

LAPORAN TEORI

PENGOLAHAN CITRA DIGITAL



NAMA : Nyimas Adinda Paradiza

NIM : 202331041

KELAS : PCD B

DOSEN : Ir. Darma Rusjdi, M.Kom

NO.PC : -

ASISTEN : 1. Davina Najwa Ermawan
2. Fakhrul Fauzi Nugraha Tarigan
3. Viana Salsabila Fairuz Syahla
4. Muhammad Hanief Febriansyah

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2025

1. Konsep Dasar Operasi Konvolusi dalam Pengolahan Citra Digital

Operasi konvolusi adalah proses di mana sebuah kernel (matriks kecil) digeser di seluruh citra untuk memproses piksel dengan mempertimbangkan nilai-nilai tetangganya. Hasil konvolusi digunakan untuk berbagai keperluan, seperti deteksi tepi, pengaburan (blur), dan penajaman. Nilai piksel baru ditentukan dari hasil perkalian elemen kernel dengan piksel citra di sekitarnya, lalu dijumlahkan.

2. Metode Ketetanggaan dalam Pengolahan Citra

Metode ketetanggaan melibatkan analisis nilai piksel di sekitar piksel pusat dalam area lokal (misal 3×3). Metode ini penting untuk operasi seperti filtering, deteksi tepi, dan segmentasi, karena mempertimbangkan konteks lokal dalam pengambilan keputusan terhadap nilai piksel.

3. Prinsip Average Filter dan Median Filter serta Kelebihan-Kekurangannya

Average Filter menghitung rata-rata dari nilai piksel tetangga untuk menghaluskan noise, cocok untuk noise ringan tapi bisa mengaburkan tepi. Kelebihannya Sederhana, cepat sedangkan kekurangannya Mengaburkan detail.

Median Filter mengganti piksel pusat dengan nilai tengah dari tetangganya, sangat efektif untuk mengurangi salt and pepper noise dan lebih baik mempertahankan tepi. Kelebihannya Efektif untuk noise impuls dan kekurangannya Lebih lambat, kompleks.

4. Speckle Noise vs Gaussian Noise

Speckle noise adalah jenis noise granular yang muncul terutama dalam citra hasil pemindaian berbasis gelombang seperti ultrasound, radar, atau citra medis. Noise ini bersifat multiplikatif, artinya nilai noise tergantung pada intensitas piksel asli, dan menghasilkan tampilan kasar seperti bintik-bintik.

Gaussian noise, sebaliknya, bersifat additif dan distribusinya mengikuti kurva normal (Gaussian). Ia ditambahkan secara acak ke seluruh piksel tanpa tergantung pada nilai asli piksel tersebut.

Perbedaan utama :

- Speckle noise bergantung pada intensitas piksel (multiplikatif), sedangkan Gaussian noise tidak (additif).
- Speckle lebih umum pada citra medis dan radar, sedangkan Gaussian lebih sering ditemukan akibat sensor elektronik.
- Speckle lebih sulit dihilangkan karena mempengaruhi struktur tekstur citra.

5. Manfaat Operasi Konvolusi dalam Komputer Visi

Dalam bidang computer vision, operasi konvolusi sangat krusial untuk berbagai aplikasi, terutama dalam deteksi fitur, klasifikasi objek, dan segmentasi citra. Salah satu aplikasi utama adalah dalam Convolutional Neural Networks (CNNs), di mana lapisan konvolusi digunakan untuk secara otomatis mengekstraksi fitur penting dari citra input. Dengan menggunakan berbagai kernel, CNN dapat mengenali pola sederhana seperti garis dan tepi, hingga fitur kompleks seperti bentuk wajah atau objek tertentu.

LAPORAN PRAKTIKUM

PENGOLAHAN CITRA DIGITAL



INTELLIGENT COMPUTING

NAMA : Nyimas Adinda Paradiza

NIM : 202331041

KELAS : PCD B

DOSEN : Ir. Darma Rusjdi, M.Kom

NO.PC : -

ASISTEN : 1. Davina Najwa Ermawan

2. Fakhrol Fauzi Nugraha Tarigan

3. Viana Salsabila Fairuz Syahla

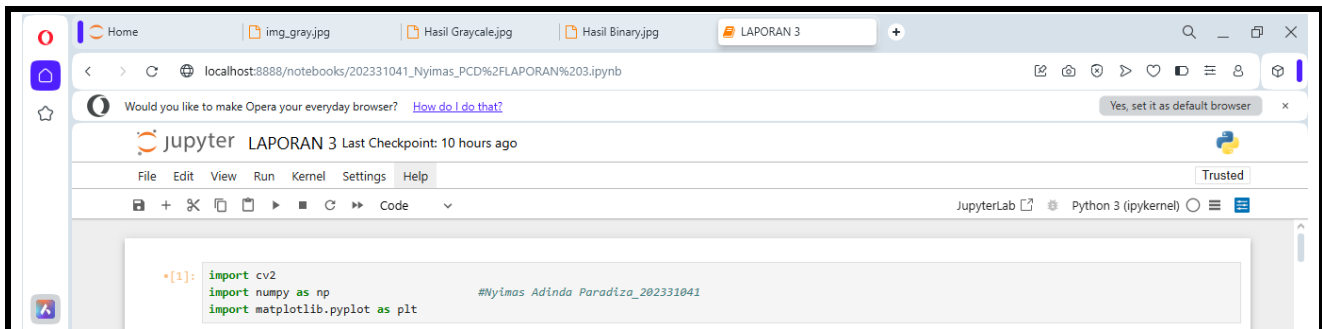
4. Muhammad Hanief Febriansyah

INSTITUT TEKNOLOGI PLN

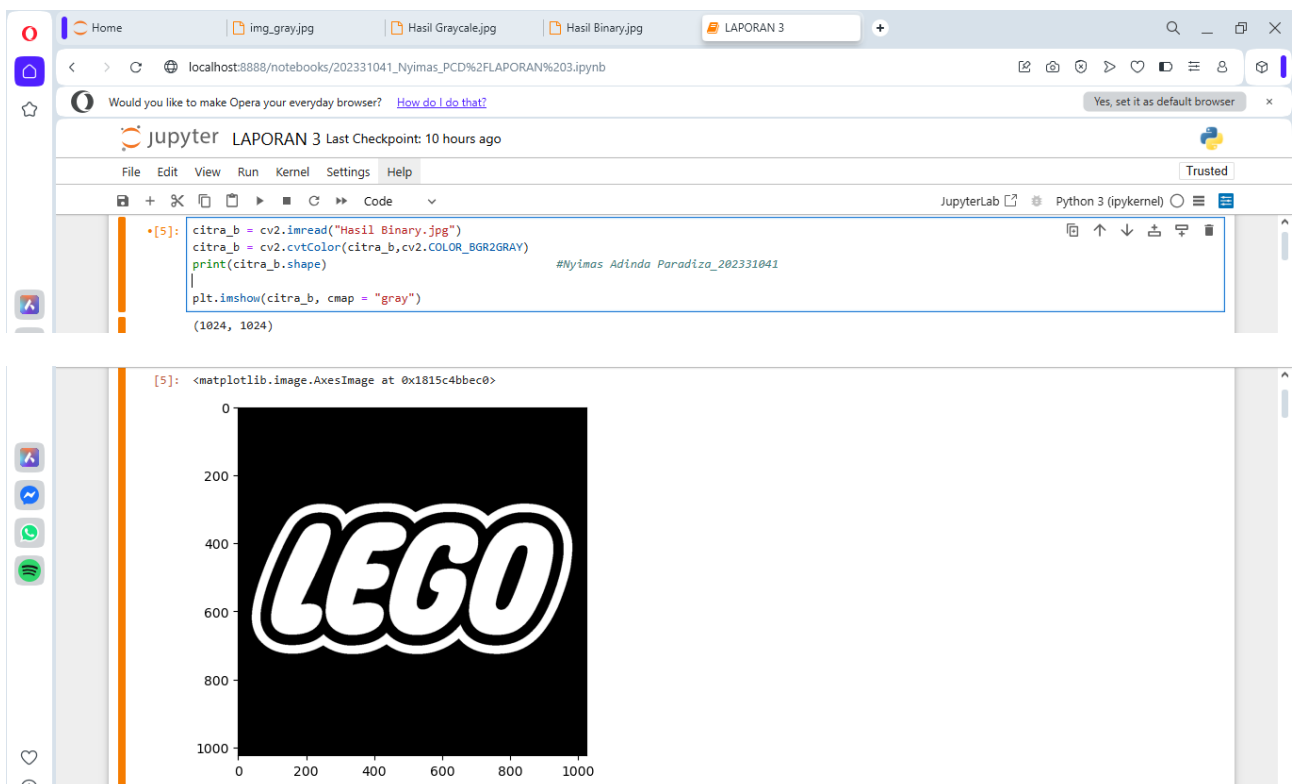
TEKNIK INFORMATIKA

2025

Laporan 3



1. import cv2
 - a. Ini adalah library OpenCV (Open Source Computer Vision Library).
 - b. Digunakan untuk proses pengolahan citra, seperti membaca gambar, mengubah warna, mengaburkan, mendeteksi tepi, dan lain-lain.
2. import numpy as np
 - a. numpy adalah library Python untuk komputasi numerik.
 - b. Sangat penting dalam pengolahan citra karena gambar disimpan sebagai array (matriks).
 - c. np digunakan untuk manipulasi matriks gambar, membuat filter, kernel, atau operasi matematika.
3. import matplotlib.pyplot as plt
 - a. Library ini digunakan untuk menampilkan gambar atau grafik secara visual.
 - b. plt adalah singkatan dari pyplot, modul dalam matplotlib.
 - c. Biasanya dipakai untuk menampilkan hasil pengolahan gambar.



4. citra_b = cv2.imread("Hasil Binary.jpg")

Digunakan untuk membaca gambar dari file. Fungsi cv2.imread() mengambil gambar dan menyimpannya sebagai array (matriks piksel) dalam format BGR (Blue, Green, Red).

5. `citra_b = cv2.cvtColor(citra_b, cv2.COLOR_BGR2GRAY)`

Digunakan untuk mengubah gambar berwarna menjadi grayscale (hitam-putih). Fungsi `cv2.cvtColor()` mengkonversi format warna gambar dari BGR ke Grayscale sehingga hanya menyimpan informasi intensitas cahaya.

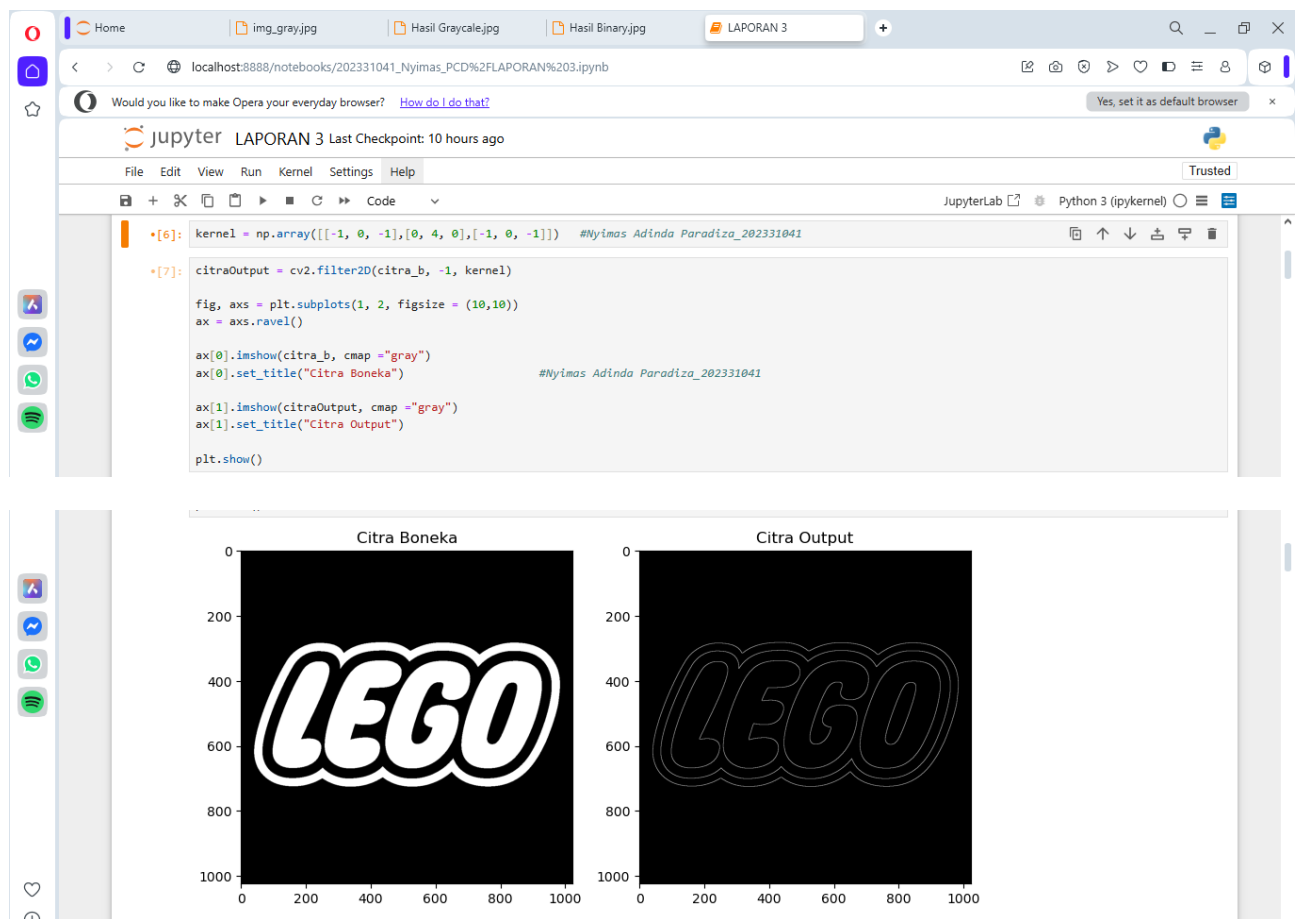
6. `print(citra_b.shape)`

Menampilkan dimensi gambar, yaitu jumlah baris dan kolom piksel. `shape` akan memberikan informasi ukuran array gambar (tinggi dan lebar citra), dan jika masih berwarna, juga jumlah saluran warna.

7. `plt.imshow(citra_b, cmap = "gray")`

Menampilkan gambar grayscale ke layar menggunakan matplotlib. Parameter `cmap="gray"` memastikan gambar ditampilkan dalam skala abu-abu agar sesuai dengan isi data citra grayscale.

8. Dan terakhir ada outputan gambar, gambar di atas.



9. `kernel = np.array([[-1, 0, -1],[0, 4, 0],[-1, 0, -1]])`

Membuat kernel (matriks filter) untuk deteksi tepi (edge detection). Kernel ini merupakan jenis Laplacian operator, yang berfungsi untuk mendeteksi perubahan intensitas piksel, yaitu tepi atau batas objek dalam gambar.

10. `citraOutput = cv2.filter2D(citra_b, -1, kernel)`

Melakukan konvolusi gambar dengan kernel yang sudah dibuat. Fungsi `filter2D` mengaplikasikan filter ke seluruh piksel gambar. Parameter `-1` artinya output akan memiliki kedalaman warna yang sama dengan input (`citra_b`).

11. `fig, axs = plt.subplots(1, 2, figsize=(10,10))`

Membuat kanvas dengan 1 baris dan 2 kolom untuk menampilkan dua gambar berdampingan. figsize menentukan ukuran tampilan dalam satuan inci.

12. `ax = axs.ravel()`

Mengubah susunan objek `axs` dari array 2D menjadi 1D, agar mudah diakses per indeks saat menampilkan gambar.

13. `ax[0].imshow(citra_b, cmap="gray")`

Menampilkan gambar asli (sebelum difilter) dalam warna abu-abu (grayscale).

14. `ax[0].set_title("Citra Boneka")`

Memberikan judul pada gambar pertama.

15. `ax[1].imshow(citraOutput, cmap="gray")`

Menampilkan gambar hasil filter (citra yang telah mengalami deteksi tepi) dalam warna grayscale.

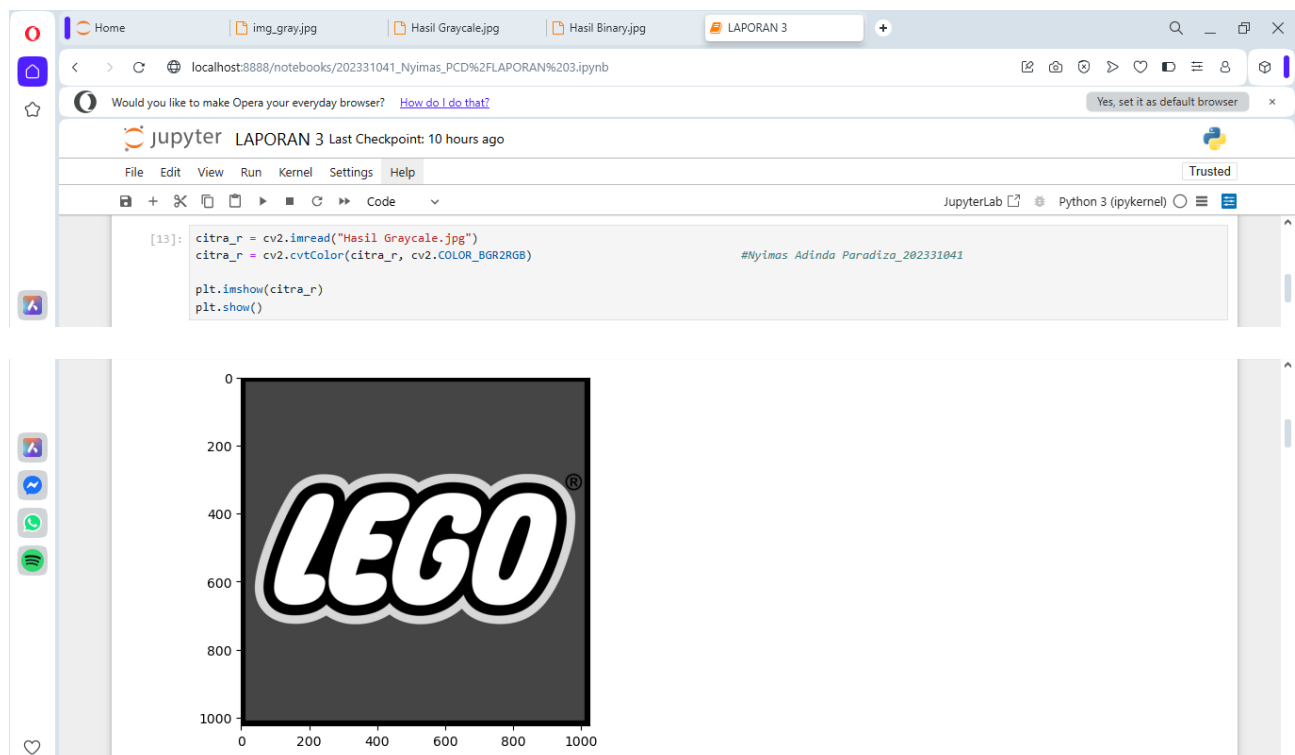
16. `ax[1].set_title("Citra Output")`

Memberi judul pada gambar hasil filter.

17. `plt.show()`

Menampilkan seluruh plot (gambar) dalam satu jendela.

18. Dan terakhir terdapat gambar outputannya.



19. `citra_r = cv2.imread("Hasil Graycale.jpg")`

Membaca gambar dari file bernama "Hasil Graycale.jpg". Saat dibaca dengan `cv2.imread`, gambar otomatis disimpan dalam format BGR (Blue, Green, Red).

20. `citra_r = cv2.cvtColor(citra_r, cv2.COLOR_BGR2RGB)`

Mengubah format warna gambar dari BGR ke RGB, agar warnanya tampil dengan benar saat ditampilkan menggunakan matplotlib. Hal ini penting karena OpenCV (`cv2`) dan matplotlib (`plt`) menggunakan urutan warna berbeda:

OpenCV → BGR

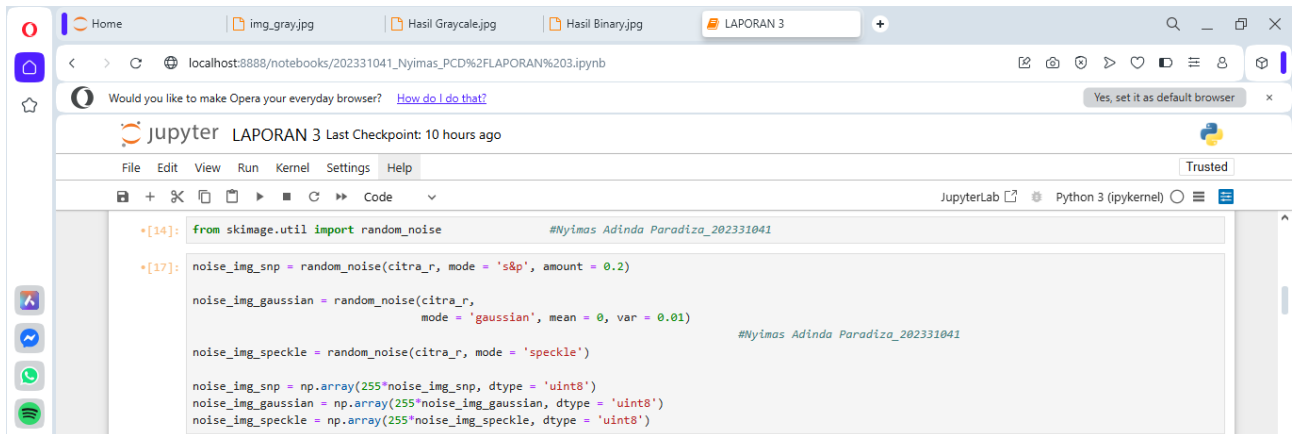
Matplotlib → RGB

21. `plt.imshow(citra_r)`

Menampilkan gambar dalam format RGB agar warna sesuai dengan aslinya.

22. `plt.show()`

Menampilkan gambar di jendela keluaran matplotlib.



```

+ [14]: from skimage.util import random_noise #Nyimas Adinda Paradiza_202331041
+ [17]: noise_img_snp = random_noise(citra_r, mode = 's&p', amount = 0.2)

noise_img_gaussian = random_noise(citra_r,
                                   mode = 'gaussian', mean = 0, var = 0.01) #Nyimas Adinda Paradiza_202331041

noise_img_speckle = random_noise(citra_r, mode = 'speckle')

noise_img_snp = np.array(255*noise_img_snp, dtype = 'uint8')
noise_img_gaussian = np.array(255*noise_img_gaussian, dtype = 'uint8')
noise_img_speckle = np.array(255*noise_img_speckle, dtype = 'uint8')

```

23. `from skimage.util import random_noise`

Mengimpor fungsi `random_noise` dari pustaka `skimage` (Scikit-image). Fungsi ini digunakan untuk menambahkan noise (derau/gangguan) ke dalam citra, untuk simulasi atau pengujian pemrosesan citra.

24. `noise_img_snp = random_noise(citra_r, mode='s&p', amount=0.2)`

Menambahkan noise Salt & Pepper ke citra (`citra_r`). Mode `'s&p'`: noise berbentuk titik hitam dan putih acak. Amount 0.2: artinya 20% piksel pada gambar akan diganggu (diubah menjadi hitam atau putih).

25. `noise_img_gaussian = random_noise(citra_r, mode='gaussian', mean=0, var=0.01)`

Menambahkan noise Gaussian ke gambar. Mode `'gaussian'`: noise mengikuti distribusi normal (berbentuk kabut). Mean 0 dan varian 0.01: menentukan tingkat keparahan gangguan (semakin besar varian, semakin kabur citranya).

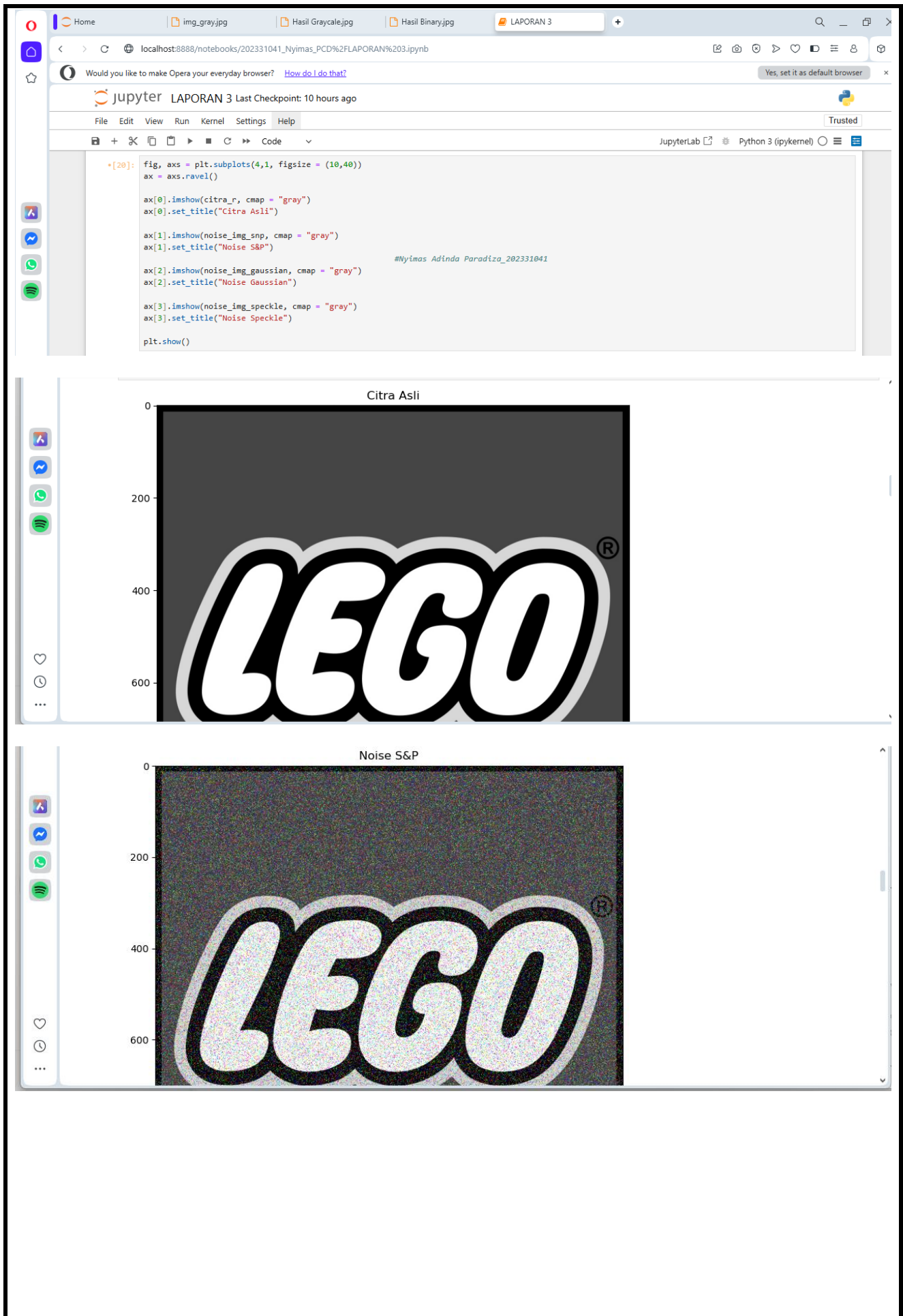
26. `noise_img_speckle = random_noise(citra_r, mode='speckle')`

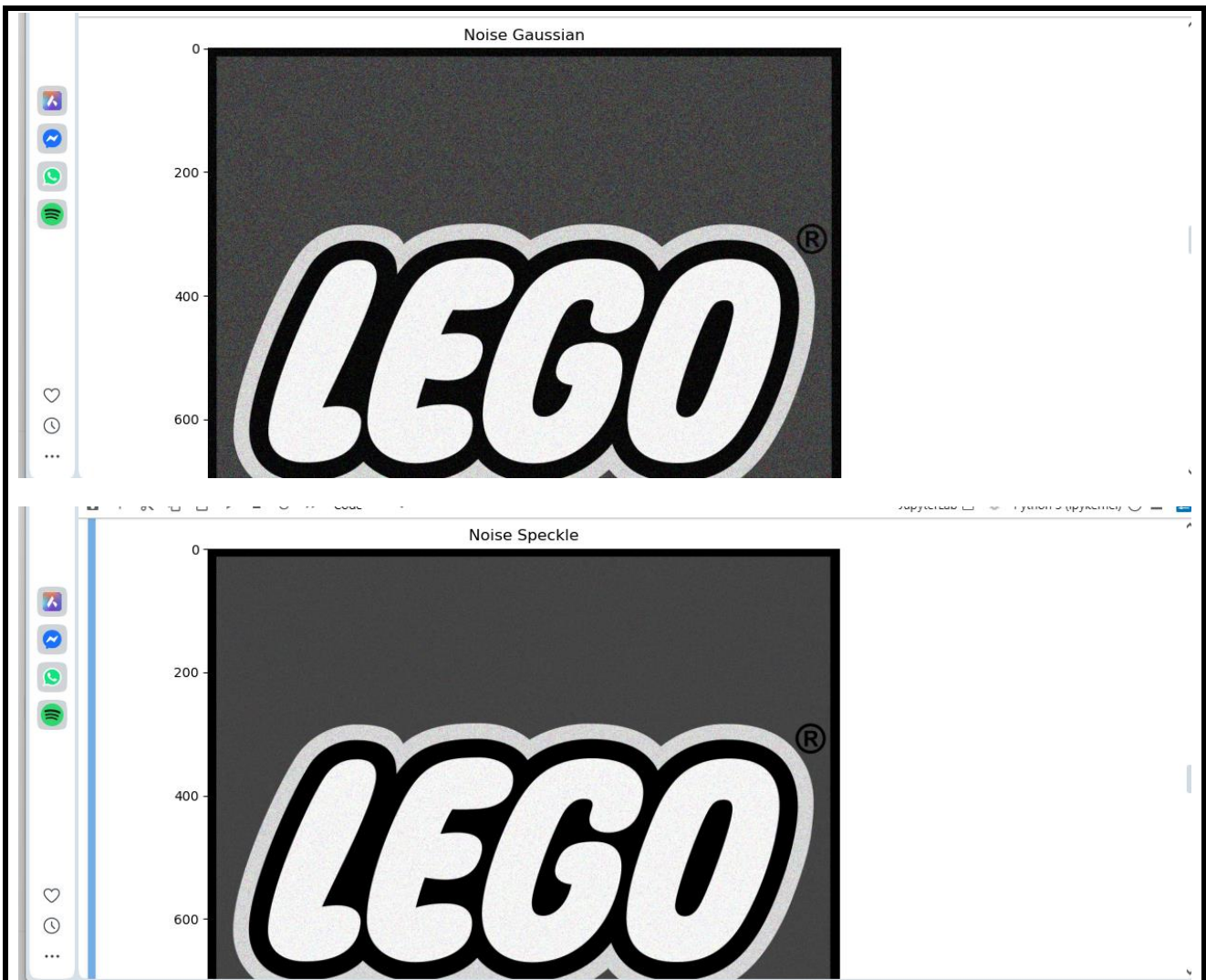
Menambahkan noise speckle, yang biasanya muncul seperti bercak acak menyebar. Mode `'speckle'` sering digunakan untuk mensimulasikan noise yang umum terjadi pada pencitraan medis atau radar.

27. `noise_img_snp = np.array(255*noise_img_snp, dtype='uint8')`

Mengubah nilai piksel dari float (0.0–1.0) menjadi uint8 (0–255) agar dapat ditampilkan dan diproses kembali dalam format standar citra. Hal ini karena `random_noise` menghasilkan array dengan nilai float.

Laporan 3





28. `fig, axs = plt.subplots(4,1, figsize = (10,40))`

Membuat figur (kanvas) dengan 4 baris dan 1 kolom gambar.

- a. Artinya akan ditampilkan 4 gambar secara vertikal.
- b. Ukuran `figsize = (10, 40)` memberikan ruang vertikal cukup untuk masing-masing subplot.

29. `ax = axs.ravel()`

Mengubah struktur `axs` menjadi array 1 dimensi, agar lebih mudah diakses satu per satu dengan indeks `[0]` hingga `[3]`.

30. `ax[0].imshow(citra_r, cmap = "gray")`

Menampilkan citra asli tanpa noise. `cmap = "gray"` dipakai meskipun gambar berwarna. Jika citra berwarna RGB, sebaiknya `cmap` dihapus agar warnanya tidak kacau.

31. `ax[1].imshow(noise_img_snp, cmap = "gray")`

Menampilkan citra dengan noise Salt & Pepper, yaitu bintik hitam dan putih acak.

32. `ax[2].imshow(noise_img_gaussian, cmap = "gray")`

Menampilkan citra dengan noise Gaussian, yaitu efek kabut atau buram secara acak.

33. `ax[3].imshow(noise_img_speckle, cmap = "gray")`

Menampilkan citra dengan noise Speckle, yaitu gangguan titik-titik menyebar di seluruh gambar.

34. `plt.show()`

Menampilkan seluruh subplot yang sudah dibuat.
 35. Dan ada beberapa outputannya yang ada.

```

* [21]: kernel_3_3 = np.ones((3, 3), np.float32)/9 #Nyimas Adinda Paradiza_202331041

* [22]: img_snp_avg_filter = cv2.filter2D(noise_img_snp, cv2.CV_8U, kernel_3_3,
      (-1, -1), delta = 0,
      borderType = cv2.BORDER_DEFAULT) #Nyimas Adinda Paradiza_202331041

      img_snp_median_filter = cv2.medianBlur(noise_img_snp, 3)

* [23]: img_gaussian_avg_filter = cv2.filter2D(noise_img_gaussian, cv2.CV_8U,
      kernel_3_3,
      (-1, -1), delta = 0,
      borderType = cv2.BORDER_DEFAULT) #Nyimas Adinda Paradiza_202331041

      img_gaussian_median_filter = cv2.medianBlur(noise_img_gaussian, 3)

* [24]: img_speckle_avg_filter = cv2.filter2D(noise_img_speckle, cv2.CV_8U,
      kernel_3_3,
      (-1, -1), delta = 0,
      borderType = cv2.BORDER_DEFAULT) #Nyimas Adinda Paradiza_202331041

      img_speckle_median_filter = cv2.medianBlur(noise_img_speckle, 3)
  
```

36. `kernel_3_3 = np.ones((3, 3), np.float32)/9`

Membuat kernel rata-rata 3x3 (average filter).

- Nilainya 1/9 di setiap elemen matriks 3x3.
- Digunakan untuk menghitung rata-rata nilai piksel di sekitarnya, sehingga bisa mengurangi noise ringan.

37. `img_snp_avg_filter = cv2.filter2D(...)`

Menerapkan filter rata-rata (average) pada citra `noise_img_snp` (yang terkena noise salt & pepper).

- Fungsi `cv2.filter2D` melakukan konvolusi antara citra dan kernel yang ditentukan.

38. `img_snp_median_filter = cv2.medianBlur(noise_img_snp, 3)`

Menerapkan filter median 3x3 pada citra yang sama.

- Median filter sangat efektif untuk menghilangkan salt & pepper noise, karena mengganti nilai piksel dengan nilai tengah (median) dari tetangganya.

39. `img_gaussian_avg_filter = cv2.filter2D(...)`

Menerapkan filter rata-rata untuk mengurangi Gaussian noise.

- Efektif tetapi bisa menyebabkan gambar menjadi sedikit buram.

40. `img_gaussian_median_filter = cv2.medianBlur(noise_img_gaussian, 3)`

Menerapkan median filter pada citra dengan Gaussian noise.

- Median kurang efektif untuk Gaussian noise, karena noise ini menyebar halus, bukan berupa outlier tajam seperti salt & pepper.

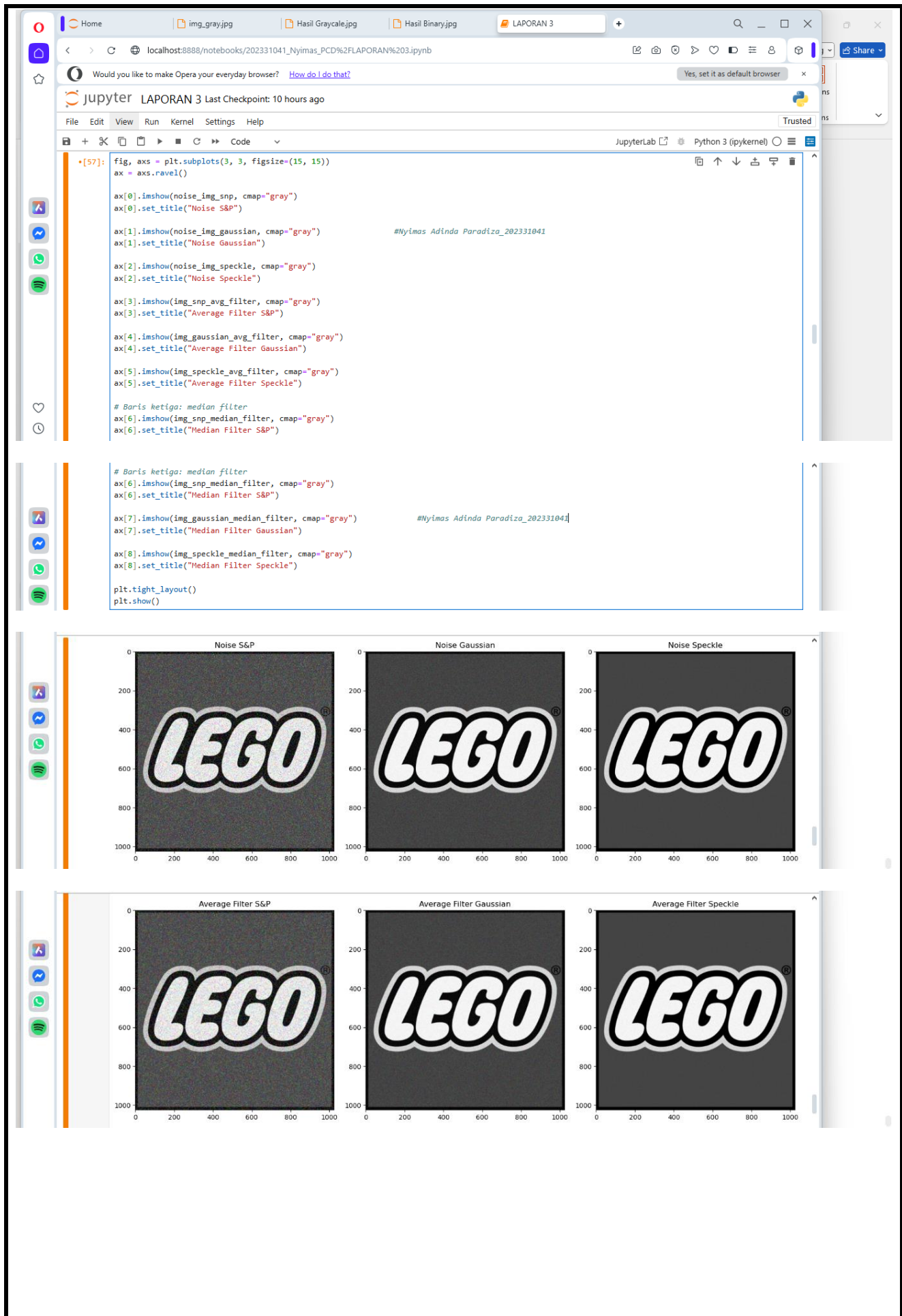
41. `img_speckle_avg_filter = cv2.filter2D(...)`

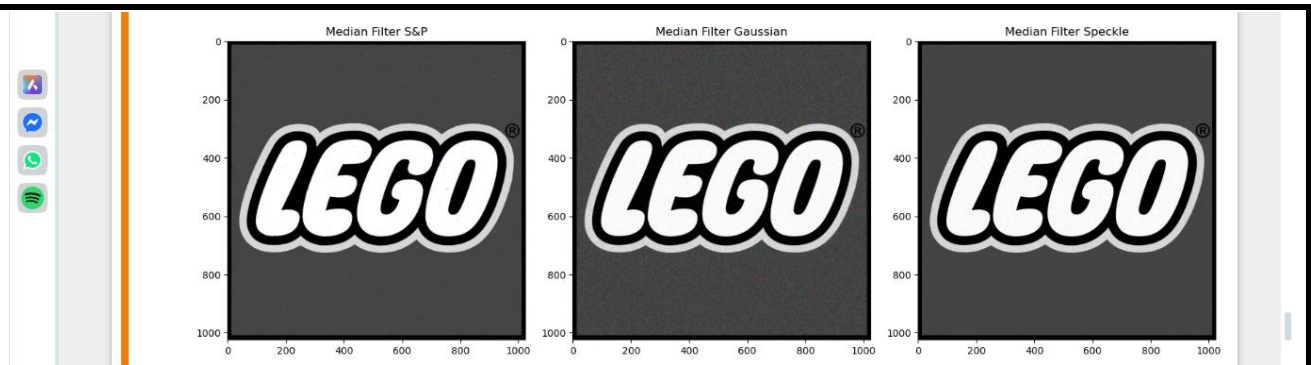
Menerapkan average filter untuk mereduksi speckle noise. Speckle noise juga bisa dikurangi dengan perataan piksel.

42. `img_speckle_median_filter = cv2.medianBlur(noise_img_speckle, 3)`

Menerapkan median filter pada speckle noise. Median bisa mengurangi efek bercak acak meskipun tidak setepat untuk salt & pepper.

Laporan 3





43. `fig, axs = plt.subplots(3, 3, figsize=(15, 15))`

Membuat kanvas untuk menampilkan 9 gambar dalam format 3 baris dan 3 kolom. Ukuran keseluruhan tampilan diatur agar besar dan jelas.

44. `ax = axs.ravel()`

Mengubah susunan array subplot dari bentuk dua dimensi menjadi satu dimensi, agar lebih mudah diakses satu per satu dengan indeks.

45. `ax[0].imshow(noise_img_snp, cmap="gray")`

Menampilkan gambar yang telah diberi noise salt & pepper, ditampilkan dalam skala abu-abu.

46. `ax[0].set_title("Noise S&P")`

Memberi judul pada gambar tersebut agar mudah dikenali sebagai citra dengan salt & pepper noise.

47. `ax[1].imshow(noise_img_gaussian, cmap="gray")`

Menampilkan gambar yang telah diberi noise Gaussian (berkabut), dalam skala abu-abu.

48. `ax[1].set_title("Noise Gaussian")`

Memberi judul pada gambar tersebut sebagai citra dengan Gaussian noise.

49. `ax[2].imshow(noise_img_speckle, cmap="gray")`

Menampilkan gambar dengan speckle noise, yaitu bercak acak di seluruh gambar.

50. `ax[2].set_title("Noise Speckle")`

Memberi judul pada gambar tersebut agar dikenali sebagai citra dengan speckle noise.

51. `ax[3].imshow(img_snp_avg_filter, cmap="gray")`

Menampilkan hasil dari average filter pada gambar salt & pepper noise. Filter ini meratakan piksel.

52. `ax[3].set_title("Average Filter S&P")`

Memberi judul pada gambar hasil filtering noise S&P dengan average filter.

53. `ax[4].imshow(img_gaussian_avg_filter, cmap="gray")`

Menampilkan hasil average filter pada citra Gaussian noise.

54. `ax[4].set_title("Average Filter Gaussian")`

Judul untuk gambar hasil filter Gaussian noise dengan average filter.

55. `ax[5].imshow(img_speckle_avg_filter, cmap="gray")`

Menampilkan hasil average filter pada citra speckle noise.

56. `ax[5].set_title("Average Filter Speckle")`

Judul untuk hasil filtering speckle noise dengan average filter.

57. `ax[6].imshow(img_snp_median_filter, cmap="gray")`

Menampilkan hasil median filter pada citra salt & pepper. Median filter cocok untuk noise ini.

58. `ax[6].set_title("Median Filter S&P")`

Judul untuk hasil median filter pada noise salt & pepper.

59. `ax[7].imshow(img_gaussian_median_filter, cmap="gray")`

Menampilkan hasil median filter pada citra Gaussian noise.

60. `ax[7].set_title("Median Filter Gaussian")`

Judul untuk hasil median filter pada noise Gaussian.

61. `ax[8].imshow(img_speckle_median_filter, cmap="gray")`

Menampilkan hasil median filter pada citra speckle noise.

62. `ax[8].set_title("Median Filter Speckle")`

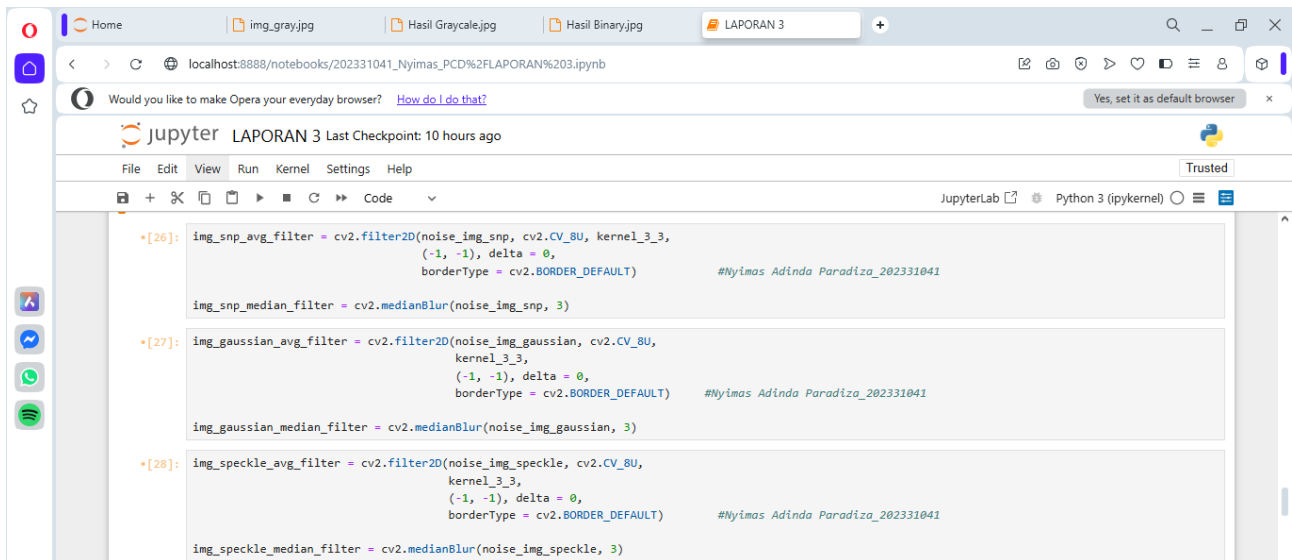
Judul untuk hasil median filter pada noise speckle.

63. `plt.tight_layout()`

Mengatur tata letak agar semua subplot rapi dan tidak saling menumpuk.

64. `plt.show()`

Menampilkan seluruh gambar dalam satu tampilan.



65. `img_snp_avg_filter = cv2.filter2D(noise_img_snp, cv2.CV_8U, kernel_3_3, (-1, -1), delta = 0, borderType = cv2.BORDER_DEFAULT)`

- Menerapkan filter rata-rata (average filter) ke citra `noise_img_snp` yang mengandung Salt & Pepper noise.
- Menggunakan `kernel_3_3`, yaitu matriks 3x3 berisi nilai 1/9 untuk meratakan nilai piksel di sekitarnya.
- Fungsi `cv2.filter2D()` melakukan konvolusi, yaitu menghitung rata-rata dari setiap blok piksel 3x3 dan menggantikan piksel pusat dengan nilai tersebut.
- Tujuannya adalah untuk mengurangi bintik hitam-putih acak, namun average filter kurang efektif untuk noise S&P karena bisa membuat hasil menjadi buram.

66. `img_snp_median_filter = cv2.medianBlur(noise_img_snp, 3)`

- Menerapkan median filter pada citra `noise_img_snp`.
- Ukuran filter 3 berarti mengambil blok 3x3 piksel, lalu mencari nilai tengah (median) dari 9 piksel tersebut.

- c. Median filter sangat efektif untuk menghilangkan Salt & Pepper noise karena menggantikan nilai ekstrem (0 dan 255) tanpa memengaruhi piksel lain secara berlebihan.
- d. Hasilnya lebih bersih dan tidak terlalu buram dibanding average filter.

67. `img_gaussian_avg_filter = cv2.filter2D(noise_img_gaussian, cv2.CV_8U,
kernel_3_3,
(-1, -1), delta = 0,
borderType = cv2.BORDER_DEFAULT)`

- a. Menerapkan average filter (penyaringan rata-rata) pada citra yang mengandung Gaussian noise.
- b. kernel_3_3 digunakan untuk merata-ratakan nilai intensitas piksel dalam area 3x3.
- c. Fungsi ini membantu mengurangi kabut halus yang disebabkan oleh noise Gaussian.
- d. Efektif untuk Gaussian noise, meskipun bisa menyebabkan sedikit blur pada detail gambar.

68. `img_gaussian_median_filter = cv2.medianBlur(noise_img_gaussian, 3)`

- a. Menerapkan median filter pada citra dengan Gaussian noise.
- b. Mengambil nilai tengah dari area 3x3 piksel untuk menggantikan nilai pusat.
- c. Kurang efektif untuk Gaussian noise, karena jenis noise ini menyebar secara halus dan median filter lebih cocok untuk noise berbentuk titik (outlier).
- d. Hasil mungkin tidak sebersih average filter dalam konteks Gaus

69. `img_speckle_avg_filter = cv2.filter2D(noise_img_speckle, cv2.CV_8U,
kernel_3_3,
(-1, -1), delta = 0,
borderType = cv2.BORDER_DEFAULT)`

- a. Menerapkan average filter (filter rata-rata) pada citra dengan speckle noise.
- b. Filter ini bekerja dengan menghitung rata-rata nilai piksel dalam area 3x3, lalu menggantikan nilai pusatnya.
- c. Speckle noise muncul sebagai bercak acak, dan average filter membantu meratakan intensitas piksel untuk mengurangi efek bercak tersebut.
- d. Cukup efektif, namun bisa mengaburkan detail gambar.

70. `img_speckle_median_filter = cv2.medianBlur(noise_img_speckle, 3)`

- a. Menerapkan median filter pada citra dengan speckle noise.
- b. Filter ini mengambil nilai tengah (median) dari area 3x3 piksel, menggantikan nilai pusat dengan nilai median tersebut.
- c. Median filter dapat mengurangi bercak-bercak ekstrem, namun hasilnya bisa bervariasi tergantung tingkat keparahan noise.
- d. Lebih halus dari average dalam beberapa kasus, tapi bisa kurang efektif bila speckle menyebar luas.

Laporan 3



75. `ax[1].imshow(img_snp_avg_filter, cmap="gray")`

Menampilkan hasil citra setelah diterapkan average filter, yang berfungsi untuk meredam noise salt & pepper.

76. `ax[1].set_title("Average Filter S&P Reduction")`

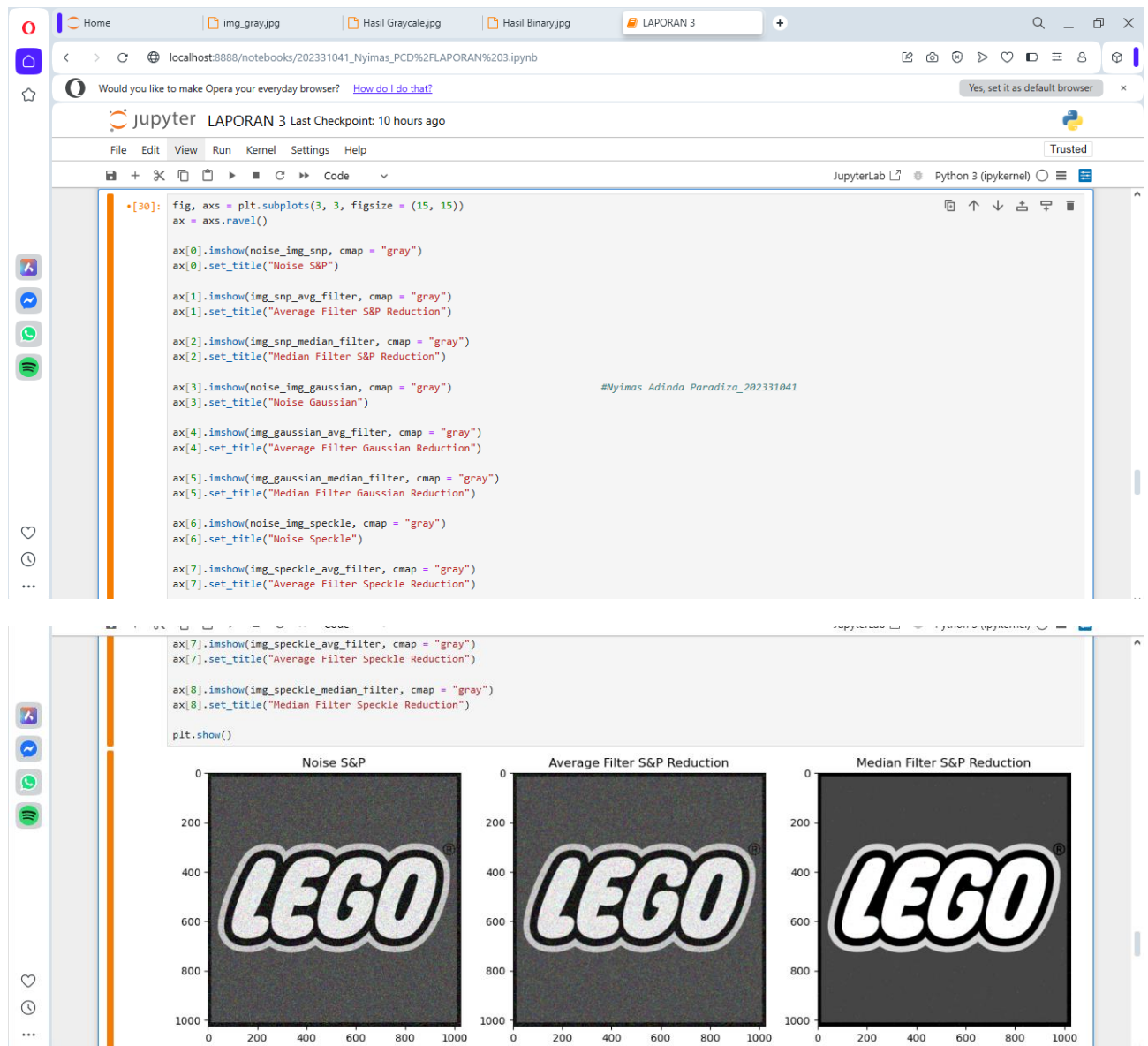
Memberi judul pada subplot kedua sebagai hasil perbaikan (reduksi noise) dengan average filter.

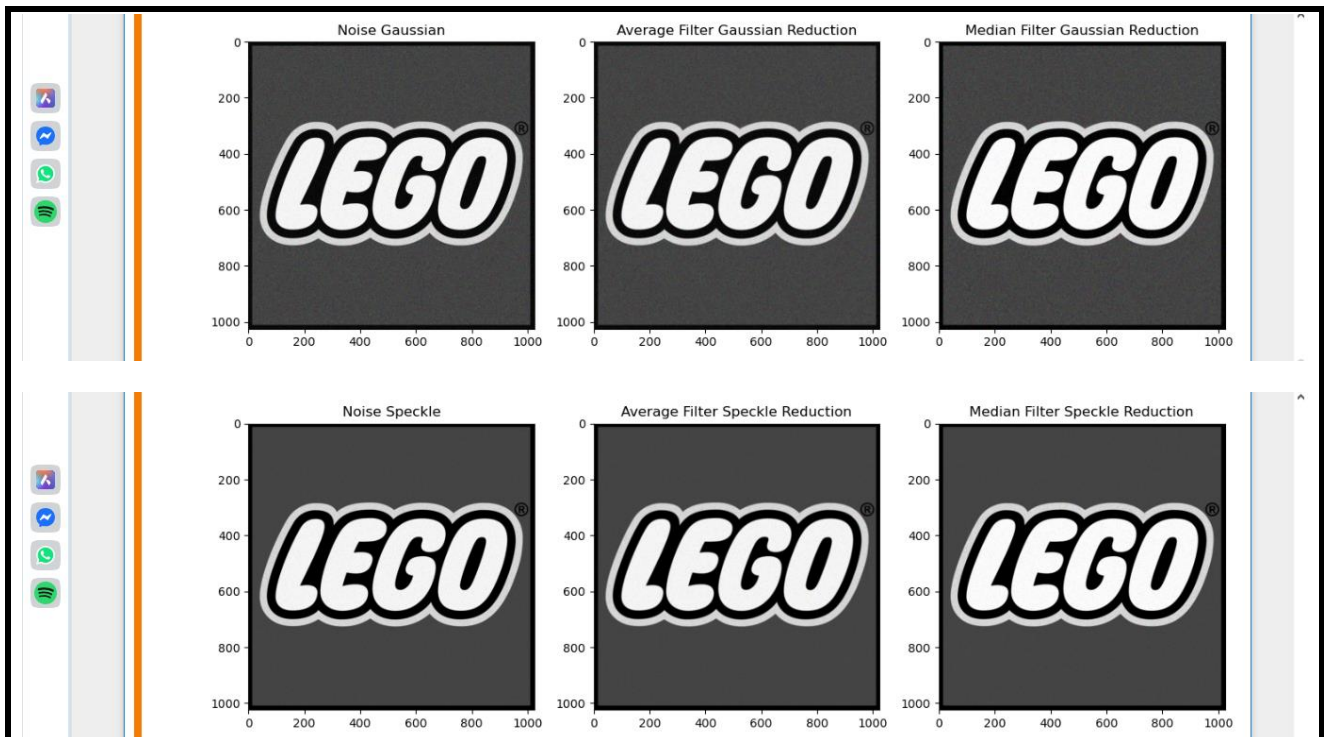
77. `plt.tight_layout()`

Mengatur posisi subplot agar tidak saling menumpuk dan tampil rapi.

78. `plt.show()`

Menampilkan seluruh figure ke layar.





79. `fig, axs = plt.subplots(3, 3, figsize = (15, 15))`

Membuat area tampilan (figure) berisi 9 subplot dalam 3 baris dan 3 kolom. Ukuran 15x15 inci digunakan agar semua citra terlihat dengan jelas dan proporsional.

80. `ax = axs.ravel()`

Mengubah array subplot dari bentuk 2 dimensi menjadi 1 dimensi, agar setiap subplot bisa diakses langsung dengan indeks seperti `ax[0]`, `ax[1]`, dan seterusnya.

81. `ax[0]`: Menampilkan citra yang telah diberi noise Salt & Pepper (bintik hitam dan putih).

82. `ax[1]`: Menampilkan hasil average filter pada citra tersebut untuk mengurangi noise.

83. `ax[2]`: Menampilkan hasil median filter yang secara umum lebih efektif untuk noise jenis ini.

84. `ax[3]`: Menampilkan citra dengan Gaussian noise (noise kabut acak).

85. `ax[4]`: Menampilkan hasil average filter yang cocok untuk noise Gaussian.

86. `ax[5]`: Menampilkan hasil median filter, meskipun biasanya kurang optimal untuk Gaussian noise.

87. `ax[6]`: Menampilkan citra yang diberi Speckle noise (bercak menyebar).

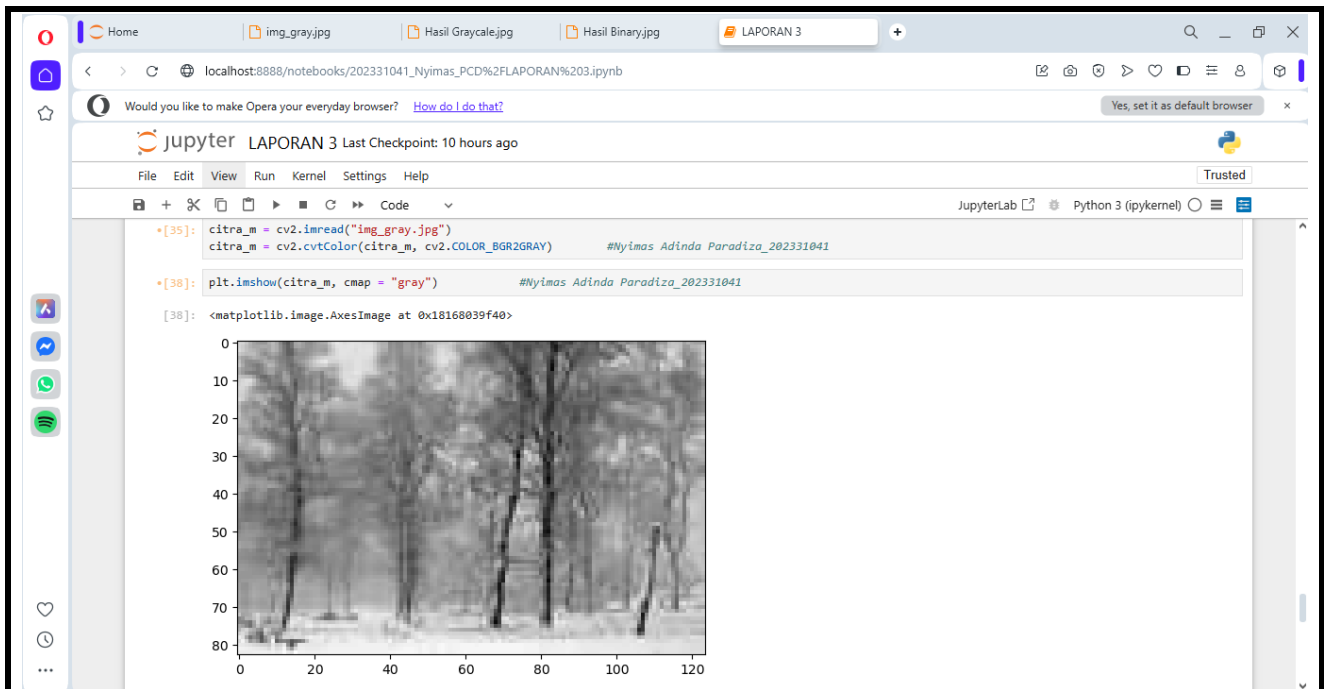
88. `ax[7]`: Menampilkan hasil average filter untuk mengurangi speckle.

89. `ax[8]`: Menampilkan hasil median filter pada noise speckle.

90. `plt.show()`

Menampilkan seluruh gambar secara bersamaan dalam satu tampilan visual.

Laporan 3



91. `citra_m = cv2.imread("img_gray.jpg")`

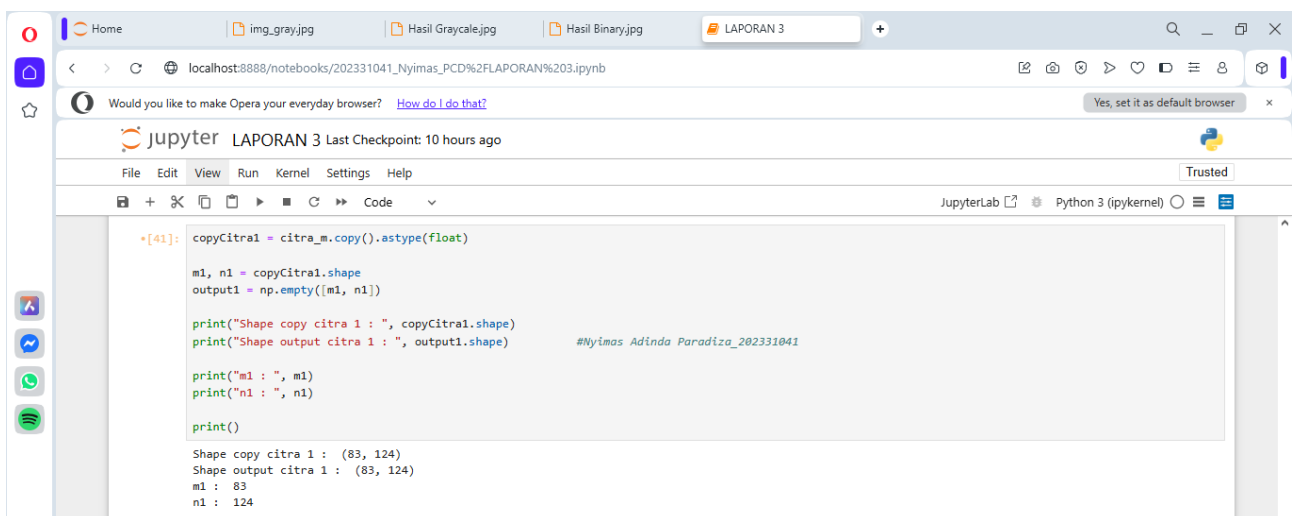
Digunakan untuk membaca citra dari file bernama "img_gray.jpg" menggunakan OpenCV. Secara default, citra dibaca dalam format BGR (Blue, Green, Red).

92. `citra_m = cv2.cvtColor(citra_m, cv2.COLOR_BGR2GRAY)`

Mengubah citra dari format warna BGR menjadi Grayscale (abu-abu). Konversi ini hanya menyisakan satu channel intensitas piksel (tanpa warna), cocok untuk pengolahan citra berbasis intensitas cahaya.

93. `plt.imshow(citra_m, cmap = "gray")`

Menampilkan citra dalam mode grayscale menggunakan Matplotlib. `cmap="gray"` memastikan citra ditampilkan dalam skala abu-abu, bukan warna default RGB.



94. `copyCitra1 = citra_m.copy().astype(float)`

- Membuat salinan dari citra grayscale (`citra_m`) dan mengubah tipe datanya menjadi float.
- Hal ini dilakukan agar nilai piksel bisa diproses lebih fleksibel, misalnya untuk operasi matematika seperti konvolusi atau normalisasi.

95. `m1, n1 = copyCitra1.shape`

- Mengambil ukuran dimensi dari citra (`m1` = jumlah baris/piksel tinggi, `n1` = jumlah kolom/piksel lebar).
- Digunakan untuk mengetahui ukuran gambar yang akan diproses.

96. `output1 = np.empty([m1, n1])`

- Membuat array kosong (`output1`) dengan ukuran yang sama seperti `copyCitra1`.
- Digunakan untuk menyimpan hasil pengolahan dari citra.

97. `print("Shape copy citra 1 : ", copyCitra1.shape)`

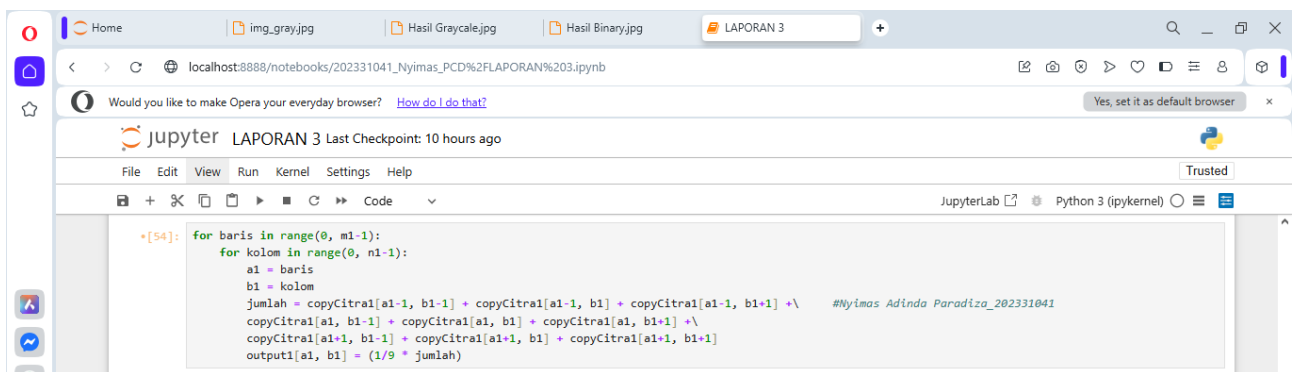
Menampilkan ukuran dari salinan citra sebagai verifikasi bentuk data.

98. `print("Shape output citra 1 : ", output1.shape)`

Menampilkan ukuran dari array hasil output untuk memastikan ukurannya sesuai dengan citra input.

99. `print("m1 : ", m1)` dan `print("n1 : ", n1)`

Menampilkan nilai tinggi (`m1`) dan lebar (`n1`) dari citra, untuk kepentingan debugging atau verifikasi.



1. `for baris in range(0, m1-1):`

Melakukan iterasi dari baris pertama hingga baris kedua terakhir citra. `m1` adalah jumlah baris citra. Dikurangi 1 agar tidak melewati batas saat mengambil piksel di sekitarnya (hindari error saat akses piksel di luar batas).

2. `for kolom in range(0, n1-1):`

Melakukan iterasi dari kolom pertama hingga kolom kedua terakhir, dengan alasan yang sama seperti baris — yaitu untuk menghindari akses indeks yang keluar dari batas gambar.

3. `a1 = baris` dan `b1 = kolom`

Menyimpan posisi indeks baris dan kolom ke dalam variabel `a1` dan `b1` untuk digunakan dalam perhitungan piksel.

4. `jumlah = ...`

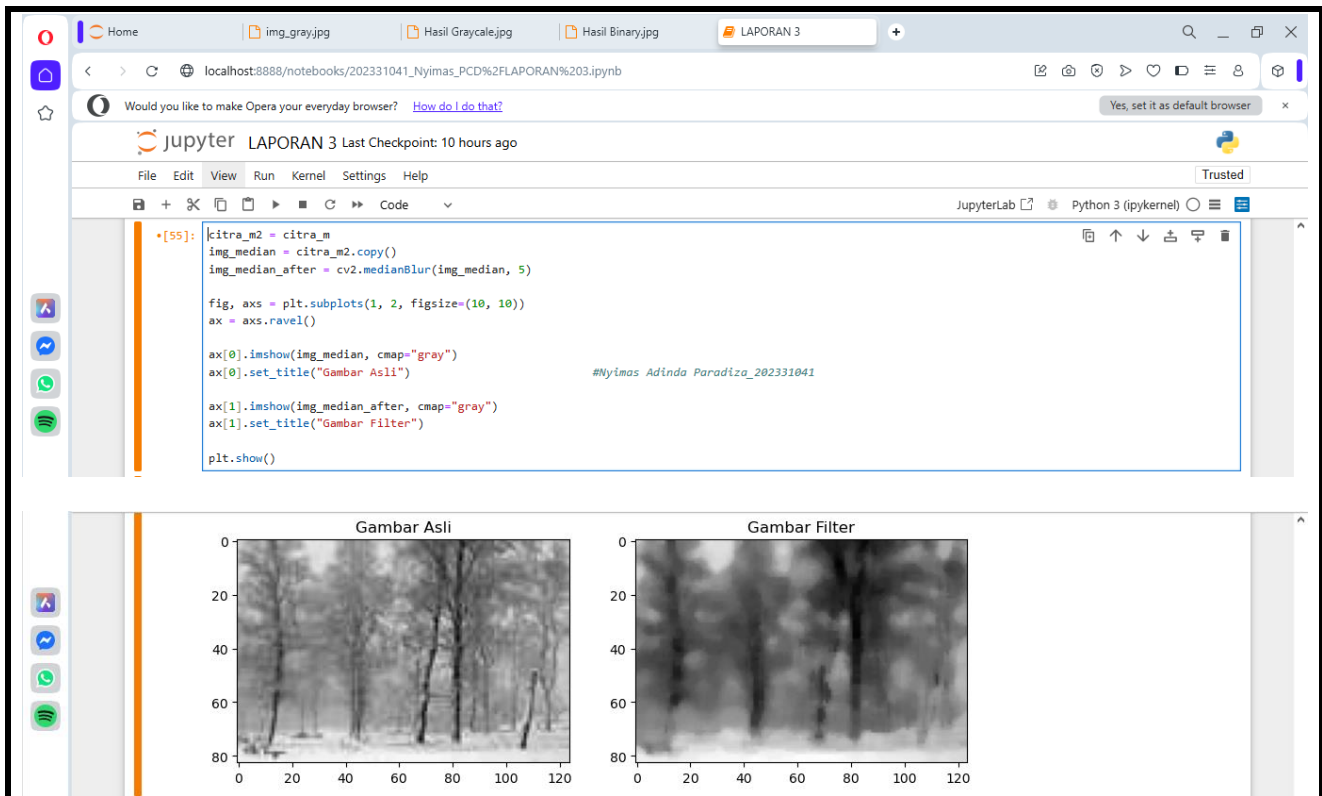
Menjumlahkan nilai piksel dari wilayah 3x3 yang mengelilingi titik pusat (`a1, b1`).

- Ini adalah proses konvolusi manual dengan kernel berisi nilai 1 (tanpa dikalikan di sini).
- Baris kode ini mengambil piksel dari atas-kiri sampai bawah-kanan dalam blok 3x3.

5. `output1[a1, b1] = (1/9 * jumlah)`

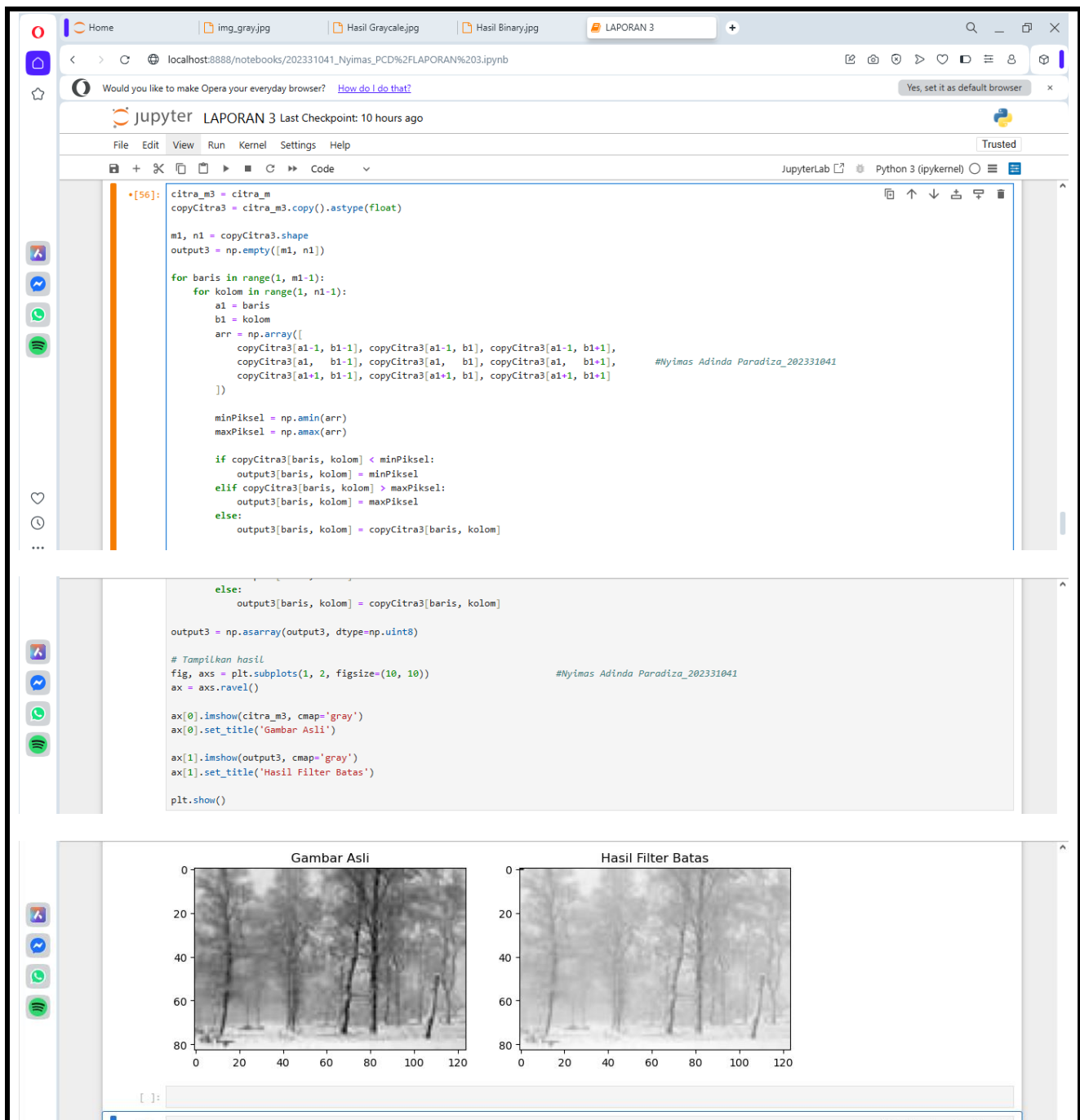
Menghitung rata-rata dari 9 piksel yang telah dijumlahkan dan menyimpan hasilnya ke posisi yang sama di array `output1`. Nilai `1/9` mewakili filter rata-rata 3x3 (total 9 piksel).

Laporan 3



6. `citra_m2 = citra_m`
Menyalin citra grayscale asli ke variabel baru `citra_m2`. Tujuannya agar data asli tetap ada dan tidak langsung dimodifikasi.
7. `img_median = citra_m2.copy()`
Membuat salinan baru dari `citra_m2` agar dapat diproses secara independen.
8. `img_median_after = cv2.medianBlur(img_median, 5)`
Menerapkan median filter pada gambar `img_median` dengan ukuran kernel 5x5.
 - a. Filter ini menggantikan setiap piksel dengan nilai tengah (median) dari tetangganya.
 - b. Ukuran 5 berarti area filter mencakup 25 piksel (5x5), memberikan efek perataan yang lebih kuat.
 - c. Median filter sangat efektif untuk menghilangkan noise tipe titik, seperti salt & pepper.
9. `fig, axs = plt.subplots(1, 2, figsize=(10, 10))`
Membuat area tampilan (figure) yang terdiri dari 2 gambar secara horizontal (1 baris, 2 kolom). Ukuran besar agar citra terlihat jelas.
10. `ax = axs.ravel()`
Mengubah array subplot menjadi bentuk 1 dimensi agar dapat diakses lebih mudah.
11. `ax[0].imshow(img_median, cmap="gray")`
Menampilkan gambar asli sebelum diberi filter, dalam skala abu-abu.
12. `ax[0].set_title("Gambar Asli")`
Memberi judul pada subplot pertama agar dikenali sebagai gambar sebelum filtering.
13. `ax[1].imshow(img_median_after, cmap="gray")`
Menampilkan gambar setelah diberi median filter, dalam skala abu-abu.
14. `ax[1].set_title("Gambar Filter")`
Memberi judul pada subplot kedua sebagai hasil dari proses filtering.
15. `plt.show()`
Menampilkan kedua gambar dalam satu tampilan untuk keperluan perbandingan visual.

Laporan 3



16. `citra_m3 = citra_m`
Menduplikasi citra asli grayscale ke variabel baru `citra_m3`, untuk menjaga citra asli tetap utuh saat diproses.
17. `copyCitra3 = citra_m3.copy().astype(float)`
Membuat salinan dari `citra_m3` dan mengonversi tipe data ke float, agar perhitungan lebih fleksibel tanpa pembulatan dini.
18. `m1, n1 = copyCitra3.shape`
Mengambil dimensi citra:
 - a. `m1` adalah jumlah baris (tinggi citra)
 - b. `n1` adalah jumlah kolom (lebar citra)
19. `output3 = np.empty([m1, n1])`

Menyiapkan array kosong (output3) untuk menyimpan hasil dari filter batas. Ukurannya sama dengan citra input.

20. for baris in range(1, m1-1):

 for kolom in range(1, n1-1):

 Dilakukan iterasi dari piksel baris dan kolom ke-1 hingga ke-m1-2 dan n1-2, agar tetap berada dalam batas gambar dan dapat mengambil tetangga 3x3.

21. arr = np.array([...])

 Mengambil nilai piksel dari area 3x3 di sekitar piksel pusat dan menyimpannya dalam array.

22. minPiksel = np.amin(arr)

 maxPiksel = np.amax(arr)

 Digunakan untuk mencari batas bawah dan batas atas dari nilai piksel pada area tersebut.

23. if copyCitra3[baris, kolom] < minPiksel:

 output3[baris, kolom] = minPiksel

elif copyCitra3[baris, kolom] > maxPiksel:

 output3[baris, kolom] = maxPiksel

else:

 output3[baris, kolom] = copyCitra3[baris, kolom]

 a. Jika nilai piksel lebih kecil dari minimum tetangganya → diganti dengan nilai minimum.

 b. Jika lebih besar dari maksimum tetangganya → diganti dengan nilai maksimum.

 c. Jika nilainya berada di antara minimum dan maksimum → tetap dipertahankan.

 d. Ini membantu mengurangi efek piksel ekstrem, terutama karena noise tipe impuls.

24. output3 = np.asarray(output3, dtype=np.uint8)

 Mengonversi hasil akhir kembali ke tipe data uint8 agar dapat ditampilkan sebagai citra.

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

 Menampilkan dua gambar dalam satu baris:

 a. ax[0]: Gambar asli (citra_m3)

 b. ax[1]: Gambar hasil setelah diberi filter batas (output3)