PENGOLAHAN CITRA · MATERI 04

Image Enhancement & Filtering

Pelajari konsep melalui contoh yang dekat dengan keseharian, coba ilustrasi dan foto sendiri, lalu jalankan praktikum secara bertahap di Google Colab.

Pengantar → konsep → ilustrasi dan upload foto → video → praktikum

1. Mulai dari gambaran sederhana

Bayangkan foto selembar daun yang memperlihatkan urat halus dan bintik-bintik gangguan. Ketika gambar dihaluskan, bintik dapat berkurang, tetapi urat daun juga mungkin ikut menghilang. Ketika tepinya dipertajam, bentuk daun dapat terlihat lebih tegas, tetapi bintik yang tersisa bisa makin menonjol. Contoh ini memperlihatkan bahwa memperbaiki citra bukan sekadar membuatnya lebih halus, lebih terang, atau lebih tajam. Kita perlu menentukan informasi apa yang ingin dipertahankan dan gangguan apa yang ingin dikurangi.

Pada materi ini, foto unggahan menjadi titik pembanding. Kode menambahkan dua jenis gangguan secara terkontrol: noise Gaussian yang menimbulkan fluktuasi nilai, serta noise impuls berupa titik hitam dan putih. Keduanya bukan klaim tentang proses pemotretan asli. Dengan foto sumber yang sama dan seed acak tetap, pengaruh tiga filter dapat diamati tanpa berganti-ganti input. Box blur merata-ratakan lingkungan, Gaussian blur memberi bobot lebih besar di dekat pusat, sedangkan median memilih nilai tengah. Perbedaan cara kerja itu menjelaskan mengapa satu filter tidak selalu unggul untuk semua gangguan.

Setelah penyaringan, pembahasan beralih ke sharpening, penambahan brightness, serta perubahan kontras melalui histogram equalization. Setiap operasi memiliki konsekuensi: penajaman dapat menguatkan noise, penambahan brightness dapat memotong daerah terang, dan penguatan kontras dapat membuat variasi yang tidak diinginkan semakin terlihat. Hasil dibaca melalui tampilan citra, detail pada lokasi yang sama, dan MSE terhadap foto sebelum noise tambahan. Angka MSE membantu menyatakan kedekatan nilai pixel, tetapi tidak menggantikan penilaian detail. Foto unggahan sendiri juga belum tentu bebas noise kamera, sehingga pembanding ini mempunyai batas yang perlu tetap diingat.

---

2. Pahami konsep secara berurutan

2.1 Enhancement dimulai dari tujuan yang jelas

Image enhancement mengubah citra agar informasi tertentu lebih mudah dilihat atau diproses. Mengurangi bintik, memperjelas batas benda, dan meningkatkan keterbacaan daerah gelap

adalah tujuan berbeda. Hasil yang tampak menarik belum tentu mempertahankan ukuran, tekstur, atau pola yang dibutuhkan analisis. Karena itu istilah lebih baik harus disertai alasan: misalnya gangguan berkurang sementara urat daun masih terlihat, bukan hanya seluruh permukaan menjadi halus.

Pengolahan tidak otomatis memulihkan informasi yang tidak pernah tertangkap kamera. Area yang sudah putih rata akibat pencahayaan berlebih tidak menyimpan rincian yang bisa dimunculkan kembali hanya dengan mengubah brightness. Dalam contoh ini, `referensi` adalah salinan foto unggahan sebelum noise tambahan. Istilah referensi menunjukkan fungsi pembanding dalam percobaan, bukan jaminan foto tersebut merupakan ground truth yang bebas semua gangguan.

2.2 Dua jenis noise dibuat dengan kondisi terkontrol

Noise Gaussian tambahan dibuat dari distribusi normal dengan rata-rata nol dan simpangan baku 20, kemudian dijumlahkan ke nilai foto. Hasil dibatasi ke 0–255 sebelum kembali menjadi uint8. Fluktuasi ini dapat menaikkan atau menurunkan komponen pixel; istilah Gaussian di sini menerangkan distribusi gangguan, bukan operasi penghalusan. Noise Gaussian dan Gaussian blur memiliki nama terkait, tetapi fungsi keduanya berbeda: yang pertama menambah gangguan, yang kedua menyaring citra.

Noise impuls pada `salt_pepper` mengganti lokasi terpilih menjadi hitam atau putih. Ambang acak di bawah 0,025 menghasilkan hitam, sedangkan di atas 0,975 menghasilkan putih. Peluang gabungannya 0,05, bukan janji tepat lima persen pixel pada setiap foto. Penggantian dilakukan pada ketiga channel lokasi yang sama. Generator `np.random.default_rng(42)` membuat urutan acak dapat diulang jika seluruh sel dijalankan dengan urutan yang sama; seed bukan parameter peningkat kualitas filter.

2.3 Kernel menentukan lingkungan yang dipakai filter

Filter spasial menggunakan tetangga di sekitar pixel. Kernel 5 × 5 mencakup lingkungan lima baris dan lima kolom. Box blur menghitung rata-rata dengan bobot seragam, sehingga satu nilai ekstrem dapat memengaruhi hasil di sekitarnya. Gaussian blur juga menggabungkan nilai lingkungan, tetapi memberi bobot lebih besar di dekat pusat. Contoh memakai `cv2.GaussianBlur` dengan ukuran (5, 5) dan sigma 1,2; ukuran lingkungan dan sebaran bobot sama-sama perlu diperhatikan.

Penggabungan nilai mengurangi perubahan cepat, baik yang berasal dari noise maupun detail asli. Kernel lebih besar dapat menghaluskan lebih kuat, tetapi garis tipis dan tekstur kecil berisiko melebur. Perubahan pada batas gambar juga dipengaruhi penanganan tepi oleh library. Untuk perbandingan awal, ketiga filter memakai ukuran lingkungan 5, bukan berarti kekuatan dan perilakunya identik. Gaussian pada contoh memakai ukuran ganjil positif; median menggunakan ukuran ganjil lebih besar dari satu.

2.4 Median memilih nilai tengah, bukan rata-rata

Median menyusun nilai lingkungan menurut besarannya lalu mengambil nilai tengah. Nilai impuls yang sangat gelap atau terang tidak langsung menarik hasil seperti pada rata-rata. Itulah sebabnya median sering berguna untuk gangguan salt-and-pepper. Pada citra berwarna, penyaringan berlangsung per channel. Namun median tidak memahami mana detail asli dan mana gangguan: pola tipis yang hanya menempati sedikit bagian lingkungan juga dapat tersisih.

Pada noise Gaussian yang tersebar, Gaussian blur dapat memberi kompromi penghalusan yang berbeda dari median dan box blur. Tidak ada peringkat yang harus selalu muncul. Bandingkan ketiga keluaran pada input noisy yang sama, kemudian periksa tepi dan tekstur di lokasi yang sama. Mencampur hasil dari foto, skala tampilan, atau tingkat gangguan berbeda membuat penyebab perubahan sulit dipisahkan.

2.5 Sharpening memperkuat perbedaan lokal

Contoh sharpening memakai kernel dengan nilai pusat 5, empat tetangga langsung bernilai −1, dan sudut bernilai nol. Melalui `cv2.filter2D`, perbedaan lokal diperkuat sehingga sebagian tepi tampak lebih tegas. Masukannya adalah keluaran median dari kondisi salt-and-pepper, bukan foto referensi dan bukan keluaran Gaussian. Urutan ini menunjukkan pengurangan gangguan lebih dahulu, kemudian penajaman.

Noise yang tersisa juga merupakan perubahan lokal, sehingga sharpening dapat ikut memperkuatnya. Garis terang atau gelap di sekitar tepi dan permukaan yang tampak kasar perlu dibaca sebagai kemungkinan efek pengolahan, bukan detail baru yang pasti benar. Dengan `ddepth=-1`, keluaran mempertahankan kedalaman input uint8; nilai yang melampaui rentang representasi mengalami saturasi. Penajaman tidak membatalkan hilangnya detail akibat blur.

2.6 Brightness dan equalization mengubah intensitas secara berbeda

Brightness pada kode menambahkan 40 ke setiap komponen foto referensi setelah konversi float. `np.clip` membatasi hasil ke 0–255 sebelum dikembalikan menjadi uint8. Konversi awal mencegah penjumlahan melingkar pada uint8, tetapi clipping tetap menghilangkan perbedaan di bagian terang. Beberapa nilai berbeda dapat berakhir sama-sama 255. Operasi ini menggeser nilai tersimpan, bukan mengulang pemotretan dengan pencahayaan yang lebih baik.

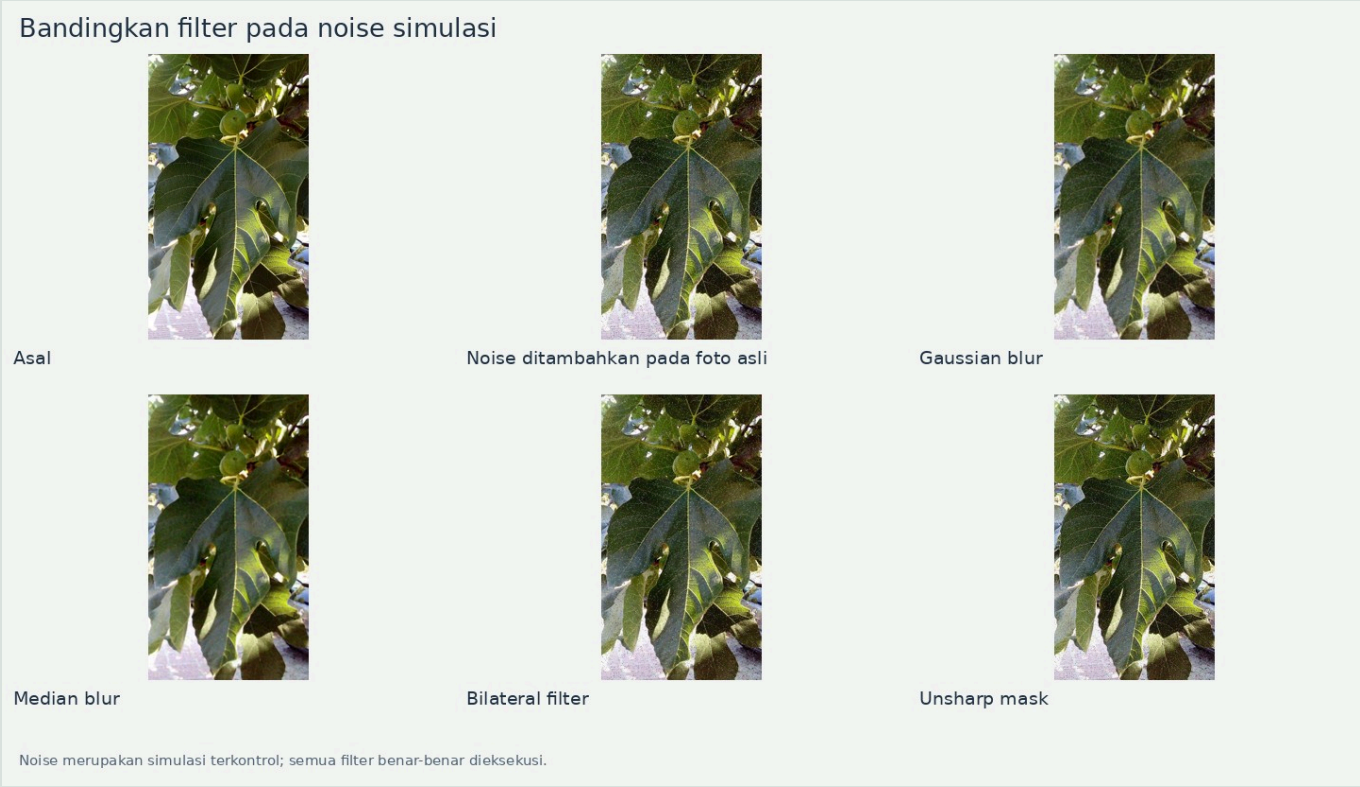
Histogram grayscale menghitung jumlah pixel pada tiap intensitas tanpa menunjukkan lokasi pixel tersebut. Equalization menggunakan distribusi kumulatif histogram untuk memetakan ulang intensitas dan dapat memperkuat kontras. Hasilnya tidak harus memiliki histogram rata sempurna dan dapat membuat noise lebih terlihat. Kode menerapkan `cv2.equalizeHist`

pada `gray`, bukan pada masing-masing channel BGR atau pada hasil brightness. Ketiga panel membandingkan cabang pengolahan berbeda dari referensi, bukan rangkaian operasi berturut-turut.

2.7 MSE melengkapi pengamatan, bukan menentukan seluruh kualitas

MSE adalah rata-rata kuadrat selisih nilai keluaran terhadap referensi. Fungsi `mse` mengubah kedua array menjadi float sebelum pengurangan agar selisih negatif dan kuadratnya tidak rusak oleh rentang uint8. Dalam contoh ini, rata-rata mencakup lokasi dan channel. Nilai lebih kecil berarti lebih dekat ke foto sebelum noise tambahan menurut ukuran tersebut, bukan bukti semua detail lebih terbaca.

Pembacaan yang utuh menggabungkan angka yang benar-benar dicetak dengan keutuhan tepi, tekstur, dan sisa bintik. Referensi mungkin sudah memiliki noise kamera; kedekatan terhadapnya tidak membuktikan pemulihan citra bersih yang sebenarnya. Tanpa pembanding sebelum gangguan, jangan mengarang skor pemulihan. Pilihan filter perlu dijelaskan melalui tujuan dan keluaran yang tersedia, bukan melalui anggapan bahwa satu teknik selalu terbaik.



Noise merupakan simulasi terkontrol; semua filter benar-benar dieksekusi.

Gambar di atas dipertahankan dari materi sumber. Ilustrasi sumber tidak dengan sendirinya membuktikan kinerja pada foto lain; perhatikan keterangan, metode, dan batasannya.

---

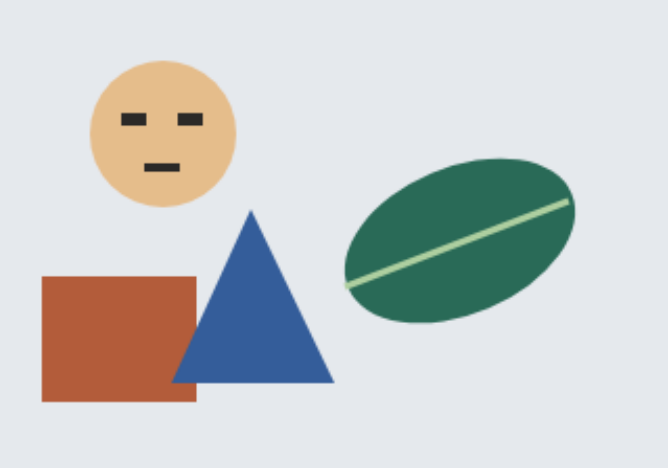
3. Ilustrasi interaktif sebelum praktikum

Noise terkontrol, filtering, dan sharpening

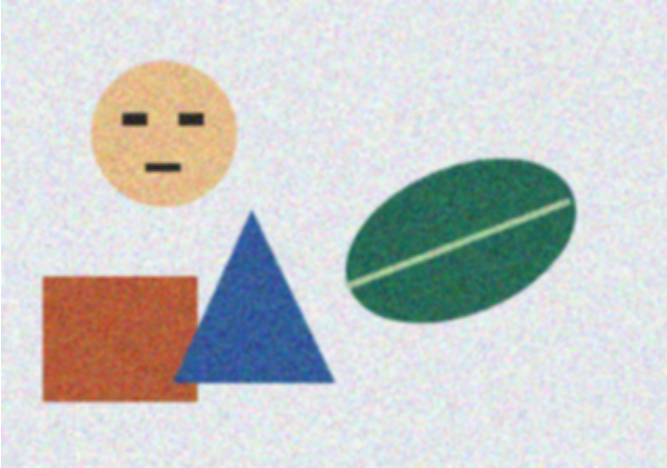
Tambahkan noise sintetis dengan seed tetap 42 lalu bandingkan box, Gaussian, median, atau unsharp mask pada array noise yang sama. MSE dihitung terhadap foto unggahan sebelum noise tambahan; foto itu bukan jaminan ground truth bebas noise kamera.

> Noise tambahan dibuat secara terkontrol agar perbandingan filter memakai input yang sama. Foto unggahan adalah referensi sebelum noise tambahan, bukan jaminan ground truth bebas noise kamera. MSE dihitung dari piksel, bukan bukti bahwa hasil paling halus selalu terbaik.

Pakai gambar contoh atau pilih foto sendiri dari HP/komputer. Foto dibaca dan diproses **hanya di browser**, tanpa dikirim atau disimpan ke server. Contoh buatan tidak digunakan sebagai bukti hasil eksperimen. Gunakan media berizin dan hindari data sensitif.



Gambar contoh sintetis · citra kerja 320 × 224 piksel



Hasil · 320 × 224 piksel; ukuran tampilan menyesuaikan kotak.

Parameter: Metode = gaussian · Kernel (sisi ganjil) = 3 · Jenis noise sintetis = gaussian · Noise: σ Gaussian atau persen pixel impuls = 20

gaussian, kernel 3 × 3: MSE 76.939 terhadap input referensi; MSE input noise 335.357.

Noise gaussian; σ=20; seed xorshift32=42, Gaussian Box–Muller (bukan RNG NumPy).

Noise dibuat ulang identik untuk setiap metode/kernel pada parameter noise dan input yang sama. Impuls mengubah ketiga channel pixel bersama; Gaussian independen per channel.

Semua tepi memakai border replicate (koordinat dijepit ke pixel terdekat). Gaussian separabel: bobot [1, 2, 1], dinormalisasi; median per channel RGB.

Sharpen adalah unsharp mask: hasil = noisy + 1 × (noisy − Gaussian(noisy)); faktor tetap 1, bukan denoise lalu sharpening.

Hitung dengan float, clamp 0–255 dan Math.round saat menjadi byte; tanpa overflow uint8. MSE RGB (alpha dikecualikan) memakai array byte yang benar-benar ditampilkan. Alpha sumber tidak diubah.

MSE lebih rendah tidak membuktikan kualitas terbaik: lihat kehilangan detail/tepi. Referensi ialah input sebelum noise tambahan, bukan ground truth bebas noise.

▼ **Angka dan parameter hasil perhitungan ilustrasi**


```
{
  "mse": 76.93873697916666,
  "noiseMSE": 335.35692429315475,
  "seed": 42,
  "kernel": 3,
  "noiseAmount": 20,
  "totalPixels": 71680,
  "sharpenFactor": 1
}
```


Pada PDF/cetakan, ilustrasi ini menjadi cuplikan statis menggunakan gambar contoh. Upload foto dan pengubahan parameter tersedia di versi web; foto pribadi tidak dimasukkan ke PDF unduhan.

4. Video pembelajaran

OpenCV Python Tutorial For Beginners 18 - Smoothing Images | Blurring Images OpenCV — ProgrammingKnowledge.

Fokus saat menonton: Bandingkan averaging, Gaussian, median dan bilateral: apa yang terjadi pada noise serta batas objek?

Buka video di YouTube jika pemutar tidak tersedia

Video referensi berbahasa Inggris. Bila ada setup lokal, API atau versi model berbeda, pelajari konsepnya dan tetap gunakan alur Google Colab pada praktikum bab ini. Ketersediaan pemutar mengikuti YouTube, pemilik video dan jaringan.

▼ **Referensi dan atribusi gambar**

- OpenCV — Smoothing Images
- OpenCV — Histograms: Histogram Equalization
- NumPy — clip

Foto daun: Ανώνυμος Βικιπαιδιστής, Wikimedia Commons, CC BY 3.0; diubah ukuran/intensitas untuk ilustrasi. Foto bus: contoh aset Ultralytics. Foto astronaut: NASA melalui scikit-image. Panel diagram/simulasi ditandai pada caption; bukan hasil eksperimen yang dikarang. Hak penggunaan sumber tetap mengikuti sumber asli.

---

5. Praktikum / latihan Google Colab

Buka Google Colab, buat notebook baru, dan jalankan satu sel kode untuk setiap langkah secara berurutan dengan `Shift` + `Enter`. Variabel dan upload disiapkan di modul ini; tidak mengambil variabel dari bab lain.

Jalankan sel import dan upload sampai `bgr` tersedia, lalu sel pembentukan `referensi` dan `input_noise`. Generator `rng` berseed 42 menghasilkan `gaussian_noise` dengan sigma 20 dan `salt_pepper` dengan peluang impuls gabungan 0,05. Pertahankan foto dan seluruh sel pembentukan noise yang sama ketika membandingkan filter. Sel berikutnya mengisi `hasil_filter` dengan input noisy, Gaussian (5, 5) sigma 1,2, median 5, dan box (5, 5). Baca MSE dari cetakan eksekusi aktual; fungsi `mse` memakai selisih float terhadap `referensi`. Bandingkan pula daerah detail yang sama, misalnya melalui irisan `img[y1:y2, x1:x2]`, tanpa menyimpulkan kualitas hanya dari angka terkecil.

Sel sharpening mengambil `denoised` dari `hasil_filter["Salt-pepper tambahan (p=0.05)"]["Median k=5"]`, lalu menerapkan `kernel_sharp` melalui `cv2.filter2D` untuk menghasilkan `sharp`. Bandingkan panel sebelum dan sesudahnya: tepi dapat lebih tegas, tetapi sisa bintik atau garis di sekitar tepi dapat ikut menguat. Jika parameter filter atau foto sebelumnya berubah, jalankan kembali sel pembanding sebelum sel sharpening agar `denoised` sesuai kondisi terbaru.

Sel terakhir membentuk `bright` langsung dari `referensi` dengan penambahan float +40 dan clipping, kemudian membuat `bright_gray`. Sementara itu, `equalized` berasal dari `cv2.equalizeHist(gray)`, bukan dari `bright_gray`. Baca citra bersama histogram dari

`cv2.calcHist`: sumbu horizontal adalah intensitas, sumbu vertikal jumlah pixel. Cetakan jumlah nilai grayscale 255 membantu memeriksa saturasi terang, tetapi tidak menghitung seluruh channel warna yang mungkin terpotong. Jika runtime direset, jalankan sel dari awal supaya foto, referensi, dan hasil perbandingan tetap saling sesuai.

Sel 1 — 1. Import library

Ketik di notebook Colab baru pada bagian awal notebook. Modul ini mandiri; tidak menggunakan variabel dari modul lain.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files
```

Sel 2 — 2. Upload satu foto

Pilih satu foto JPEG/PNG uint8, maksimum 10 MiB dan 12 juta pixel. Pembacaan mempertahankan channel asli untuk validasi, lalu menyiapkan BGR tiga channel. PNG transparan tidak dipakai agar tidak ada background tersembunyi.

```
uploaded = files.upload()
if len(uploaded) != 1:
    raise ValueError("Pilih tepat satu foto JPEG/PNG.")
nama, data = next(iter(uploaded.items()))
if not nama.lower().endswith((".jpg", ".jpeg", ".png")):
    raise ValueError("Gunakan file JPEG atau PNG.")
if not data or len(data) > 10 * 1024 * 1024:
    raise ValueError("File kosong atau melebihi 10 MiB.")
raw = cv2.imdecode(np.frombuffer(data, np.uint8),
cv2.IMREAD_UNCHANGED)
if raw is None:
    raise ValueError("File tidak dapat didecode sebagai gambar.")
h, w = raw.shape[:2]
if h * w > 12_000_000 or min(h, w) < 32 or raw.dtype != np.uint8:
    raise ValueError("Gunakan gambar uint8, sisi minimal 32 pixel,
<=12 MP.")
if raw.ndim == 2:
    bgr = cv2.cvtColor(raw, cv2.COLOR_GRAY2BGR)
elif raw.ndim == 3 and raw.shape[2] == 3:
    bgr = raw.copy()
```

```
else:
    raise ValueError("Gunakan foto grayscale/3 channel tanpa
alpha.")
print("Foto:", nama, "shape BGR:", bgr.shape)
plt.figure(figsize=(6, 5))
plt.imshow(cv2.cvtColor(bgr, cv2.COLOR_BGR2RGB))
plt.title("Foto unggahan")
plt.axis("off")
plt.show()
```

Sel 3 — 3. Buat dua kondisi noise terkontrol

Foto unggahan adalah referensi sebelum noise tambahan, bukan jaminan ground truth bebas noise kamera. Noise Gaussian dan impuls dihasilkan dengan seed tetap dan diberi label jelas.

```
referensi = bgr.copy()
rng = np.random.default_rng(42)
noise = rng.normal(0, 20, referensi.shape)
gaussian_noise = np.clip(referensi.astype(np.float32) + noise, 0,
255).astype(np.uint8)
salt_pepper = referensi.copy()
acak = rng.random(referensi.shape[:2])
salt_pepper[acak < 0.025] = 0
salt_pepper[acak > 0.975] = 255
input_noise = {"Gaussian tambahan (sigma=20)": gaussian_noise,
               "Salt-pepper tambahan (p=0.05)": salt_pepper}
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
for ax, judul, img in zip(axes, ["Referensi sebelum noise tambahan"]
+ list(input_noise),
                          [referensi] + list(input_noise.values())):
    ax.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    ax.set_title(judul); ax.axis("off")
plt.tight_layout(); plt.show()
```

Sel 4 — 4. Bandingkan tiga teknik filtering

Sel memproses input noise yang sama dengan Gaussian, median, dan box blur. MSE dihitung pada array float untuk mencegah overflow. Semua angka berasal dari eksekusi file unggahan.

```
hasil_filter = {}
def mse(a, b):
```

```
    selisih = a.astype(np.float32) - b.astype(np.float32)
    return float(np.mean(selisih ** 2))
fig, axes = plt.subplots(2, 4, figsize=(18, 9))
for baris, (jenis, noisy) in enumerate(input_noise.items()):
    hasil = {"Input noise": noisy,
             "Gaussian k=5 sigma=1.2": cv2.GaussianBlur(noisy, (5,
5), 1.2),
             "Median k=5": cv2.medianBlur(noisy, 5),
             "Box k=5": cv2.blur(noisy, (5, 5))}
    hasil_filter[jenis] = hasil
    print("Kondisi:", jenis)
    for ax, (teknik, img) in zip(axes[baris], hasil.items()):
        skor = mse(img, referensi)
        print(teknik, "MSE terhadap referensi sebelum noise:",
round(skor, 3))
        ax.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        ax.set_title(f"{teknik}; MSE={skor:.2f}"); ax.axis("off")
plt.tight_layout(); plt.show()
```

Sel 5 — 5. Denoise lalu sharpening

Contoh mempertajam hasil median pada input salt-pepper, kemudian membandingkan dengan sebelum sharpening. Kernel dan urutan operasi harus dicatat.

```
denoised = hasil_filter["Salt-pepper tambahan (p=0.05)"]["Median
k=5"]
kernel_sharp = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]],
dtype=np.float32)
sharp = cv2.filter2D(denoised, ddepth=-1, kernel=kernel_sharp)
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
for ax, judul, img in zip(axes, ["Median: sebelum sharpening",
"Median + sharpening"],
                          [denoised, sharp]):
    ax.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    ax.set_title(judul); ax.axis("off")
plt.tight_layout(); plt.show()
```

Sel 6 — 6. Brightness, equalization, dan histogram

Brightness +40 menggunakan float dan clipping. Equalization diterapkan pada grayscale saja. Histogram dilengkapi label sumbu agar perubahan dapat dibaca.

```
bright = np.clip(referensi.astype(np.float32) + 40, 0,
255).astype(np.uint8)
gray = cv2.cvtColor(referensi, cv2.COLOR_BGR2GRAY)
bright_gray = cv2.cvtColor(bright, cv2.COLOR_BGR2GRAY)
equalized = cv2.equalizeHist(gray)
fig, axes = plt.subplots(2, 3, figsize=(15, 8))
for kolom, (judul, img) in enumerate([
        ("Grayscale referensi", gray), ("Brightness +40",
bright_gray),
        ("Histogram equalization", equalized)]):
    axes[0, kolom].imshow(img, cmap="gray", vmin=0, vmax=255)
    axes[0, kolom].set_title(judul); axes[0, kolom].axis("off")
    hist = cv2.calcHist([img], [0], None, [256], [0, 256]).ravel()
    axes[1, kolom].plot(np.arange(256), hist)
    axes[1, kolom].set_xlim(0, 255)
    axes[1, kolom].set_xlabel("Intensitas"); axes[1,
kolom].set_ylabel("Jumlah pixel")
print("Pixel grayscale bernilai 255 sebelum/sesudah brightness:",
      int(np.count_nonzero(gray == 255)),
int(np.count_nonzero(bright_gray == 255)))
plt.tight_layout(); plt.show()
```

Jika terjadi error

Error pada ukuran kernel median/Gaussian.

Gunakan ukuran ganjil positif; median contoh memakai 5, Gaussian memakai (5,5). Pastikan input uint8.

Brightness menghasilkan pixel gelap yang tidak diharapkan.

Hindari penambahan langsung pada uint8; konversi float, clip 0–255, lalu astype uint8.

Filter dinilai terbaik hanya karena MSE terkecil.

Periksa detail dan tepi pada wilayah yang sama; jelaskan trade-off serta keterbatasan referensi.

Equalization membuat noise semakin terlihat.

Pahami bahwa penguatan kontras juga menguatkan variasi noise; bandingkan dengan denoise sebelum equalization.

Runtime Colab dan file upload bersifat sementara. Simpan output penting pada notebook/Drive sebelum sesi berakhir. Jika runtime direset, jalankan ulang sel dari awal.

Gunakan media berizin dan hindari unggahan data sensitif tanpa izin.
