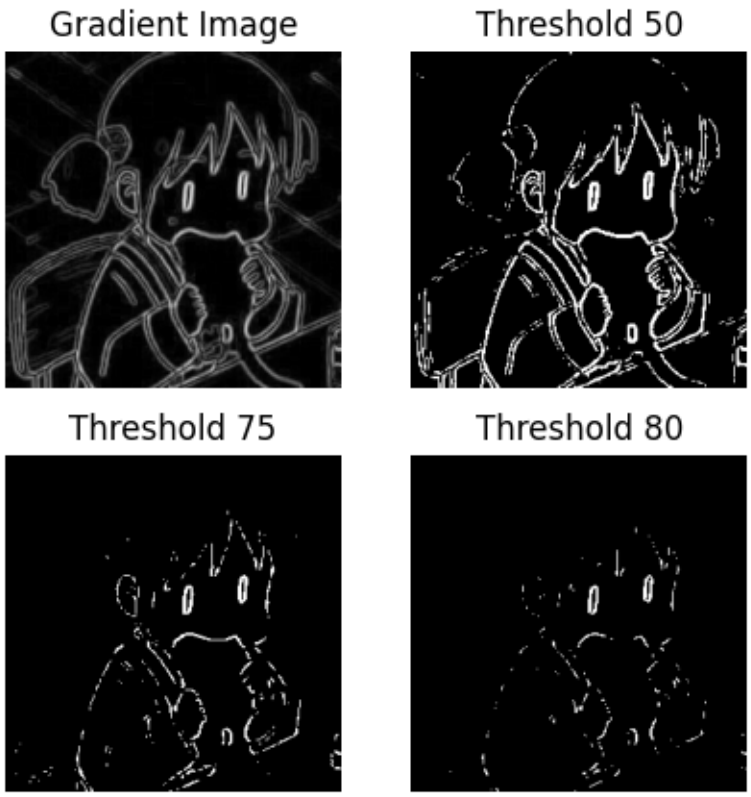


1.2 Gradient dengan Threshold

[127]:
```
_, th50 = cv.threshold(gradient_img, 50, 255, cv.THRESH_BINARY)
_, th70 = cv.threshold(gradient_img, 70, 255, cv.THRESH_BINARY)
_, th80 = cv.threshold(gradient_img, 80, 255, cv.THRESH_BINARY)

fig, ax = plt.subplots(2,2, figsize=(5,5))
ax[0,0].imshow(gradient_img, cmap='gray')
ax[0,0].set_axis_off()
ax[0,0].set_title('Gradient Image')

ax[0,1].imshow(th50, cmap='gray')
ax[0,1].set_axis_off()
ax[0,1].set_title('Threshold 50')
```

```
ax[1,0].imshow(th70, cmap='gray')
ax[1,0].set_axis_off()
ax[1,0].set_title('Threshold 75')

ax[1,1].imshow(th80, cmap='gray')
ax[1,1].set_axis_off()
ax[1,1].set_title('Threshold 80')
```

[127]: `Text(0.5, 1.0, 'Threshold 80')`



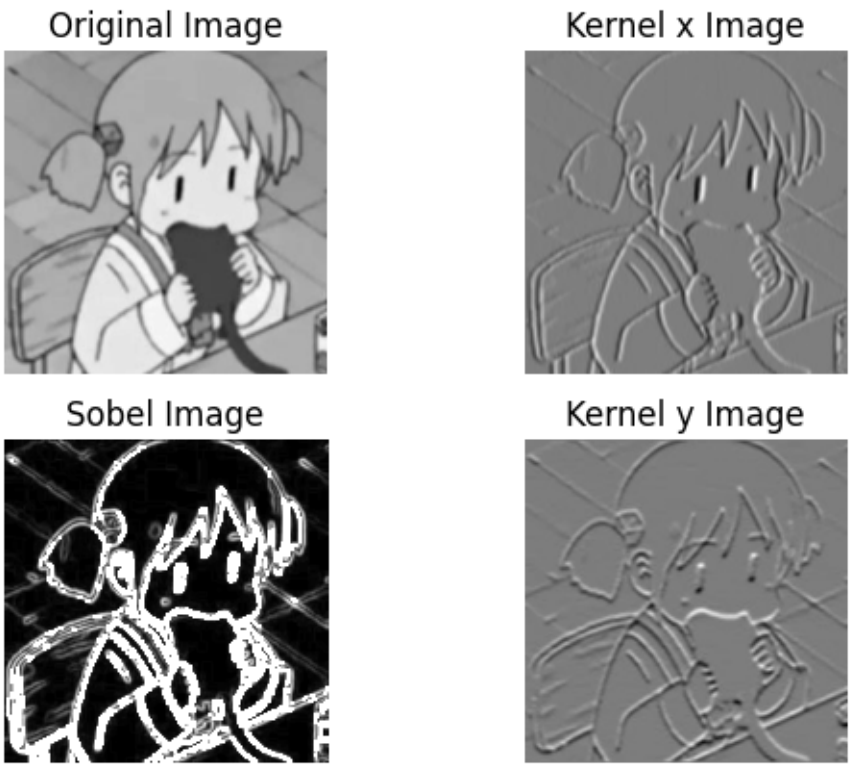
1.3 Sobel

[128]:
```
img = cv.imread('test_cropped.jpg', cv.IMREAD_GRAYSCALE)

kernelx = np.array([
        [-1, 0, 1],
        [-2, 0, 2],
        [-1, 0, 1]
])
kernely = np.array([
```

```
        [1, 2, 1],
        [0, 0, 0],
        [-1, -2, -1]
])

gx = cv.filter2D(img, cv.CV_16S, kernelx)
gy = cv.filter2D(img, cv.CV_16S, kernely)
sobel_img = np.sqrt(gx**2 + gy**2)

fig, ax = plt.subplots(2,2)
ax[0,0].imshow(img, cmap='gray')
ax[0,0].set_axis_off()
ax[0,0].set_title('Original Image')

ax[0,1].imshow(gx, cmap='gray')
ax[0,1].set_axis_off()
ax[0,1].set_title('Kernel x Image')

ax[1,0].imshow(sobel_img, cmap='gray')
ax[1,0].set_axis_off()
ax[1,0].set_title('Sobel Image')

ax[1,1].imshow(gy, cmap='gray')
ax[1,1].set_axis_off()
ax[1,1].set_title('Kernel y Image')
```

```
C:\Users\LAB SISTER\AppData\Local\Temp\ipykernel_8520\3605880758.py:16:
RuntimeWarning: invalid value encountered in sqrt
  sobel_img = np.sqrt(gx**2 + gy**2)
```

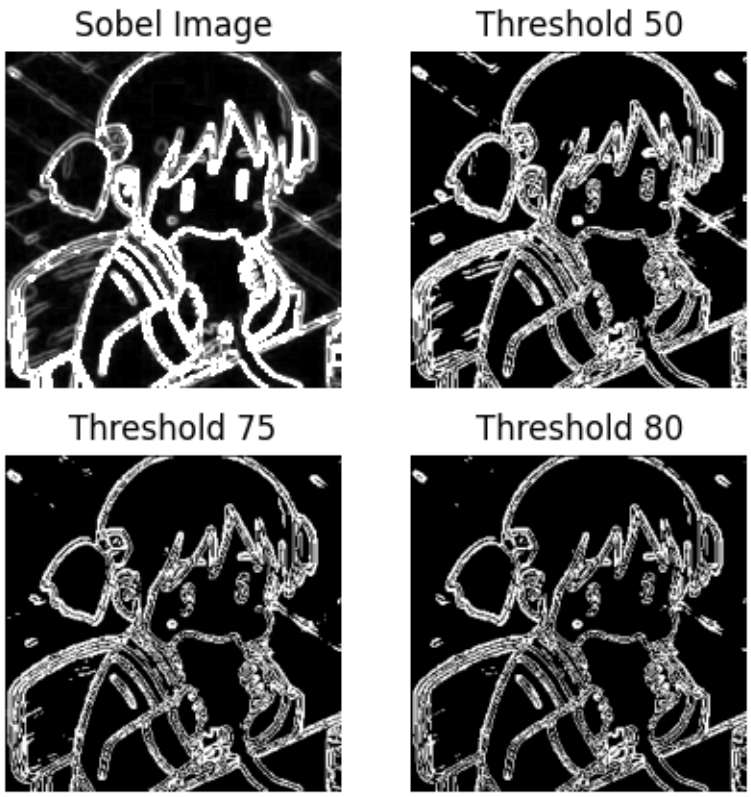
[128]: `Text(0.5, 1.0, 'Kernel y Image')`



1.4 Sobel dengan Threshold

[129]:
```
_, th50 = cv.threshold(sobel_img, 50, 255, cv.THRESH_BINARY)
_, th70 = cv.threshold(sobel_img, 70, 255, cv.THRESH_BINARY)
_, th80 = cv.threshold(sobel_img, 80, 255, cv.THRESH_BINARY)

fig, ax = plt.subplots(2,2, figsize=(5,5))
ax[0,0].imshow(sobel_img, cmap='gray')
ax[0,0].set_axis_off()
ax[0,0].set_title('Sobel Image')

ax[0,1].imshow(th50, cmap='gray')
ax[0,1].set_axis_off()
ax[0,1].set_title('Threshold 50')

ax[1,0].imshow(th70, cmap='gray')
ax[1,0].set_axis_off()
ax[1,0].set_title('Threshold 75')

ax[1,1].imshow(th80, cmap='gray')
ax[1,1].set_axis_off()
ax[1,1].set_title('Threshold 80')
```

[129]: Text(0.5, 1.0, 'Threshold 80')

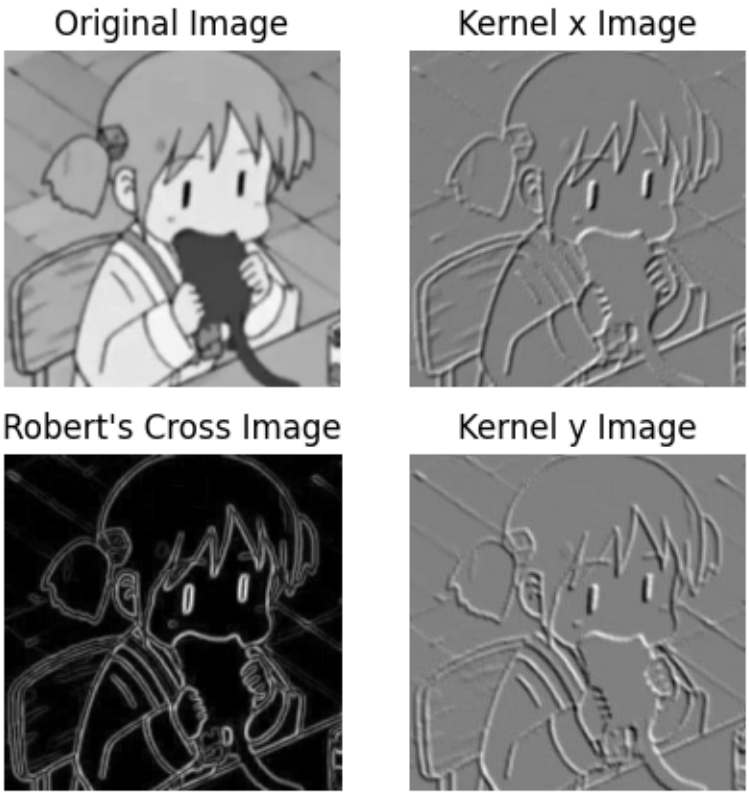


1.5 Robert's Cross

[130]:
```
img = cv.imread('test_cropped.jpg', cv.IMREAD_GRAYSCALE)

kernelx = np.array([
        [1, 0],
        [0, -1]
])
kernely = np.array([
        [0, 1],
        [-1, 0]
])

gx = cv.filter2D(img, cv.CV_16S, kernelx)
gy = cv.filter2D(img, cv.CV_16S, kernely)
robert_img = np.sqrt(gx**2 + gy**2)


fig, ax = plt.subplots(2,2, figsize=(5,5))
```

```
ax[0,0].imshow(img, cmap='gray')
ax[0,0].set_axis_off()
ax[0,0].set_title('Original Image')

ax[0,1].imshow(gx, cmap='gray')
ax[0,1].set_axis_off()
ax[0,1].set_title('Kernel x Image')

ax[1,0].imshow(robert_img, cmap='gray')
ax[1,0].set_axis_off()
ax[1,0].set_title('Robert\'s Cross Image')

ax[1,1].imshow(gy, cmap='gray')
ax[1,1].set_axis_off()
ax[1,1].set_title('Kernel y Image')
```

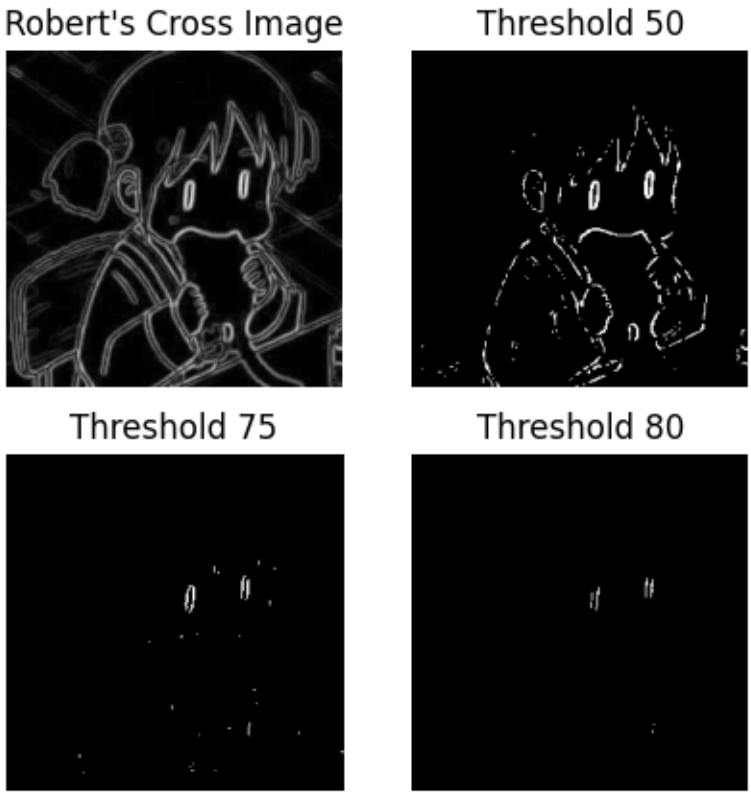
[130]: `Text(0.5, 1.0, 'Kernel y Image')`



1.6 Robert's Cross dengan Threshold

[ ]:
```
_, th50 = cv.threshold(robert_img, 50, 255, cv.THRESH_BINARY)
_, th70 = cv.threshold(robert_img, 70, 255, cv.THRESH_BINARY)
_, th80 = cv.threshold(robert_img, 80, 255, cv.THRESH_BINARY)

fig, ax = plt.subplots(2,2, figsize=(5,5))
ax[0,0].imshow(robert_img, cmap='gray')
ax[0,0].set_axis_off()
ax[0,0].set_title('Robert\'s Cross Image')

ax[0,1].imshow(th50, cmap='gray')
ax[0,1].set_axis_off()
ax[0,1].set_title('Threshold 50')

ax[1,0].imshow(th70, cmap='gray')
ax[1,0].set_axis_off()
ax[1,0].set_title('Threshold 75')

ax[1,1].imshow(th80, cmap='gray')
ax[1,1].set_axis_off()
ax[1,1].set_title('Threshold 80')
```

[ ]:
```
Text(0.5, 1.0, 'Threshold 80')
```

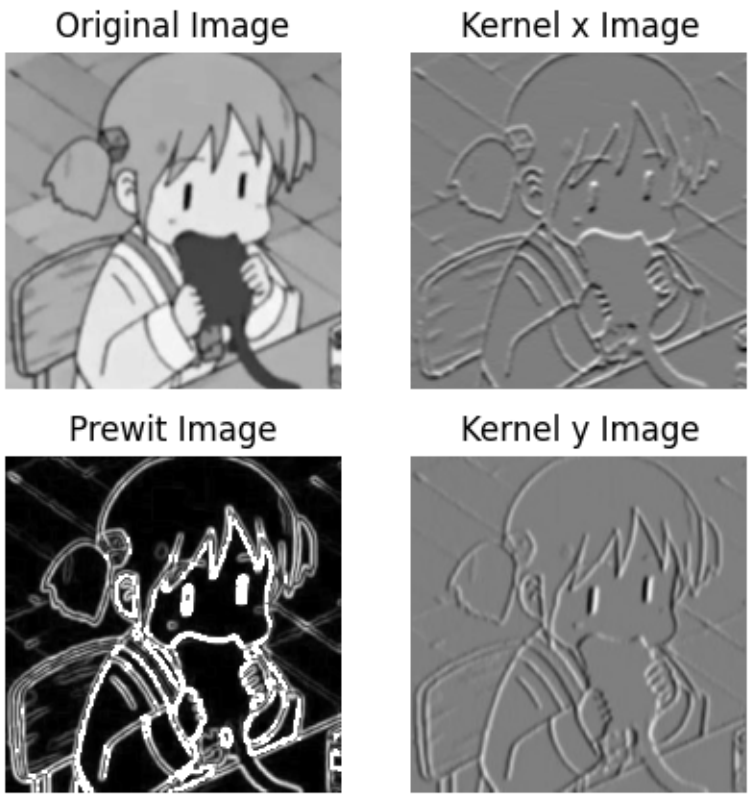


1.7 Prewit

[132]:
```
img = cv.imread('test_cropped.jpg', cv.IMREAD_GRAYSCALE)

kernelx = np.array([
        [1, 1, 1],
        [0, 0, 0],
        [-1, -1, -1]
])
kernely = np.array([
        [-1, 0, 1],
        [-1, 0, 1],
        [-1, 0, 1]
])

gx = cv.filter2D(img, cv.CV_16S, kernelx)
gy = cv.filter2D(img, cv.CV_16S, kernely)
prewit_img = np.sqrt(gx**2 + gy**2)


fig, ax = plt.subplots(2,2, figsize=(5,5))
ax[0,0].imshow(img, cmap='gray')
ax[0,0].set_axis_off()
ax[0,0].set_title('Original Image')

ax[0,1].imshow(gx, cmap='gray')
ax[0,1].set_axis_off()
ax[0,1].set_title('Kernel x Image')

ax[1,0].imshow(prewit_img, cmap='gray')
ax[1,0].set_axis_off()
ax[1,0].set_title('Prewit Image')

ax[1,1].imshow(gy, cmap='gray')
ax[1,1].set_axis_off()
ax[1,1].set_title('Kernel y Image')
```

```
C:\Users\LAB SISTER\AppData\Local\Temp\ipykernel_8520\1484552036.py:16:
RuntimeWarning: invalid value encountered in sqrt
  prewit_img = np.sqrt(gx**2 + gy**2)
```

[132]:
```
Text(0.5, 1.0, 'Kernel y Image')
```

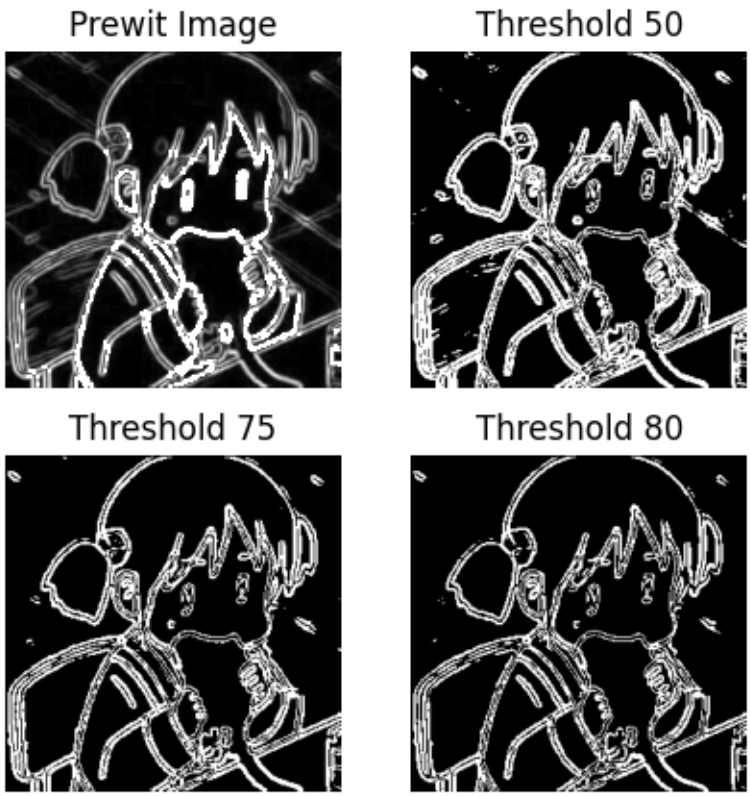


1.8 Prewit dengan Threshold

[28]:
```
_, th50 = cv.threshold(prewit_img, 50, 255, cv.THRESH_BINARY)
_, th70 = cv.threshold(prewit_img, 70, 255, cv.THRESH_BINARY)
_, th80 = cv.threshold(prewit_img, 80, 255, cv.THRESH_BINARY)

fig, ax = plt.subplots(2,2, figsize=(5,5))
ax[0,0].imshow(prewit_img, cmap='gray')
ax[0,0].set_axis_off()
ax[0,0].set_title('Prewit Image')

ax[0,1].imshow(th50, cmap='gray')
ax[0,1].set_axis_off()
ax[0,1].set_title('Threshold 50')

ax[1,0].imshow(th70, cmap='gray')
ax[1,0].set_axis_off()
ax[1,0].set_title('Threshold 75')

ax[1,1].imshow(th80, cmap='gray')
ax[1,1].set_axis_off()
```

```
ax[1,1].set_title('Threshold 80')
```

[28]: Text(0.5, 1.0, 'Threshold 80')

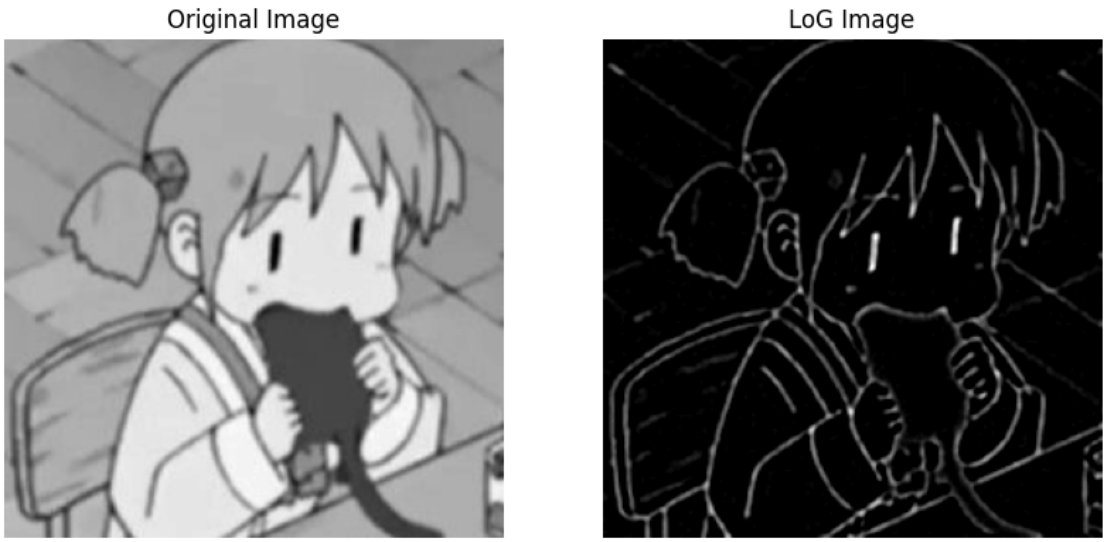


1.9 Laplacian of Gaussian

[3]:
```
img = cv.imread('test_cropped.jpg', cv.IMREAD_GRAYSCALE)

def create_LOG_kernel(size, sigma):
    R = size // 2
    y, x = np.ogrid[-R:R+1, -R:R+1]

    kernel = (1/sigma**2) * ((x**2 + y**2) / (sigma**2) - 2) * np.exp(-(x**2 +␣
 ↪y**2) / (2 * sigma**2))
    return kernel - np.mean(kernel)

def apply_LOG_filter(img, kernel_size=5, sigma=1.0):
    kernel = create_LOG_kernel(kernel_size, sigma)

    kernel = kernel / np.sum(np.abs(kernel))
```

```
    filtered_img = cv.filter2D(img, -1, kernel)
    return filtered_img


result = apply_LOG_filter(img, kernel_size=17, sigma=1.2)

fig, ax = plt.subplots(1,2, figsize=(10,10))

ax[0].imshow(img, cmap='gray')
ax[0].set_axis_off()
ax[0].set_title('Original Image')

ax[1].imshow(result, cmap='gray')
ax[1].set_axis_off()
ax[1].set_title('LoG Image')
```

[3]: Text(0.5, 1.0, 'LoG Image')



1.10 Hasil Akhir

Dari beberapa percobaan menggunakan berbagai macam algoritma deteksi tepi, dengan menggunakan gambar yang ada, didapat dari hasil yang ditunjukkan oleh algoritma **Laplacian of Gaussian** merupakan algoritma deteksi tepi yang cocok digunakan pada gambar tersebut. Dapat dilihat, hasil dari gelap terang tepian Laplacian yang dihasilkan memiliki tingkat presisi yang bagus diimbangi dengan Gaussian agar mengurangi noise yang ada pada hasil, sehingga menghasilkan hasil yang sesuai dibandingkan dengan algoritma yang lain yang menghasilkan deteksi tepi yang kurang sesuai.