PENGOLAHAN CITRA · MATERI 05

Thresholding dan Segmentasi Citra

Belajar memilih bagian gambar secara bertahap: pahami piksel dan mask, coba ilustrasi, lalu terapkan binary, adaptive, dan Otsu pada foto di Google Colab.

Pengantar → konsep → ilustrasi interaktif → video → praktikum

1. Mulai dari gambaran sederhana

Bayangkan Anda mempunyai foto daun di atas kertas. Anda ingin mengambil bagian daunnya saja, tanpa membawa kertas yang menjadi latarnya. Bagi manusia, perbedaannya mudah terlihat. Komputer perlu diberi aturan untuk menentukan piksel mana yang dipertahankan dan piksel mana yang tidak dipakai.

Proses membagi bagian gambar seperti ini disebut **segmentasi**. Pada materi ini, kita memakai **thresholding**: memilih piksel berdasarkan tingkat gelap dan terang, dengan sebuah batas yang disebut *nilai ambang* atau `T`. Hasil pilihannya disimpan dalam **mask**, yaitu gambar penanda hitam putih.

Pada contoh kita, **putih pada mask berarti dipilih** dan hitam berarti tidak dipilih. Warna ini bukan warna asli benda. Daun yang gelap tetap dapat ditandai putih karena memang itulah bagian yang ingin diambil.

Bagian yang dipilih disebut *foreground*; bagian yang tidak dipilih disebut *background*. Istilah tersebut mengikuti tujuan pengolahan, bukan selalu mengikuti benda mana yang paling dekat dengan kamera. Threshold belum mengetahui bahwa suatu bentuk adalah daun. Jika latar mempunyai intensitas mirip, latar bisa ikut terambil.

Kenali piksel dan mask
Pahami tiga metode
Coba ilustrasi
Kerjakan praktikum

Setelah membaca materi, Anda diharapkan dapat memilih polaritas yang sesuai, membandingkan binary, adaptive, dan Otsu, memakai mask untuk memisahkan gambar, serta menjelaskan bagian hasil yang masih keliru.

> **Bekal awal:** memahami gambar warna dan grayscale. Praktikum di halaman ini menyediakan import dan upload sendiri; tidak perlu memakai variabel dari materi sebelumnya.

2. Pahami cara kerjanya secara berurutan

2.1 Dari gambar warna ke grayscale

Gambar tersusun dari piksel. Pada gambar warna, setiap lokasi mempunyai beberapa nilai warna. Ketika diubah menjadi **grayscale**, setiap piksel diwakili satu nilai tingkat keabuan. Pada citra `uint8` yang dipakai di praktikum, nilai 0 paling gelap dan 255 paling terang.

Grayscale belum sama dengan mask. Grayscale masih mempunyai banyak tingkat keabuan. Mask biner pada contoh ini hanya memakai 0 atau 255 sebagai penanda pilihan. Kita mengolah grayscale terlebih dahulu, kemudian membuat keputusan melalui threshold.

Sebelum membandingkan metode, contoh memberikan *Gaussian blur* ringan dengan kernel `5 × 5`. Ini membantu menghaluskan fluktuasi kecil, tetapi juga bisa melembutkan tepi dan tangkai. Ketiga metode harus memakai hasil blur yang sama agar perbandingannya tidak tercampur dengan perlakuan awal yang berbeda.

2.2 Threshold global: satu batas untuk seluruh gambar

Binary global membandingkan semua piksel dengan satu nilai `T`. Pada `THRESH_BINARY`, intensitas yang **lebih besar** dari T menjadi 255; sisanya menjadi 0. Contoh awal memakai `T = 127`, bukan karena angka ini selalu terbaik, tetapi sebagai titik awal yang mudah dibandingkan.

Perhatikan batasnya: intensitas 127 belum lebih besar dari ambang 127. Karena itu, hasilnya masih hitam pada mode binary. Intensitas 128 sudah melewati ambang sehingga menjadi putih.

Contoh empat piksel dengan T = 127; bukan hasil pengukuran sebuah foto

Intensitas piksel	THRESH_BINARY	THRESH_BINARY_INV
90	0 · hitam	255 · putih
127	0 · hitam	255 · putih
128	255 · putih	0 · hitam
220	255 · putih	0 · hitam

Jika tingkat keabuan objek dan latar saling tumpang tindih, satu angka tidak dapat memisahkan seluruh bagian secara sempurna. Mengubah T bisa membantu satu bagian, tetapi menambah kesalahan di bagian lain.

2.3 Polaritas: putihnya harus mewakili bagian yang diinginkan

Polaritas adalah arah pemilihan. `THRESH_BINARY_INV` membalik hasil binary: piksel yang kurang dari atau sama dengan T menjadi putih, sedangkan yang lebih besar menjadi hitam.

Misalnya, daun gelap berada pada kertas terang. Mode inverse dapat menjadi titik awal agar daun ditandai putih. Jika yang putih justru kertasnya, periksa polaritas sebelum mengganti banyak parameter. Dalam praktikum, pilihan ini diatur melalui `OBJEK_GELAP`.

Membalik hitam dan putih tidak otomatis memperbaiki bentuk yang salah. Jika sebagian daun dan sebagian latar sudah tercampur dalam kelompok intensitas yang sama, invers hanya menukar pilihan keduanya.

2.4 Adaptive: ambang dari lingkungan sekitar piksel

Adaptive tidak menggunakan satu batas yang sama untuk seluruh foto. Ambangnya ditentukan dari lingkungan di sekitar setiap piksel. Cara ini dapat membantu ketika satu bagian gambar terang dan bagian lain lebih gelap karena bayangan.

Contoh memakai `ADAPTIVE_THRESH_GAUSSIAN_C`. Nilai Gaussian lokal dikurangi konstanta `C` untuk membentuk ambang. `blockSize = 31` berarti lingkungan berukuran 31 × 31 piksel, bukan ukuran gambar. BlockSize harus ganjil dan lebih dari satu. Nilai awal `C = 5` dapat dicoba setelah kita memahami pengaruhnya.

Adaptive juga dapat menangkap serat daun atau tekstur latar. Hasilnya mungkin berupa fragmen atau mask berongga, bukan bentuk daun yang penuh. Mampu menemukan perubahan intensitas lokal belum berarti berhasil mengambil seluruh objek.

2.5 Otsu: ambang global yang dihitung dari histogram

Histogram menunjukkan banyaknya piksel pada setiap tingkat keabuan. Otsu mencari pemisahan yang meminimalkan variasi di dalam dua kelompok intensitas, lalu mengembalikan nilai ambang yang dihitung. Di kode, nilainya diterima sebagai `T_otsu`.

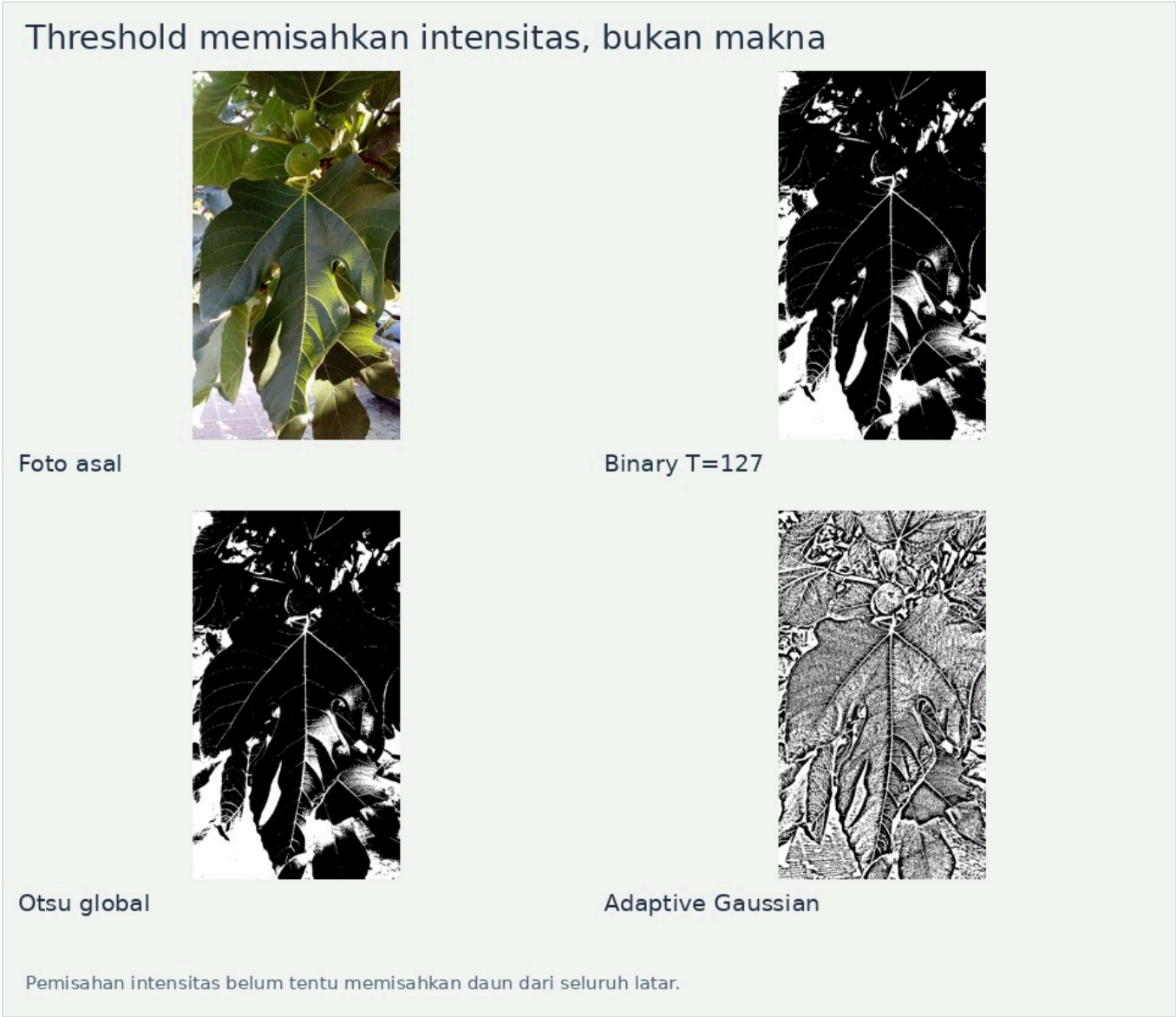
Otsu bukan cara lain untuk selalu memperoleh angka 127. Nilainya mengikuti gambar yang diproses. Dua kelompok intensitas pada histogram juga belum tentu sama dengan dua kategori yang kita inginkan. Daun dan bayangan latar dapat berada dalam kelompok gelap yang sama.

Walaupun Otsu tetap mengeluarkan angka ketika histogram sulit dipisahkan, angka itu bukan jaminan hasil akurat. Karena ambangnya global, perubahan pencahayaan lokal juga tidak otomatis teratasi.

Dasar membandingkan metode

Metode	Cara menentukan ambang	Hal yang perlu diperiksa

Metode	Cara menentukan ambang	Hal yang perlu diperiksa
Binary global	Satu T ditentukan pengguna.	Apakah objek dan latar cukup berbeda intensitasnya?
Adaptive Gaussian	Nilai dari lingkungan lokal, dikurangi C.	Apakah tekstur ikut terpilih atau bentuk objek menjadi berongga?
Otsu	Satu T dihitung dari histogram.	Apakah dua kelompok intensitas sesuai dengan objek yang ingin diambil?



Pemisahan intensitas belum tentu memisahkan daun dari seluruh latar.

Panel di atas merupakan ilustrasi pada materi sumber. Baca hasilnya bersama foto, terutama pada tepi, tangkai, dan tekstur latar. Panel ini tidak digunakan untuk menyatakan persentase

akurasi atau menjamin bahwa satu metode selalu paling baik.

2.6 Pakai mask, lalu lihat lokasi kesalahannya

Mask pada contoh adalah `uint8` satu channel dengan tinggi dan lebar yang sama seperti gambar. Citra warna mempunyai tiga channel, tetapi mask cukup satu channel. Mask dipakai oleh `bitwise_and` untuk mempertahankan piksel terpilih; invers mask dipakai untuk mengambil background.

Bagian hitam pada hasil ekstraksi berarti tidak ditampilkan oleh mask, bukan kategori baru. Gunakan **overlay**, yaitu penanda warna di atas gambar asli, agar batas pilihan lebih mudah diperiksa.

Opening dapat membantu menghapus bintik kecil dan *closing* dapat menutup celah kecil. Keduanya bersifat opsional. Tangkai yang tipis juga bisa hilang setelah perapihan, sehingga hasil yang lebih bersih belum tentu lebih benar.

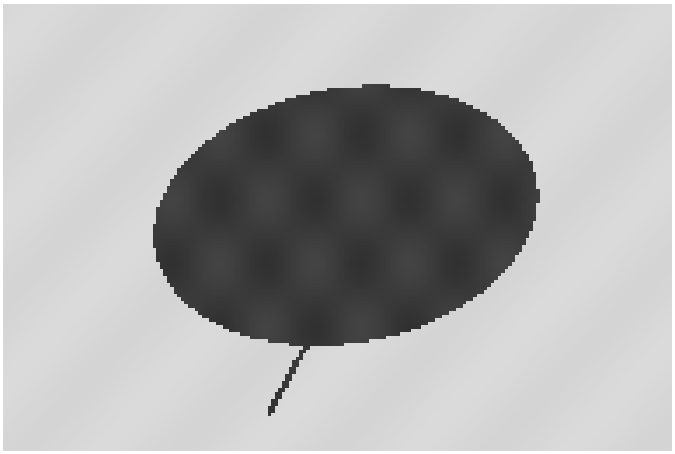
> **Penilaian yang jujur:** tanpa anotasi acuan atau *ground truth*, jelaskan kesalahan secara visual dan kualitatif. Sebutkan bagian yang hilang atau latar yang ikut terambil. Jangan mengubah kesan “terlihat bagus” menjadi angka akurasi.

---

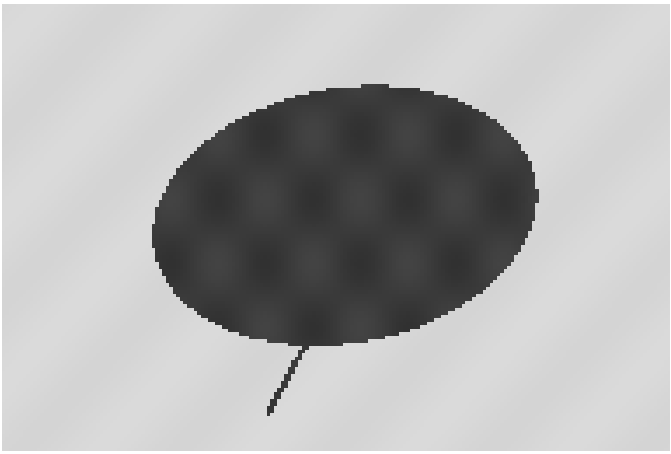
3. Coba ilustrasi sebelum praktikum

Pakai gambar daun buatan atau **pilih foto dari HP/komputer**. Ubah ambang, polaritas, dan metode untuk melihat perubahan mask dan overlay. Gambar contoh adalah ilustrasi sintetis; foto pilihan Anda dipakai sebagai input nyata, tetapi hasilnya tetap bukan pengukuran akurasi.

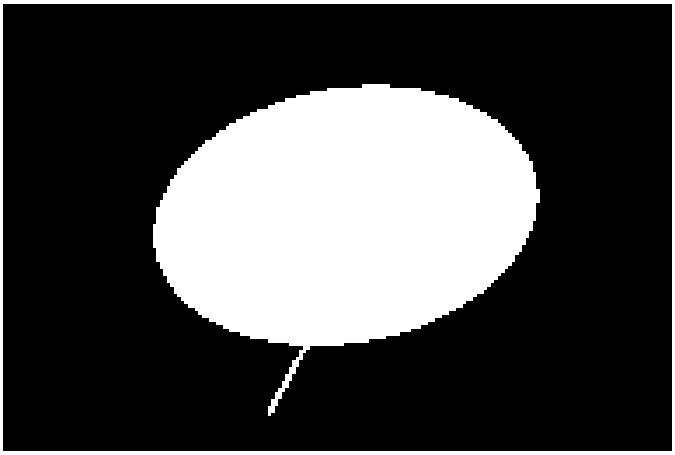
> Foto dibaca dan diproses **hanya di browser**, tidak dikirim atau disimpan ke server. Simulasi memakai JavaScript, bukan menjalankan OpenCV/Colab. Foto warna ditampilkan sebagai input asli dan diubah ke grayscale untuk ketiga metode, tanpa blur awal. Praktikum Colab memakai Gaussian blur sehingga hasilnya dapat berbeda.



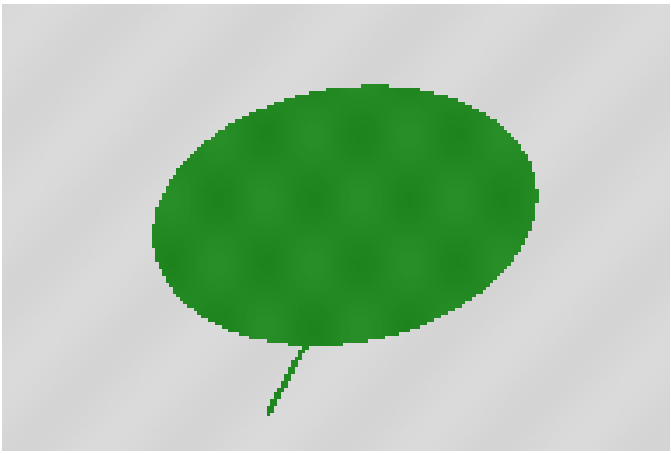
Input · gambar contoh sintetis



Grayscale · input untuk threshold



Mask · putih berarti dipilih



Overlay pilihan pada citra

Binary global · T=127 · inverse, pilih lebih gelap · contoh pencahayaan merata.

6.419 dari 24.576 piksel dipilih. Ini jumlah pilihan mask, bukan persentase ketepatan.

Dengan inverse, intensitas kurang dari atau sama dengan 127 menjadi putih. Geser T untuk melihat perubahan pilihan.

Dalam PDF/cetakan, ilustrasi ini adalah cuplikan statis. Untuk mengubah parameter, buka versi web materi.

Cara membacanya: periksa daun, latar, dan tangkai. Jika warna putih meliputi latar, tinjau polaritas. Jika bentuk daun berlubang, jangan langsung menganggap hasilnya lebih baik hanya karena detailnya lebih banyak. Jumlah piksel terpilih bukan ukuran akurasi.

4. Video pembelajaran

OpenCV Python Image Thresholding — Kevin Wood | Robotics & AI.

Fokus saat menonton: Amati bagaimana threshold membuat mask biner, bagaimana nilai ambang dipilih dan kapan latar tidak seragam menjadi masalah.

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 — Image Thresholding
- OpenCV — Morphological Transformations
- OpenCV — Basic Operations on Images

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

Sekarang lakukan alur yang sama menggunakan foto Anda. Buat notebook baru melalui Google Colab, beri judul yang mudah dikenali, lalu jalankan sel berikut secara berurutan dengan `Shift` + `Enter`.

> Gunakan satu sel kode untuk setiap langkah. Sesudah hasil awal muncul, ubah satu parameter, jalankan kembali bagian yang terkait, dan bandingkan perubahan pada gambar. Contohnya, coba T=80, lalu 127, lalu 170 dengan foto dan pengaturan lain yang sama. Angka tersebut hanya bahan pengamatan, bukan nilai yang pasti cocok.

5.1 Siapkan library

Import menyiapkan OpenCV, NumPy, tampilan gambar, dan fasilitas upload Colab. Jalankan sel ini sekali di awal notebook.

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

5.2 Unggah satu gambar

Pilih tepat satu JPEG atau PNG yang berizin. Kode memeriksa file, ukuran, tipe data, dan channel. Gambar grayscale juga dapat dipakai; kode menyiapkannya sebagai BGR. Gunakan gambar tanpa transparansi.

```
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()
```

5.3 Buat tiga mask dari input yang sama

Mulai dengan T=127, BLOCK_SIZE=31, dan C=5. Pilih OBJEK_GELAP=True untuk objek yang relatif gelap, atau False bila objek lebih terang. Variabel parameter diletakkan di awal agar mudah diubah satu per satu.

```
gray = cv2.cvtColor(bgr, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (5, 5), 0)
OBJEK_GELAP = True
T = 127
BLOCK_SIZE = 31
C = 5
if not 0 <= T <= 255:
    raise ValueError("T harus berada di antara 0 dan 255.")
if BLOCK_SIZE <= 1 or BLOCK_SIZE % 2 == 0:
    raise ValueError("BLOCK_SIZE harus ganjil dan lebih dari 1.")
mode = cv2.THRESH_BINARY_INV if OBJEK_GELAP else cv2.THRESH_BINARY
_, binary = cv2.threshold(blur, T, 255, mode)
adaptive = cv2.adaptiveThreshold(
    blur, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, mode, BLOCK_SIZE, C)
T_otsu, otsu = cv2.threshold(blur, 0, 255, mode | cv2.THRESH_OTSU)
masks = {"Binary": binary, "Adaptive": adaptive, "Otsu": otsu}
print("Objek gelap:", OBJEK_GELAP, "T global:", T)
print("BLOCK_SIZE:", BLOCK_SIZE, "C:", C, "T Otsu:", T_otsu)
for metode, mask in masks.items():
    print(metode, "piksel terpilih:", int(np.count_nonzero(mask)))
```

5.4 Bandingkan foto, histogram, dan mask

Lihat lokasi yang sama pada foto dan setiap mask. Garis histogram menampilkan T yang Anda tetapkan serta nilai Otsu yang benar-benar dihitung. Banyaknya piksel putih bukan ukuran akurasi.

```
fig, axes = plt.subplots(2, 3, figsize=(15, 9))
axes[0, 0].imshow(cv2.cvtColor(bgr, cv2.COLOR_BGR2RGB)); axes[0,
0].set_title("Foto sumber")
axes[0, 1].imshow(blur, cmap="gray", vmin=0, vmax=255); axes[0,
1].set_title("Grayscale + blur")
hist = cv2.calcHist([blur], [0], None, [256], [0, 256]).ravel()
axes[0, 2].plot(np.arange(256), hist)
axes[0, 2].axvline(T, color="red", label=f"T tetap={T}")
```

```
axes[0, 2].axvline(T_otsu, color="green", label=f"T Otsu=
{T_otsu:.0f}")
axes[0, 2].set_xlabel("Intensitas"); axes[0, 2].set_ylabel("Jumlah
pixel")
axes[0, 2].legend(); axes[0, 2].set_title("Histogram input
threshold")
for ax, (metode, mask) in zip(axes[1], masks.items()):
    ax.imshow(mask, cmap="gray", vmin=0, vmax=255);
ax.set_title(metode); ax.axis("off")
axes[0, 0].axis("off"); axes[0, 1].axis("off")
plt.tight_layout(); plt.show()
```

5.5 Ambil objek, background, dan overlay

PILIH dapat diisi Binary, Adaptive, atau Otsu. Mulai dengan RAPIKAN=False agar mask akhir sama dengan mask awal. Ubah ke True hanya jika opening/closing membantu; bandingkan tangkai, ujung, dan lubang sebelum menerima hasilnya.

```
PILIH = "Otsu"
RAPIKAN = False
if PILIH not in masks:
    raise ValueError("PILIH harus Binary, Adaptive, atau Otsu.")
mask_awal = masks[PILIH].copy()
mask_bersih = mask_awal.copy()
if RAPIKAN:
    kernel = np.ones((3, 3), dtype=np.uint8)
    mask_bersih = cv2.morphologyEx(mask_bersih, cv2.MORPH_OPEN,
kernel)
    mask_bersih = cv2.morphologyEx(mask_bersih, cv2.MORPH_CLOSE,
kernel)
foreground = cv2.bitwise_and(bgr, bgr, mask=mask_bersih)
background = cv2.bitwise_and(bgr, bgr,
                             mask=cv2.bitwise_not(mask_bersih))
overlay = bgr.copy()
warna = np.empty_like(bgr); warna[:] = (0, 255, 0)
campuran = cv2.addWeighted(bgr, 0.6, warna, 0.4, 0)
overlay[mask_bersih > 0] = campuran[mask_bersih > 0]
jumlah = int(np.count_nonzero(mask_bersih))
print("Metode:", PILIH, "Morfologi aktif:", RAPIKAN)
print("Piksel terpilih:", jumlah, "dari", mask_bersih.size)
if jumlah == 0 or jumlah == mask_bersih.size:
```

```
    print("Mask kosong/penuh; periksa polaritas, foto, dan
parameter.")
fig, axes = plt.subplots(1, 5, figsize=(20, 5))
for ax, judul, img in zip(axes,
        ["Mask awal", "Mask akhir", "Objek", "Background",
"Overlay"],
        [mask_awal, mask_bersih, foreground, background, overlay]):
    if img.ndim == 2:
        ax.imshow(img, cmap="gray", vmin=0, vmax=255)
    else:
        ax.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    ax.set_title(judul); ax.axis("off")
plt.tight_layout(); plt.show()
```

Membaca hasil latihan dan mengatasi kendala

Latar justru menjadi bagian yang terpilih.

Periksa OBJEK_GELAP dan arah binary/inverse. Putih pada mask harus sesuai area yang ingin dipertahankan.

Adaptive menolak ukuran blok.

Gunakan BLOCK_SIZE yang ganjil dan lebih dari satu; input contoh harus grayscale uint8.

Mask adaptive penuh garis atau lubang.

Bandingkan BLOCK_SIZE dan C satu per satu. Tekstur yang terdeteksi bukan jaminan seluruh bentuk objek berhasil diambil.

Tangkai atau ujung hilang setelah morfologi.

Kembalikan RAPIKAN=False dan periksa mask awal. Perapihan tidak wajib, terutama jika mengubah detail yang diperlukan.

Variabel tidak dikenali setelah runtime direset.

Jalankan lagi sel dari awal. File upload dan variabel runtime Colab bersifat sementara.

> Simpan notebook dan hasil yang penting ke Drive sebelum sesi berakhir. Gunakan media berizin dan jangan unggah data sensitif tanpa izin. Simulasi browser membantu memahami konsep; hasil pada foto Anda tetap perlu diperiksa sendiri.

Pengolahan Citra · Pendahuluan dan daftar seluruh materi