PENGOLAHAN CITRA · MATERI 01

Pengenalan Pengolahan Citra & Setup Python

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 Anda memotret sehelai daun dengan telepon, lalu membuka foto itu di layar laptop. Dari jauh, yang terlihat adalah bentuk daun, urat, dan bayangannya. Ketika diperbesar cukup jauh, permukaannya tampak tersusun dari kotak-kotak kecil. Kotak tersebut membantu membayangkan piksel: satu sampel gambar pada sebuah lokasi. Komputer tidak langsung membaca sampel itu sebagai ‘daun’. Ia lebih dahulu menyimpan dan mengolah angka yang mewakili intensitas atau komponen warnanya.

Angka-angka inilah titik berangkat pengolahan citra. Sebuah program dapat membaca foto, memeriksa dimensinya, mengubah tampilannya, atau mengambil nilai pada lokasi tertentu. Namun, sebelum melakukan perubahan, kita perlu mengenali data yang masuk. Foto berwarna dan gambar abu-abu tidak selalu memiliki susunan array yang sama. Dua berkas dengan dimensi sama juga belum tentu memiliki ukuran file yang sama. Menyamakan semua istilah itu dengan ‘besar gambar’ akan membuat pembacaan hasil membingungkan.

Pada materi ini, Anda akan mengikuti perjalanan tiga gambar: dari berkas yang diunggah, menjadi array atau susunan angka dalam Python, lalu menjadi gambar yang ditampilkan kembali. Anda akan mengenali tinggi, lebar, jumlah piksel, kanal warna, dan nilai piksel. Contoh sehari-hari tersebut cukup untuk memahami dasar kerja program tanpa perlu model kecerdasan buatan atau rumus yang rumit.

Lingkungan kerjanya adalah Google Colab. Kode dijalankan pada komputer jarak jauh melalui browser, bukan melalui instalasi Python di laptop Anda. Karena itu, memahami urutan sel dan sifat penyimpanan sementara sama pentingnya dengan memahami gambar. Setelah alur ini terasa masuk akal, operasi seperti memotong foto atau mengatur kecerahan akan lebih mudah dipahami sebagai perubahan data, bukan sekadar efek visual.

2. Pahami konsep secara berurutan

2.1 Dari foto menjadi data piksel

Citra digital adalah representasi gambar sebagai sampel-sampel pada kisi baris dan kolom. Setiap lokasi disebut piksel. Pada foto daun, satu piksel bisa berada pada urat daun, bagian permukaan, atau latar. Piksel tidak menyimpan nama objek tersebut; nilainya hanya mewakili informasi intensitas atau komponen warna pada lokasi itu. Mengenali objek memerlukan langkah penafsiran lebih lanjut, sedangkan materi ini berfokus pada membaca dan memeriksa representasinya.

Pengolahan citra berarti menerapkan operasi pada data gambar untuk tujuan tertentu. Contohnya, memperkecil foto agar ringan ditampilkan atau mengatur intensitas agar perbedaan bagian lebih terlihat. Alurnya dimulai dari input, pemeriksaan data, operasi, dan pengamatan output. Pemeriksaan ditempatkan di awal karena file yang gagal dibaca tidak dapat diperlakukan sebagai citra yang sah, sekalipun nama berkasnya berakhiran JPG.

2.2 Dimensi, jumlah piksel, dan arti resolusi

Dimensi spasial menyatakan lebar dan tinggi kisi piksel. Dalam penulisan gambar, urutannya lazim berupa lebar × tinggi. Dalam array NumPy, `img.shape` diawali tinggi lalu lebar karena array diakses menurut baris dan kolom. Karena itu, `h, w = img.shape[:2]` mengambil tinggi dan lebar, bukan sebaliknya. Pada citra warna, angka berikutnya menyatakan jumlah kanal.

Jumlah piksel diperoleh dari lebar dikalikan tinggi, tanpa mengalikan kanal. Tiga komponen warna pada satu lokasi tetap merupakan satu piksel, bukan tiga lokasi berbeda. Dalam materi ini, resolusi merujuk pada dimensi piksel, bukan ukuran cetak dalam sentimeter atau kerapatan cetak DPI. Memperbanyak piksel melalui pembesaran tidak otomatis menghadirkan detail baru: foto yang sejak awal tidak fokus tetap kehilangan informasi yang tidak pernah terekam dengan jelas.

2.3 Kanal, grayscale, dan urutan warna

Kanal adalah komponen nilai yang disimpan untuk setiap lokasi. Citra grayscale, atau citra tingkat keabuan, biasanya memiliki array dua dimensi dengan satu nilai intensitas per piksel. Citra warna lazim memiliki tiga kanal. OpenCV menggunakan urutan BGR, yaitu biru, hijau, lalu merah. Matplotlib justru menafsirkan tiga komponen sebagai RGB, yaitu merah, hijau, lalu biru. Menampilkan array BGR langsung sebagai RGB dapat menukar penampilan merah dan biru.

Untuk menjaga warna tampilan, kode mengonversi BGR menjadi RGB sebelum `imshow()`. Pengukuran tetap menggunakan array asli. Beberapa PNG mempunyai kanal alpha yang menyatakan tingkat transparansi; urutannya dalam array OpenCV menjadi BGRA, lalu dikonversi ke RGBA untuk tampilan. Pilihan `cv2.IMREAD_UNCHANGED` mempertahankan

representasi kanal saat decoding, sehingga gambar grayscale tidak otomatis dipaksa menjadi tiga kanal.

Tipe data `uint8` berarti bilangan bulat tanpa tanda dengan delapan bit per komponen, berentang 0–255. Pada grayscale, nilai rendah tampak gelap dan nilai tinggi tampak terang. Pada warna, ketiga komponen membentuk warna bersama-sama; nilai tinggi pada satu kanal saja tidak berarti putih. Rentang ini bukan aturan semua citra digital: gambar dengan kedalaman bit lain dapat memiliki rentang berbeda. Kode membatasi input ke uint8 agar pengamatan konsisten.

2.4 Ukuran berkas berbeda dari ukuran array

File JPEG atau PNG menyimpan data dalam format tertentu. Decoding adalah proses membaca representasi itu menjadi array piksel. Dalam kode, `np.frombuffer(data, np.uint8)` lebih dahulu menyediakan deretan byte berkas kepada `cv2.imdecode()`; deretan tersebut belum merupakan kisi piksel yang siap diukur. Hasil decoding diperiksa agar kegagalan baca tidak diteruskan ke langkah pengukuran.

Nilai `file_bytes` berasal dari `len(data)`, yaitu jumlah byte berkas unggahan. Sebaliknya, `img.nbytes` menunjukkan jumlah byte elemen array hasil decoding, bukan seluruh penggunaan memori program. Untuk uint8, setiap komponen memakai satu byte; karena itu, array warna tiga kanal memerlukan tiga byte komponen per piksel. Kanal alpha menambahkan satu komponen lagi.

Ukuran file dapat jauh berbeda dari ukuran array karena kompresi dan informasi tambahan di dalam berkas. JPEG menggunakan kompresi lossy yang dapat membuang sebagian informasi; PNG menggunakan kompresi lossless. Dua foto berdimensi sama dapat berbeda ukuran filenya karena isi dan pengaturan penyimpanan. Maka, file lebih besar bukan bukti tunggal bahwa gambar lebih tajam atau lebih berguna.

2.5 Koordinat dan pembacaan satu piksel

Koordinat gambar biasa ditulis `(x, y)`: x bergerak ke kanan dan y bergerak ke bawah dari sudut kiri atas. Akses NumPy menggunakan `img[y, x]` karena urutannya baris lalu kolom. Indeks dimulai dari nol; x harus lebih kecil daripada lebar dan y lebih kecil daripada tinggi. Koordinat yang sama dengan ukuran dimensi sudah berada di luar batas.

Sel contoh memakai `x, y = w // 2, h // 2` untuk memilih titik tengah. Pada grayscale, hasilnya satu nilai; pada BGR, hasilnya tiga komponen sesuai urutan kanal. Titik tengah tidak selalu berada pada objek: pada satu foto ia mungkin mengenai daun, pada foto lain mengenai latar. Jadi, perbedaan nilai tidak boleh langsung ditafsirkan sebagai perbedaan objek tanpa melihat lokasinya.

2.6 Menjalankan alur pembacaan di Colab

Notebook Colab berisi sel yang dijalankan pada runtime, yaitu lingkungan eksekusi Python jarak jauh. Sel import menyediakan pustaka, sel upload membentuk variabel `gambar`, lalu sel berikutnya menggunakan variabel itu. Kode memeriksa tepat tiga JPEG/PNG berbeda, batas 10 MiB per file, maksimal 12 juta piksel, serta tipe dan kanal yang didukung. Batas tersebut adalah penjagaan input praktikum, bukan definisi universal gambar digital.

Gunakan Matplotlib untuk menampilkan gambar di notebook, bukan mengharapkan jendela desktop lokal. Ketika runtime direset, variabel dan file sementara dapat hilang meskipun notebook masih tersedia. Jika muncul `NameError`, jalankan kembali import dan upload sebelum pengukuran. Simpan notebook beserta output yang ingin dipertahankan; jangan menganggap unggahan sementara sebagai arsip permanen.



Foto sumber: Wikimedia Commons, CC BY 3.0; perbesaran tidak menambah detail baru.

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

---

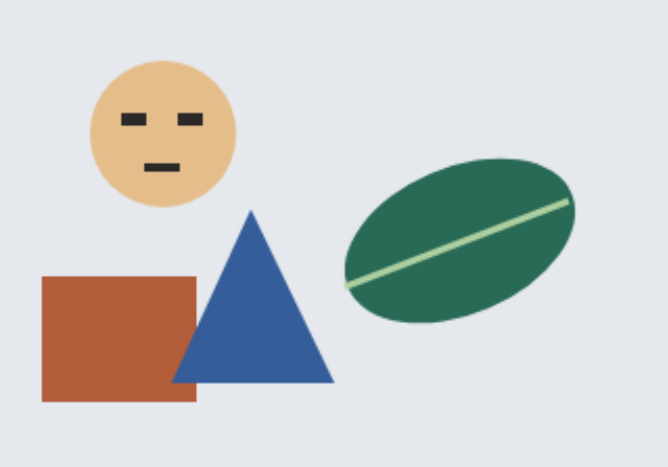
3. Ilustrasi interaktif sebelum praktikum

Inspeksi piksel dan kanal citra

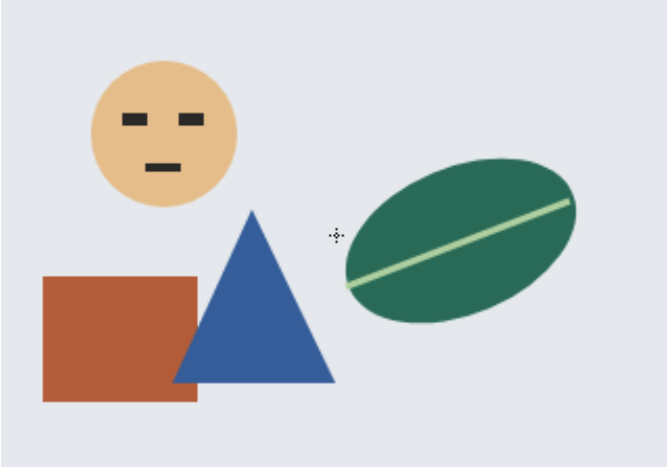
Pilih koordinat pada citra kerja: x adalah kolom, y adalah baris (indeks mulai 0). RGBA dibaca dari buffer browser, bukan kanal asli berkas. Penanda silang menunjukkan posisi terpilih; angka di bawah berasal dari citra sebelum penanda.

Pembacaan piksel berasal dari citra kerja browser yang sudah diubah menjadi RGBA. Ini bukan pemeriksaan channel asli berkas JPEG/PNG. Jumlah byte array, dimensi piksel, dan ukuran file merupakan hal berbeda; gambar contoh diberi label sintetis.

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: Posisi x / kolom (%) = 50 · Posisi y / baris (%) = 50 · Tampilan kanal = rgb

Piksel [baris, kolom] = [112, 160]; RGB = (229, 233, 237); gray = 232.

RGBA = (229, 233, 237, 255); urutan BGR = (237, 233, 229). BGR hanya urutan nilai, bukan klaim format berkas.

Gray = round(0.299R + 0.587G + 0.114B) = 232; dihitung dari RGB, bukan mengandalkan buffer gray masukan.

Citra kerja 320 × 224: 71680 byte per kanal 8-bit; RGBA 286720 byte (4 kanal). Dimensi asli 320 × 224.

Ukuran berkas asli: tidak tersedia. Ukuran berkas terkompresi tidak sama dengan byte RGBA kerja; jumlah kanal berkas asli tidak diketahui dari buffer browser.

▼ **Angka dan parameter hasil perhitungan ilustrasi**


```
{
  "row": 112,
  "column": 160,
  "rgba": [
    229,
    233,
    237,
    255
  ],
  "bgr": [
    237,
    233,
    229
  ],
  "gray": 232,
  "workingWidth": 320,
  "workingHeight": 224,
  "channelBytes": 71680,
  "workingRGBABytes": 286720,
  "originalFileBytes": null,
  "originalFileChannels": "unknown",
  "view": "rgb"
}
```


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

25 - Reading Images, Splitting Channels, Resizing using openCV in Python — DigitalSreeni.

Fokus saat menonton: Amati cara membaca gambar, shape array, nilai piksel dan urutan BGR. Bedakan ukuran berkas dengan resolusi.

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 — Getting Started with Images
- NumPy — Indexing on ndarrays
- Google Colab — FAQ

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.

Buka notebook Google Colab dan jalankan empat sel kode secara berurutan. Sel 1 mengimpor pustaka. Pada Sel 2, pilih tepat tiga file JPEG/PNG yang isinya berbeda dalam satu dialog `files.upload()`. Gunakan gambar milik sendiri atau berizin tanpa data sensitif. Setiap file harus memenuhi batas 10 MiB, maksimal 12 juta piksel, bertipe uint8, dan memiliki satu, tiga, atau empat kanal. Jika validasi berhenti, perbaiki pilihan file lalu jalankan kembali sel upload sebelum melanjutkan.

Sel 3 menampilkan ketiganya dan mencetak laporan. Cocokkan `shape` dengan `lebar` dan `tinggi`; bedakan `jumlah_pixel` dari `channel`, serta `file_bytes` dari `array_bytes`. Sel 4 membaca titik tengah melalui `img[y, x]`. Lihat lokasi itu pada gambar sebelum menafsirkan nilainya sebagai bagian objek atau latar. Jika mencoba koordinat lain, pertahankan x dan y di dalam batas gambar serta baca komponen sesuai urutan kanal yang dicetak.

Jika warna tampak tertukar, periksa konversi BGR ke RGB atau BGRA ke RGBA pada tampilan, bukan mengubah urutan array secara sembarang. Jika runtime terputus atau direset, mulai kembali dari import dan upload karena variabel maupun file sementara mungkin sudah hilang. Simpan notebook dan output yang ingin tetap dibaca setelah sesi berakhir.

Sel 1 — 1. Import library di Colab

Buat notebook baru, kemudian jalankan sel import ini terlebih dahulu. Library umum Colab tidak perlu diinstal lokal.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files
print("OpenCV:", cv2.__version__)
```

Sel 2 — 2. Upload dan validasi tiga gambar

Pilih tepat tiga JPEG/PNG berbeda dalam satu dialog. Maksimum 10 MiB dan 12 juta pixel per gambar; input harus uint8 dengan 1, 3, atau 4 channel.

```
uploaded = files.upload()
if len(uploaded) != 3:
    raise ValueError("Pilih tepat tiga file gambar berbeda; jalankan
ulang sel upload.")
if len(set(uploaded.values())) != 3:
    raise ValueError("Isi ketiga file harus berbeda, bukan salinan
file yang sama.")
gambar = []
for nama, data in uploaded.items():
    if not nama.lower().endswith((".jpg", ".jpeg", ".png")):
        raise ValueError(f"{nama}: gunakan JPEG atau PNG.")
    if not data or len(data) > 10 * 1024 * 1024:
        raise ValueError(f"{nama}: file kosong atau melebihi 10
MiB.")
    img = cv2.imdecode(np.frombuffer(data, np.uint8),
cv2.IMREAD_UNCHANGED)
    if img is None:
        raise ValueError(f"{nama}: tidak dapat didecode sebagai
gambar.")
    h, w = img.shape[:2]
    channel = 1 if img.ndim == 2 else img.shape[2]
    if h * w > 12_000_000 or img.dtype != np.uint8 or channel not in
(1, 3, 4):
        raise ValueError(f"{nama}: gunakan gambar uint8 <=12 MP,
channel 1/3/4.")
    gambar.append({"nama": nama, "img": img, "file_bytes":
```

```
len(data)})
print("Tiga gambar berhasil dibaca.")
```

Sel 3 — 3. Tampilkan gambar dan laporan ukuran

Output membedakan dimensi spasial, jumlah pixel, jumlah channel, byte file, dan byte array. Urutan channel hanya dikonversi saat tampilan.

```
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
for ax, item in zip(axes, gambar):
    img = item["img"]
    h, w = img.shape[:2]
    c = 1 if img.ndim == 2 else img.shape[2]
    if c == 1:
        ax.imshow(img, cmap="gray", vmin=0, vmax=255)
    else:
        flag = cv2.COLOR_BGR2RGB if c == 3 else cv2.COLOR_BGRA2RGBA
        ax.imshow(cv2.cvtColor(img, flag))
    ax.set_title(f"{item['nama']}\n{w} × {h} pixel; {c} channel")
    ax.axis("off")
    print({"nama": item["nama"], "shape": img.shape, "lebar": w,
           "tinggi": h, "jumlah_pixel": w*h, "channel": c,
           "dtype": str(img.dtype), "file_bytes":
item["file_bytes"],
           "array_bytes": img.nbytes})
plt.tight_layout()
plt.show()
```

Sel 4 — 4. Baca pixel pada titik tertentu

Contoh memilih titik tengah yang selalu valid. Untuk latihan, ubah koordinat pada setiap gambar, tetap periksa rentangnya, dan jelaskan apakah titik berada pada objek atau latar.

```
for item in gambar:
    img = item["img"]
    h, w = img.shape[:2]
    x, y = w // 2, h // 2
    if not (0 <= x < w and 0 <= y < h):
        raise ValueError("Koordinat pixel di luar gambar.")
    c = 1 if img.ndim == 2 else img.shape[2]
    urutan = {1: "grayscale", 3: "B, G, R", 4: "B, G, R, A"}[c]
```

```
    print(f"{item['nama']}: (x,y)=({x},{y}), nilai=
{img[y,x].tolist()}, urutan={urutan}")
```

Jika terjadi error

Jumlah file tidak tepat tiga atau isi file identik.

Pilih tiga file berbeda sekaligus dan jalankan ulang sel upload.

Foto tampil dengan merah dan biru tertukar.

Gunakan COLOR_BGR2RGB; untuk empat channel gunakan COLOR_BGRA2RGBA.

IndexError ketika membaca pixel.

Gunakan img[y, x] dengan y < tinggi dan x < lebar; indeks dimulai dari nol.

NameError setelah runtime terputus.

Jalankan ulang import dan upload sebelum sel pengukuran. File unggahan sementara bukan penyimpanan permanen.

> 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.
