PENGOLAHAN CITRA · MATERI 02

Operasi Dasar Citra dengan OpenCV

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

Anda mempunyai satu foto tanaman untuk dibaca di layar ponsel, ditampilkan pada halaman web, dan dipakai mengamati satu helai daun. Ketiga kebutuhan itu tidak selalu memerlukan perubahan yang sama. Untuk halaman web, foto mungkin cukup diperkecil. Untuk mengamati daun tertentu, area di sekelilingnya mungkin perlu dipotong. Jika posisi kamera miring, yang dibutuhkan justru perubahan orientasi. Memilih operasi dimulai dari masalah yang ingin diselesaikan, bukan dari efek yang paling menarik.

Enam operasi dasar pada materi ini adalah resize, crop, rotate, flip, brightness, dan contrast. Empat operasi pertama berhubungan dengan ukuran, area, atau susunan posisi piksel. Dua operasi terakhir mengubah nilai intensitasnya. Semuanya bisa menghasilkan gambar yang tampak berbeda, tetapi jenis informasi yang berubah tidak sama. Foto yang diperkecil masih dapat memperlihatkan seluruh tanaman; foto yang dipotong mungkin tidak lagi memperlihatkan batang atau latarnya.

Perubahan juga memiliki batas. Memperbesar foto tidak otomatis menambah detail daun. Mencerahkan bagian yang sudah sangat terang dapat menghapus perbedaan halus di permukaannya. Mencerminkan gambar mungkin masuk akal untuk membandingkan bentuk, tetapi menjadi masalah ketika ada tulisan yang harus tetap terbaca. Karena itu, hasil pengolahan perlu dibaca bersama tujuan dan parameter yang digunakan.

Praktik di Google Colab akan memakai satu gambar sebagai sumber bagi keenam operasi secara terpisah. Dengan titik awal yang sama, Anda dapat membandingkan pengaruh setiap parameter tanpa tercampur perubahan sebelumnya. Alurnya berakhir pada penyimpanan gambar dan parameter dalam ZIP, sehingga hasil yang tampak di notebook dapat diperiksa kembali sebagai berkas. Konsep berikut membantu menghubungkan setiap bagian kode dengan perubahan yang benar-benar terlihat.

2. Pahami konsep secara berurutan

2.1 Bekal membaca perubahan citra

Citra merupakan kisi piksel; citra warna pada kode dibaca sebagai array tiga kanal BGR. `img.shape` berurutan tinggi, lebar, kanal, sedangkan ukuran tujuan OpenCV ditulis lebar, tinggi. Akses sebuah lokasi menggunakan `img[y, x]`. Saat ditampilkan dengan Matplotlib, BGR dikonversi ke RGB agar warna ditafsirkan dengan benar. Pembacaan melalui `cv2.IMREAD_COLOR` menyediakan representasi warna tiga kanal untuk seluruh operasi.

Operasi geometrik mengubah penempatan atau cakupan spasial piksel, sedangkan operasi intensitas mengubah nilai komponennya. Fungsi `enam_transformasi()` selalu menggunakan `img` sebagai sumber, bukan menumpuk hasil operasi sebelumnya. Jadi, panel brightness bukan hasil resize yang kemudian dicerahkan. Pemisahan ini membuat sebab perubahan lebih mudah dikenali dan menjaga perbandingan tetap jelas.

2.2 Resize — mengubah dimensi

Resize mengubah jumlah baris dan kolom untuk mewakili area gambar. Parameter `skala` diterapkan pada lebar dan tinggi. Faktor yang sama menjaga aspect ratio, yaitu perbandingan lebar terhadap tinggi, sedekat mungkin. Namun, ukuran akhir harus bilangan bulat. Kode memakai `round()`, sehingga pembulatan bisa menimbulkan selisih kecil rasio, terutama pada gambar kecil atau dimensi ganjil. Pembatas `max(1, ...)` mencegah dimensi menjadi nol.

Piksel keluaran dihitung melalui interpolasi, yaitu cara menaksir nilai pada kisi baru dari piksel sumber. Kode memilih `INTER_AREA` saat memperkecil dan `INTER_LINEAR` untuk skala lainnya. Memperkecil dapat menghilangkan detail halus; memperbesar hanya menaksir sampel tambahan, bukan menemukan kembali informasi yang tidak terekam. Resize berbeda dari kompresi file: dimensi piksel dan cara menyimpan berkas adalah dua hal terpisah.



Foto asal daun dan buah tin · 500 × 887 piksel



Resize — skala 0,5 · 250 × 444 piksel

2.3 Crop — memilih bagian gambar

Crop mempertahankan area tertentu, disebut region of interest atau ROI, dan membuang area di luarnya. Jika resize membuat seluruh tanaman tampil lebih kecil, crop dapat

mempertahankan hanya satu daun pada ukuran sampel aslinya. Bentuk objek di dalam potongan tidak diregangkan. Namun, konteks di luar batas hilang; potongan yang terlalu sempit bisa membuang bagian objek yang justru penting.

Praktikum memilih potongan tengah melalui `crop_fraction`, yaitu fraksi lebar sekaligus fraksi tinggi yang dipertahankan, bukan persentase luas. Ukuran potongan dihitung dengan `int()` dan dibatasi minimal satu piksel. Slicing `img[y1:y1 + ch, x1:x1 + cw]` memakai batas akhir yang tidak disertakan. `.copy()` membuat array terpisah agar perubahan potongan tidak ikut mengubah citra sumber. ROI galeri dapat berbeda dari potongan tengah kode karena dipilih untuk kebutuhan ilustrasi.



Foto asal daun dan buah tin · 500 × 887 piksel



Crop — ROI area daun · 370 × 533 piksel

2.4 Rotate — memutar orientasi

Rotate memutar gambar terhadap suatu pusat. `cv2.getRotationMatrix2D()` membentuk matriks pemetaan koordinat, lalu `cv2.warpAffine()` menerapkannya. Pada kode ini, `sudut` positif berarti berlawanan arah jarum jam dan pusat berada di tengah kisi. Faktor skala rotasinya tetap 1, sehingga parameter ini dipakai untuk memutar, bukan sekaligus memperbesar objek.

Ukuran kanvas keluaran tetap `(w, h)`. Akibatnya, sebagian tepi gambar bisa keluar dari kanvas dan terpotong, sementara posisi yang tidak memiliki sumber diisi hitam oleh `BORDER_CONSTANT` dan `borderValue=(0, 0, 0)`. Hitam itu bukan bagian objek baru. Mempertahankan seluruh gambar memerlukan perluasan kanvas dan penyesuaian translasi. Rotasi juga tidak dengan sendirinya memperbaiki distorsi perspektif, seperti dokumen yang difoto dari samping.



Foto asal daun dan buah tin · 500 × 887 piksel



Rotate — 30° berlawanan arah jarum jam · 500 × 887 piksel

2.5 Flip — membuat pencerminan

Flip menukar susunan posisi piksel secara teratur tanpa mengubah dimensi. `flip_code=1` mencerminkan kiri–kanan, `0` mencerminkan atas–bawah, dan `-1` melakukan keduanya.

Berbeda dari rotasi dengan sudut bebas, pencerminan ini tidak membutuhkan taksiran nilai piksel baru: nilai sumber hanya berpindah tempat.

Pada foto daun, flip dapat membantu melihat bentuk dari orientasi cermin. Pada foto dengan tulisan, huruf ikut tercermin dan mungkin tidak layak dipakai untuk pembacaan teks. Jadi, perubahan yang secara teknis berhasil belum tentu sesuai tujuan. Perhatikan apakah arah, tulisan, atau label objek memiliki makna yang harus dipertahankan.



Foto asal daun dan buah tin · 500 × 887 piksel



Flip — cermin horizontal · 500 × 887 piksel

2.6 Brightness — menggeser intensitas

Brightness pada kode berarti menambahkan offset `beta` ke setiap komponen. Beta positif membuat komponen lebih tinggi; beta negatif menurunkannya. Perhitungan dilakukan setelah `img.astype(np.float32)`, lalu `np.clip(..., 0, 255)` membatasi hasil sebelum dikembalikan menjadi uint8. Langkah ini menghindari aritmetika langsung pada uint8 yang dapat melingkar ketika melewati rentang.

Clipping menahan nilai di bawah nol menjadi nol dan nilai di atas 255 menjadi 255. Akibatnya, beberapa nilai berbeda dapat menjadi sama sehingga detail hilang. Menaikkan brightness tidak memulihkan area yang sejak awal sudah jenuh. `convertScaleAbs()` bukan pengganti identik untuk semua kebutuhan ini karena mengambil nilai absolut; nilai negatif bisa berubah positif, bukan sekadar dibatasi ke nol.



Foto asal daun dan buah tin · 500 × 887 piksel



Brightness — offset β = +35 · 500 × 887 piksel

2.7 Contrast — mengalikan intensitas

Contrast pada praktikum memakai gain `alpha`: setiap komponen dikalikan alpha, kemudian dibatasi ke rentang uint8. Sebelum clipping, alpha lebih dari satu memperbesar selisih

antarnilai, sedangkan alpha antara nol dan satu memperkecilnya. Karena pengalian berpusat pada nol, gain juga dapat mengubah tingkat terang keseluruhan; ini bukan pengaturan kontras yang mempertahankan suatu nilai tengah.

Brightness dan contrast dapat dirangkum sebagai `output = clip(alpha × input + beta, 0, 255)`. Namun, kode memisahkan keduanya: brightness hanya menambah beta, contrast hanya mengalikan alpha. Saat gain terlalu besar, area terang dapat menjadi seragam akibat saturasi. Bandingkan perbedaan detail, bukan hanya kesan bahwa hasil lebih mencolok.



Foto asal daun dan buah tin · 500 × 887 piksel



Contrast — gain α = 1,4 · 500 × 887 piksel

2.8 Menyimpan hasil dan parameter

`cv2.imwrite()` menyimpan array BGR sebagai PNG dan nilai baliknya diperiksa untuk mendeteksi kegagalan. PNG tidak menambahkan kompresi lossy, tetapi juga tidak mengembalikan detail yang sudah hilang akibat crop atau clipping. Sel akhir mengemas citra sumber, enam hasil, dan `parameter.json` dalam ZIP. Direktori baru dibuat setiap kali sel dijalankan agar hasil sebelumnya tidak tertimpa. Karena file runtime Colab bersifat sementara, unduh arsip yang ingin dipertahankan sebelum sesi berakhir.

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

---

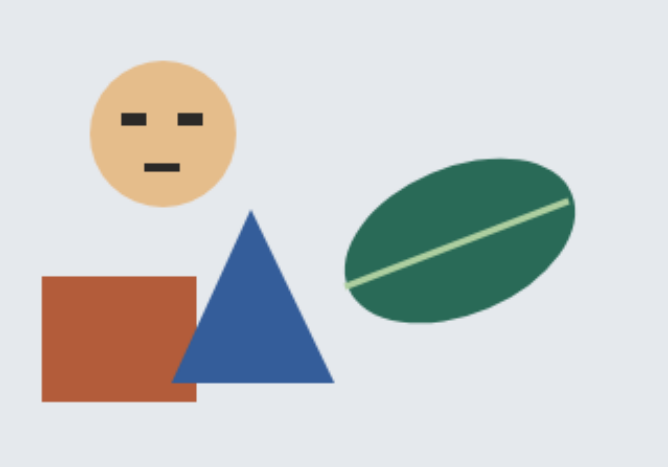
3. Ilustrasi interaktif sebelum praktikum

Enam operasi citra pada piksel nyata

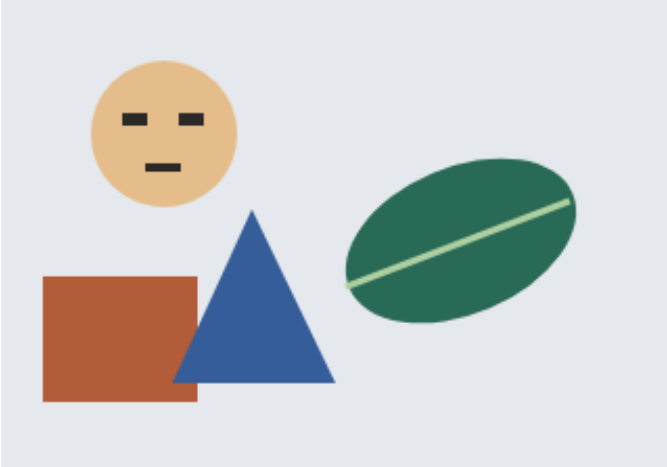
Resize, crop, rotasi, flip, brightness, dan contrast bekerja langsung pada buffer RGBA. Level bersifat kontekstual sesuai operasi (lihat rincian). Sudut hanya untuk rotasi, sumbu hanya untuk flip. Sampling nearest-neighbor, bukan klaim identik OpenCV. Resize dan crop mengembalikan dimensi hasil yang sebenarnya.

Enam operasi benar-benar mengubah piksel atau ukuran hasil. Ini adalah implementasi ilustratif JavaScript, bukan runtime OpenCV. Interpolasi dan pembulatan dapat berbeda dari Colab; perhatikan angka dimensi, bukan hanya ukuran tampilan kotak.

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: Operasi = resize · Level: resize %, crop % (maks.100), brightness 100=0, contrast 100=1× = 100 · Sudut rotasi (°; positif CCW) = 30 · Sumbu flip (khusus flip) = horizontal

resize: 320 × 224 → 320 × 224; 71680 piksel, 286720 byte RGBA.

Resize: level 100% memberi skala seragam 1× untuk kedua sumbu; dimensi dibulatkan, minimal 1 piksel. Rasio aspek dipertahankan sejauh pembulatan memungkinkan. Nearest-neighbor memakai floor((x+0.5)×Wmasuk/Wkeluar), dan analog untuk y.

Semua hasil adalah komputasi piksel deterministik. Buffer sumber tidak diubah. Nearest-neighbor dapat bergerigi; tidak ada pembandingan OpenCV palsu. Dimensi keluaran harus dipakai untuk kanvas dan ekspor PNG.

▼ **Angka dan parameter hasil perhitungan ilustrasi**


```
{
  "operation": "resize",
  "inputWidth": 320,
```

```
  "inputHeight": 224,
  "outputWidth": 320,
  "outputHeight": 224,
  "outputRGBABytes": 286720,
  "level": 100,
  "angle": null,
  "axis": null,
  "brightnessOffset": null,
  "contrastGain": null,
  "cropLeft": null,
  "cropTop": null,
  "sampling": "nearest-neighbor"
}
```


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 - Crop, Resize, Flip, Rotate, Translate (Affine Transformations) — DIY Engineers. Video berbahasa Inggris; fokuskan pengamatan pada crop, resize, flip dan rotate. Bagian translate merupakan pengayaan. Jika video membahas setup lokal, tetap ikuti praktikum Google Colab pada halaman ini.

Buka video di YouTube jika pemutar tidak tersedia

Pemutar disediakan oleh YouTube. Saat video dimuat/diputar, koneksi melibatkan layanan pihak ketiga; ketersediaan pemutaran mengikuti kebijakan pemilik video dan jaringan.

▼ **Referensi dan atribusi gambar**

1. Google Colab — FAQ resmi: notebook, runtime dan berbagi.
2. Google Colab — files.upload() dan files.download().
3. OpenCV — Geometric Transformations of Images.
4. OpenCV — Basic Operations on Images.
5. OpenCV — Changing the Contrast and Brightness of an Image.

Foto: Ficus carica - Fig leaf oleh Ανώνυμος Βικιπαιδιστής · CC BY 3.0. Versi galeri diubah ukurannya, dipotong, diputar, dicerminkan atau disesuaikan intensitasnya untuk pengajaran;

bukan bukti kondisi kesehatan tanaman. Hak penggunaan ulang foto mengikuti lisensi sumber. Tautan sumber tetap tersedia terlepas dari pembatasan copy pada halaman.

---

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.

Siapkan satu gambar berizin, kemudian jalankan lima sel di Google Colab menurut urutan import → upload → parameter/transformasi → visualisasi → penyimpanan dan download. Sel 2 meminta tepat satu gambar melalui `files.upload()`, membacanya menjadi `img`, dan memeriksa dimensi minimal 2 × 2 piksel. Sel 3 menyediakan `skala`, `sudut`, `crop_fraction`, `flip_code`, `beta`, dan `alpha`. Mulai dari nilai bawaan untuk mengenali hasil masing-masing operasi; keenam hasil dihitung secara terpisah dari gambar asli.

Sel 4 menyusun citra asal dan enam hasil dari `hasil` ke dalam `panels`. Perhatikan dimensi pada judul, area yang hilang pada crop, tepi hitam pada rotate, serta detail yang menyatu pada brightness atau contrast. Jika mencoba parameter baru, ubah Sel 3 lalu jalankan kembali Sel 3, 4, dan 5; Sel 4 perlu diulang agar `panels` tidak masih berisi hasil lama. Ikuti batas parameter dalam fungsi, termasuk skala 0.1–2, sudut -180–180, fraksi crop lebih dari 0 sampai 1, serta pilihan flip -1, 0, atau 1.

Sel 5 menyimpan `00_asli.png`, keenam hasil PNG, dan `parameter.json` dalam direktori baru, kemudian mengunduh ZIP melalui `files.download()`. Pertahankan ZIP sebagai salinan hasil dan konfigurasi percobaan. Jika unduhan tidak muncul, periksa izin browser atau unduh arsip melalui panel Files Colab. Jika runtime direset dan variabel hilang, jalankan kembali seluruh urutan mulai import dan upload; file sementara di runtime bukan penyimpanan permanen.

Sel 1 — Import library

Jalankan sekali saat memulai runtime.

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

```
import zipfile

print("OpenCV:", cv2.__version__)
print("NumPy:", np.__version__)
```

Sel 2 — Upload satu gambar

Pilih tepat satu gambar melalui dialog upload.

```
# Upload satu gambar. Penggunaan data pribadi/sensitif mengikuti
izin yang berlaku.
uploaded = files.upload()
if len(uploaded) != 1:
    raise ValueError("Pilih tepat satu gambar, lalu jalankan ulang
sel upload.")
filename, raw_bytes = next(iter(uploaded.items()))
img = cv2.imdecode(np.frombuffer(raw_bytes, dtype=np.uint8),
cv2.IMREAD_COLOR)
if img is None:
    raise ValueError("File tidak dapat dibaca sebagai gambar.")
h, w = img.shape[:2]
if min(h, w) < 2:
    raise ValueError("Gunakan gambar minimal 2 x 2 piksel.")
print("Nama:", filename, "shape:", img.shape, "dtype:", img.dtype)
```

Sel 3 — Parameter dan enam transformasi

Ubah parameter di awal blok ini, lalu jalankan kembali untuk eksperimen baru.

```
# Parameter: ubah nilainya dan jalankan sel ini kembali.
skala = 0.5
sudut = 30.0
crop_fraction = 0.5
flip_code = 1
beta = 35.0
alpha = 1.4

def enam_transformasi(img, skala, sudut, crop_fraction, flip_code,
beta, alpha):
```

```
    values = [skala, sudut, crop_fraction, beta, alpha]
    if not all(np.isfinite(value) for value in values):
        raise ValueError("Semua parameter harus berupa angka
finite.")
    if not 0.1 <= skala <= 2:
        raise ValueError("Skala harus 0.1 sampai 2.")
    if not -180 <= sudut <= 180:
        raise ValueError("Sudut harus -180 sampai 180.")
    if not 0 < crop_fraction <= 1:
        raise ValueError("Crop fraction harus > 0 dan <= 1.")
    if flip_code not in (-1, 0, 1):
        raise ValueError("Flip harus -1, 0 atau 1.")
    if not -255 <= beta <= 255 or not 0 < alpha <= 3:
        raise ValueError("Beta: -255..255; alpha: >0 sampai 3.")
    h, w = img.shape[:2]
    cw, ch = max(1, int(w * crop_fraction)), max(1, int(h *
crop_fraction))
    x1, y1 = (w - cw) // 2, (h - ch) // 2
    M = cv2.getRotationMatrix2D(((w - 1) / 2.0, (h - 1) / 2.0),
sudut, 1.0)
    pixels = img.astype(np.float32)
    return {
        "01_resize": cv2.resize(img, (max(1, round(w * skala)),
max(1, round(h * skala))),
                                interpolation=cv2.INTER_AREA if
skala < 1 else cv2.INTER_LINEAR),
        "02_crop": img[y1:y1 + ch, x1:x1 + cw].copy(),
        "03_rotate": cv2.warpAffine(img, M, (w, h),
borderMode=cv2.BORDER_CONSTANT,
                                    borderValue=(0, 0, 0)),
        "04_flip": cv2.flip(img, flip_code),
        "05_brightness": np.clip(pixels + beta, 0,
255).astype(np.uint8),
        "06_contrast": np.clip(pixels * alpha, 0,
255).astype(np.uint8),
    }

hasil = enam_transformasi(img, skala, sudut, crop_fraction,
flip_code, beta, alpha)
print("Jumlah transformasi:", len(hasil))
```

Sel 4 — Tampilkan perbandingan hasil

Jalankan setelah Sel 3 untuk melihat citra asal dan enam hasil.

```
# Tampilkan citra asal dan seluruh hasil; Matplotlib menggunakan
RGB.
fig, axes = plt.subplots(2, 4, figsize=(16, 8))
panels = {"00_asli": img, **hasil}
for ax, (name, image) in zip(axes.flat, panels.items()):
    ax.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    ax.set_title(f"{name}\n{image.shape[1]} x {image.shape[0]}")
    ax.axis("off")
for ax in axes.flat[len(panels):]:
    ax.axis("off")
plt.tight_layout()
plt.show()
```

Sel 5 — Simpan dan unduh ZIP

Jalankan setelah Sel 4. ZIP berisi gambar asal, enam hasil dan parameter.

```
# Gunakan direktori baru setiap run; hasil eksperimen lain tidak
tertimpa.
import tempfile
output_dir = Path(tempfile.mkdtemp(prefix="materi02_",
dir="/content"))
for name, image in panels.items():
    if not cv2.imwrite(str(output_dir / (name + ".png")), image):
        raise OSError("Gagal menyimpan " + name)
params = {"skala": skala, "sudut": sudut, "crop_fraction":
crop_fraction,
          "flip_code": flip_code, "beta": beta, "alpha": alpha,
          "input_filename": filename, "input_shape":
list(img.shape),
          "opencv_version": cv2.__version__, "numpy_version":
np.__version__}
(output_dir / "parameter.json").write_text(json.dumps(params,
indent=2), encoding="utf-8")
zip_path = output_dir.with_suffix(".zip")
with zipfile.ZipFile(zip_path, "w",
```

```
compression=zipfile.ZIP_DEFLATED) as archive:
    for path in sorted(output_dir.iterdir()):
        archive.write(path, arcname=path.name)
print("ZIP siap:", zip_path.name)
files.download(str(zip_path))
```

Setiap transformasi memakai **citra asli**, bukan hasil operasi sebelumnya. File `00_asli.png`, enam hasil dan `parameter.json` dikemas dalam ZIP. Direktori baru dibuat untuk setiap run agar hasil tidak tertimpa.

Jika terjadi error

- **NameError:** jalankan sel sebelumnya; variabel hilang setelah reset runtime.
- **Upload kosong atau lebih dari satu gambar:** ulangi sel upload dan pilih tepat satu file.
- **Gambar tidak terbaca:** pilih file gambar valid, bukan dokumen yang hanya berganti ekstensi.
- **Warna salah:** pastikan BGR dikonversi ke RGB sebelum `plt.imshow()`.
- **ZIP tidak terunduh:** cek izin download browser dan ulangi sel akhir; alternatif klik kanan ZIP pada panel Files Colab lalu Download.
- **Parameter ditolak:** sesuaikan batas dalam fungsi; pertahankan pesan error sebagai bukti pengujian negatif.