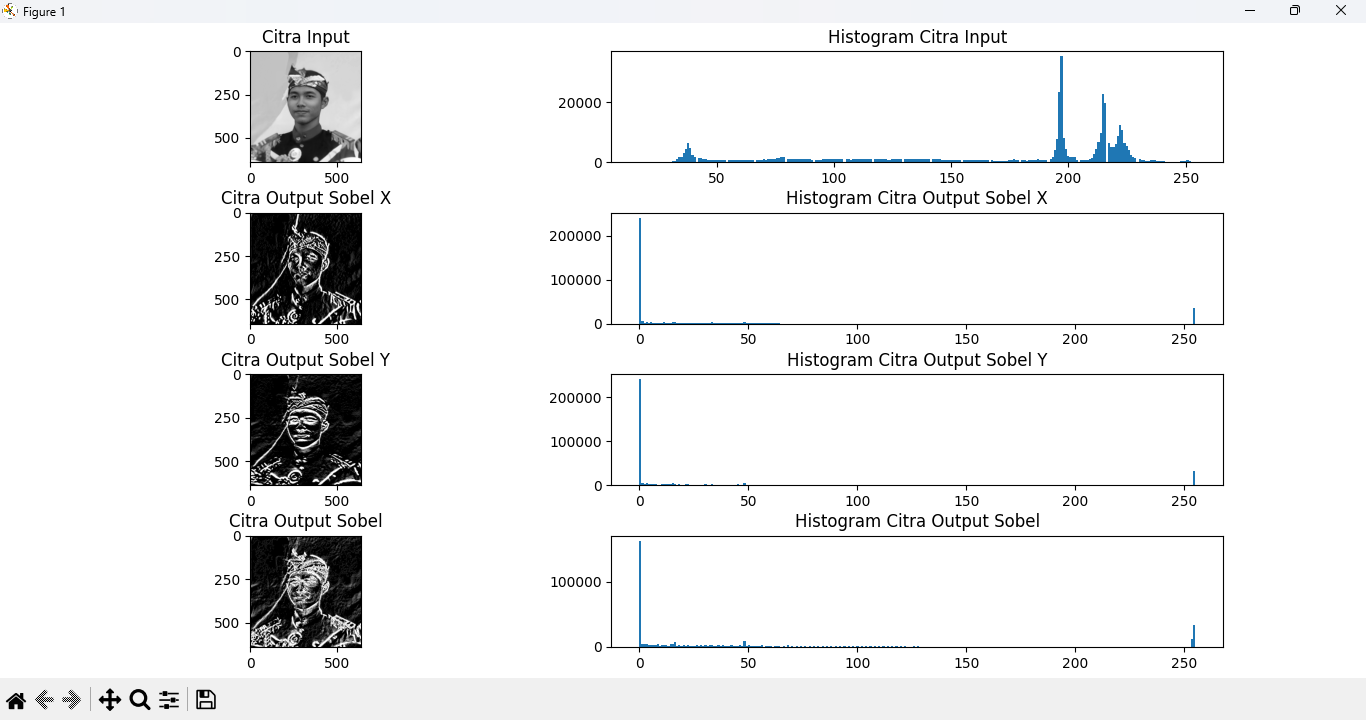
Nama : Rizky Mahesa Ramadhan

NIM : 1207070112

Mata Kuliah : Praktikum Pengolahan Citra Digital – TKK

PRAKTIKUM 9

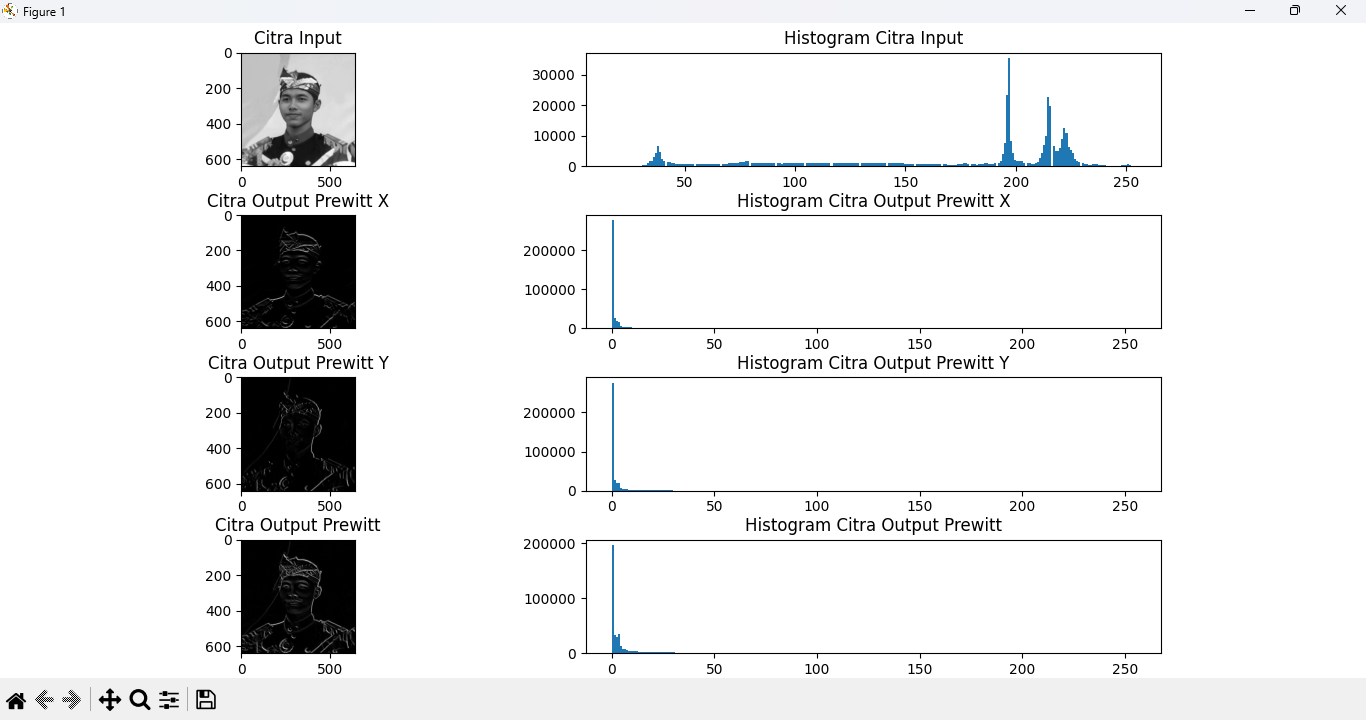
1. **Praktikum 9.1 – Sobel**



Program ini menggunakan Sobel untuk mendeteksi tepi pada gambar. Pertama, gambar input dibaca menggunakan fungsi **cv.imread()** dengan mode **cv.IMREAD\_GRAYSCALE** untuk mengubahnya menjadi citra grayscale. Kemudian, Sobel diterapkan menggunakan fungsi **cv.Sobel()**. Dalam kasus ini, Sobel diterapkan secara terpisah untuk menghitung gradien dalam arah horizontal (**img\_sobelx**) dan arah vertikal (**img\_sobely**). Selanjutnya, kedua citra hasil gradien tersebut digabungkan dengan menjumlahkan intensitas pikselnya untuk membentuk citra hasil deteksi tepi (**img\_sobel**). Program ini juga menampilkan citra input, histogram citra input, citra hasil deteksi tepi dalam arah X (**img\_sobelx**), histogram citra hasil deteksi tepi dalam arah X, citra hasil deteksi tepi dalam arah Y (**img\_sobely**), histogram citra hasil deteksi tepi dalam arah Y, citra hasil deteksi tepi (**img\_sobel**), dan histogram citra hasil deteksi tepi. Histogram digunakan untuk memvisualisasikan distribusi intensitas piksel dalam citra.

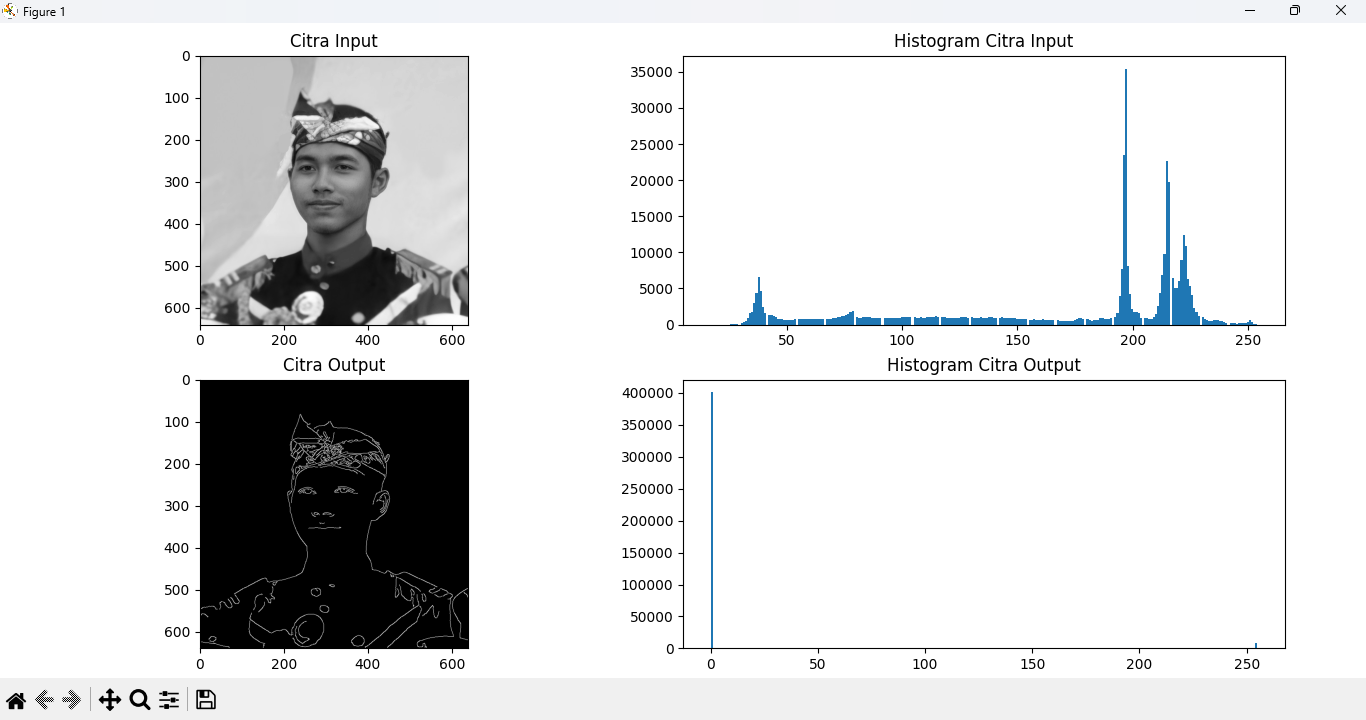
Hasilnya adalah citra yang menunjukkan lokasi tepi dengan intensitas yang lebih tinggi. Untuk kelebihannya, Sobel adalah metode deteksi tepi yang sederhana dan mudah diimplementasikan. Metode ini cenderung menghasilkan tepi yang cukup tajam. Selain itu, kekurangan Sobel tidak mempertimbangkan arah tepi tegak lurus terhadap tepi yang sebenarnya.

1. **Praktikum 9.2 – Prewitt**



Program ini menggunakan Prewitt untuk mendeteksi tepi pada gambar. Pada awalnya, gambar input dibaca dan diubah menjadi citra grayscale. Prewitt diterapkan pada citra grayscale menggunakan fungsi **cv.filter2D()**. Dalam hal ini, kernel Prewitt secara terpisah didefinisikan untuk menghitung gradien dalam arah horizontal (**kernelx**) dan arah vertikal (**kernely**). Hasil deteksi tepi dalam kedua arah tersebut kemudian digabungkan dengan menjumlahkan intensitas pikselnya untuk membentuk citra hasil deteksi tepi (**img\_prewitt**). Program ini juga menampilkan citra input, histogram citra input, citra hasil deteksi tepi dalam arah X (**img\_prewittx**), histogram citra hasil deteksi tepi dalam arah X, citra hasil deteksi tepi dalam arah Y (**img\_prewitty**), histogram citra hasil deteksi tepi dalam arah Y, citra hasil deteksi tepi (**img\_prewitt**), dan histogram citra hasil deteksi tepi. Kelebihan dan kekurangan metode Prewitt serupa dengan metode Sobel.

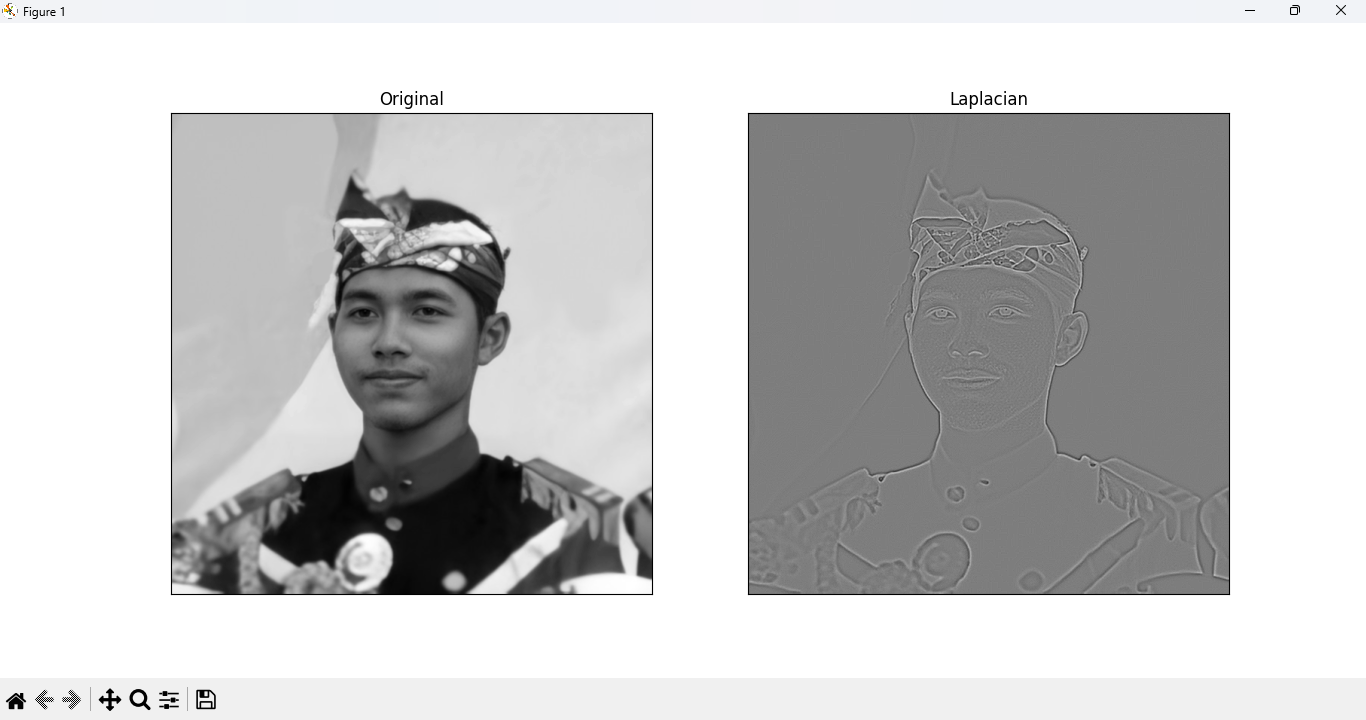
1. **Praktikum 9.3 – Canny**



Program ini menggunakan metode Canny untuk mendeteksi tepi pada gambar. Gambar input dibaca dan diubah menjadi citra grayscale. Metode Canny diterapkan pada citra grayscale menggunakan fungsi **cv.Canny()**. Dalam kasus ini, parameter threshold lower dan threshold upper diatur menjadi 100 dan 200. Hasil deteksi tepi disimpan dalam variabel **img\_canny**. Program ini juga menampilkan citra input, histogram citra input, citra hasil deteksi tepi (**img\_canny**), dan histogram citra hasil deteksi tepi.

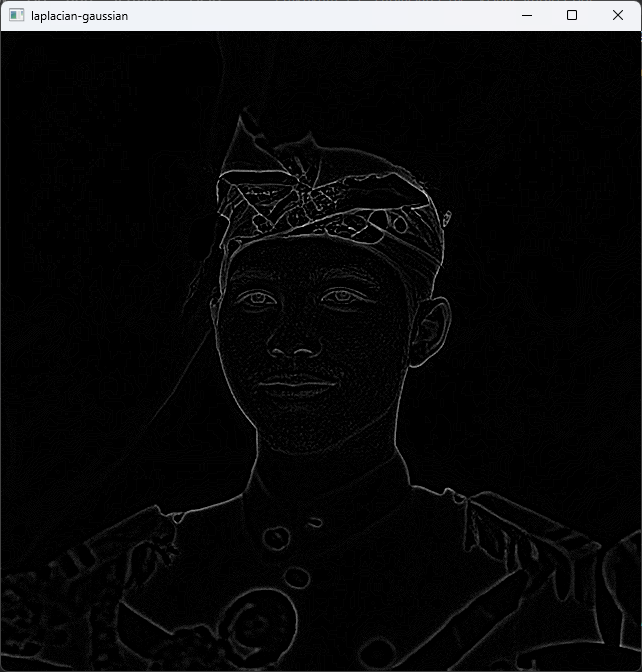
Hasilnya adalah citra yang menampilkan tepi dengan intensitas tinggi. Kelebihan Metode Canny memberikan deteksi tepi yang akurat dan tajam. Metode ini juga dapat menghilangkan noise dan menghubungkan tepi yang terputus. Kekurangannya, Metode Canny memerlukan lebih banyak komputasi dan implementasi yang lebih kompleks.

1. **Praktikum 9.4 – Laplacian (part 1)**



Program ini menggunakan Laplacian untuk mendeteksi tepi pada gambar. Gambar input dibaca dan dikonversi menjadi citra grayscale. Noise pada citra grayscale dihilangkan menggunakan filter Gaussian dengan ukuran kernel 3x3 dengan filter Gaussian (**cv2.GaussianBlur()**). Laplacian diterapkan pada citra hasil penghilangan noise menggunakan fungsi **cv2.Laplacian()** untuk mendeteksi perubahan intensitas piksel yang tajam. Hasil deteksi tepi (laplacian) ditampilkan menggunakan matplotlib. Kelebihan metode Laplacian memberikan deteksi tepi yang tajam, terutama pada tepi yang memiliki perubahan intensitas yang drastis. Kekurangannya, metode ini dapat sensitif terhadap noise pada gambar.

1. **Praktikum 9.5 – Laplacian (part 2)**



Program ini juga menggunakan Laplacian untuk mendeteksi tepi pada gambar. Gambar input dibaca dan diubah menjadi citra grayscale. Dilakukan penghilangan noise pada citra grayscale dengan filter Gaussian. Laplacian diterapkan pada citra hasil penghilangan noise menggunakan fungsi **cv2.Laplacian()**. Hasil deteksi tepi disimpan dalam variabel **laplacian**. Hasil deteksi tepi dinormalisasi dengan membaginya dengan nilai maksimum dari tepi (**laplacian/laplacian.max()**). Citra hasil deteksi tepi ditampilkan menggunakan OpenCV. Program ini menampilkan citra hasil deteksi tepi yang dihasilkan oleh Laplacian setelah penghilangan noise menggunakan filter Gaussian untuk mendeteksi perubahan intensitas piksel yang tajam. Kelebihan dan kekurangan metode Laplacian pada Program Laplacian 2 serupa dengan Program Laplacian 1.

1. **KESIMPULAN**

* Jika mencari metode deteksi tepi yang akurat dan tajam dengan menghilangkan noise dan menghubungkan tepi yang terputus, direkomendasikan menggunakan metode Canny.
* Jika mencari metode yang sederhana dan mudah diimplementasikan, metode Sobel atau Prewitt dapat menjadi pilihan yang baik.
* Metode Laplacian (Program Laplacian 1 dan Laplacian 2) juga memberikan hasil yang tajam, tetapi harus memperhatikan sensitivitas terhadap noise.
* Pilihan metode terbaik bergantung pada kebutuhan spesifik dalam deteksi tepi, tingkat kompleksitas yang dapat ditoleransi, dan jenis gambar yang akan diproses.