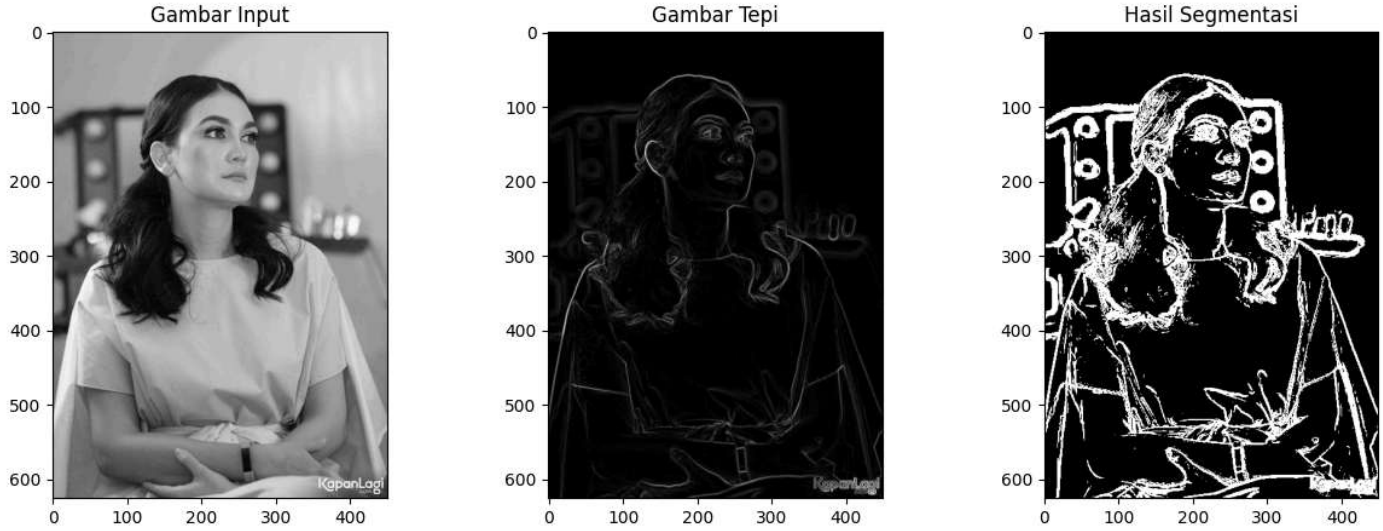
```
!pip install imageio
!pip install numpy
!pip install matplotlib
!pip install scipy
```

```
Requirement already satisfied: imageio in /usr/local/lib/python3.11/dist-packages (2.37.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from imageio) (1.26.4)
Requirement already satisfied: pillow>=8.3.2 in /usr/local/lib/python3.11/dist-packages (from imageio) (11.1.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (1.26.4)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (1.13.1)
Requirement already satisfied: numpy<2.3,>=1.22.4 in /usr/local/lib/python3.11/dist-packages (from scipy) (1.26.4)
```

```
import imageio
import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage

# Membaca gambar input
# Changed 'as_gray=True' to 'mode='F'' for backward compatibility
gambar_input = imageio.imread('SAMPEL.jpg', mode='F')

# Deteksi tepi menggunakan metode Sobel
sobel_x = ndimage.sobel(gambar_input, axis=0)
sobel_y = ndimage.sobel(gambar_input, axis=1)
gambar_tepi = np.hypot(sobel_x, sobel_y)

# Mengaplikasikan thresholding
threshold = 50
gambar_segmentasi = gambar_tepi > threshold

# Menampilkan hasil
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
axes[0].imshow(gambar_input, cmap='gray')
axes[0].set_title('Gambar Input')
axes[1].imshow(gambar_tepi, cmap='gray')
axes[1].set_title('Gambar Tepi')
axes[2].imshow(gambar_segmentasi, cmap='gray')
axes[2].set_title('Hasil Segmentasi')
plt.show()
```

```
<ipython-input-3-8912f06961d2>:8: DeprecationWarning: Starting with ImageIO v3 the behavior of this function will switch to that of ii
  gambar_input = imageio.imread('SAMPEL.jpg', mode='F')
```



1. Gambar Input : Menampilkan citra asli dalam skala keabuan (grayscale). Citra ini merupakan input awal untuk proses segmentasi.
2. Gambar Tepi : Menampilkan hasil deteksi tepi menggunakan operator Sobel. Operator Sobel menghitung gradien intensitas piksel pada arah horizontal (sobel_x) dan vertikal (sobel_y). Gambar_tepi merepresentasikan magnitude gradien, menunjukkan seberapa kuat perubahan intensitas pada setiap titik di citra. Daerah dengan tepi akan memiliki nilai magnitude gradien yang tinggi (terlihat lebih terang pada gambar), sedangkan daerah yang homogen akan memiliki nilai rendah (terlihat lebih gelap).
3. Hasil Segmentasi: Menampilkan hasil segmentasi setelah menerapkan thresholding pada gambar_tepi. Thresholding membagi piksel menjadi dua kelas: objek dan latar belakang. Piksel dengan magnitude gradien di atas threshold (50 dalam kasus ini) diklasifikasikan sebagai tepi objek (bernilai True, ditampilkan putih), sedangkan piksel di bawah threshold diklasifikasikan sebagai latar belakang (bernilai False, ditampilkan hitam). Hasilnya adalah citra biner yang menunjukkan segmentasi objek dari latar belakang.

Analisis Lebih Lanjut :

- Kualitas Segmentasi : Kualitas segmentasi bergantung pada pemilihan threshold yang tepat. Nilai threshold yang terlalu rendah dapat menghasilkan banyak noise, sedangkan nilai yang terlalu tinggi dapat menghilangkan detail tepi objek.
- Kompleksitas Objek : Metode ini efektif untuk objek dengan tepi yang jelas. Objek dengan tepi yang kabur atau tekstur yang kompleks mungkin memerlukan metode segmentasi yang lebih canggih.