PENGOLAHAN CITRA · MATERI 07

Deteksi Wajah dan Objek 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

Saat kamera ponsel menampilkan kotak di sekitar wajah, perangkat sedang menunjukkan lokasi yang dianggap menyerupai wajah. Kotak itu tidak dengan sendirinya berarti perangkat mengetahui nama orang tersebut. Perbedaan kecil ini penting: mencari letak wajah, menghitung wajah, dan mengenali identitas adalah pekerjaan yang berbeda.

Bayangkan sebuah foto bersama di ruang kelas. Sebagian orang menghadap kamera, sebagian menoleh, dan ada wajah yang tertutup teman di depannya. Kita mungkin masih dapat memperkirakan jumlah orang, tetapi program belum tentu menemukan semua wajah. Sebaliknya, pola terang-gelap pada benda lain dapat keliru dianggap wajah. Hasil deteksi lebih tepat disebut **kandidat**, yaitu dugaan yang masih perlu diperiksa pada gambar.

Dalam bab ini, kita memakai Haar cascade, model yang sudah dilatih untuk mencari pola visual tertentu. Model wajah dan model mata dimuat secara terpisah. Foto diubah menjadi grayscale, atau citra tingkat keabuan, agar pencarian bekerja pada informasi terang-gelap. Hasilnya berupa koordinat kotak, sehingga kita dapat menunjukkan lokasi dan menghitung kandidat yang ditemukan. Saat mata dicari di dalam potongan wajah, kita juga belajar menghubungkan koordinat bagian gambar dengan koordinat gambar utuh.

Pada bagian pengayaan, perhatian berpindah dari wajah ke benda berwarna hijau. Posisi pusat benda di beberapa tangkapan kamera dihubungkan menjadi jejak sederhana. Jejak ini membantu memahami pelacakan, yaitu mengikuti perubahan posisi, tetapi belum menjamin benda yang sama selalu diikuti. Unggah foto tetap menjadi jalur utama; kamera browser bersifat opsional. Dengan alur ini, kita belajar membaca keluaran secara kritis, memahami pengaruh parameter dan kondisi foto, serta menjaga izin penggunaan gambar orang lain tanpa mencampurkan deteksi dengan pengenalan identitas.

---

2. Pahami konsep secara berurutan

2.1 Memisahkan deteksi, hitungan, dan identitas

Deteksi mencari lokasi kandidat objek pada sebuah gambar. Untuk wajah, keluarannya berupa kotak dengan koordinat kiri atas x dan y, lebar w, serta tinggi h. Kotak membantu menunjukkan bagian yang dipilih model, tetapi bukan garis batas wajah yang mengikuti bentuk kulit atau rambut. Menghitung elemen hasil deteksi menghasilkan jumlah kandidat pada gambar tersebut.

Jumlah kandidat belum tentu sama dengan jumlah wajah sebenarnya. Wajah yang terlewat disebut false negative, sedangkan daerah bukan wajah yang mendapat kotak disebut false positive. Keduanya harus dilihat pada gambar, bukan hanya dari angkanya. Hitungan yang kebetulan sama dengan pengamatan manusia masih mungkin memuat satu wajah terlewat dan satu kotak keliru. Pengenalan identitas berbeda lagi: ia menjawab siapa orangnya. Praktikum ini tidak membangun proses pengenalan tersebut, dan wajah orang yang sama pada dua tangkapan tidak otomatis dicatat sebagai satu identitas unik.

2.2 Memahami penyaring bertingkat Haar cascade

Haar cascade dapat dibayangkan sebagai rangkaian penyaring yang memeriksa pola perbedaan terang-gelap pada bagian-bagian gambar. Jendela gambar yang tidak memenuhi pemeriksaan awal ditolak lebih cepat. Kandidat yang lolos diperiksa pada tahap berikutnya. Cascade berarti tahapan itu disusun berurutan, bukan seluruh bagian gambar langsung dinyatakan sebagai wajah.

Kita menggunakan model yang sudah dilatih dan disimpan dalam berkas XML. Foto yang diunggah menjadi masukan pemeriksaan, bukan bahan yang otomatis melatih model baru. Berkas untuk wajah frontal berbeda dari berkas untuk mata. Pemeriksaan `empty()` memastikan model berhasil dimuat; model yang gagal dibaca harus ditangani sebelum hasil deteksinya dibahas. Sifat model frontal juga menjelaskan mengapa wajah menghadap depan lebih sesuai dengan contoh dibanding wajah menyamping atau tertutup sebagian.

2.3 Menyiapkan gambar dan memahami pencarian multiskala

OpenCV membaca foto berwarna dalam urutan kanal BGR. Deteksi memakai salinan grayscale yang berisi intensitas, sedangkan anotasi digambar pada salinan foto berwarna. Saat ditampilkan dengan Matplotlib, foto BGR diubah menjadi RGB agar warna tidak tertukar. Memisahkan citra untuk perhitungan dan citra untuk tampilan membantu menjaga masukan asli tetap mudah dibandingkan dengan hasil.

`detectMultiScale` mencari kandidat pada beberapa skala karena wajah dapat berukuran berbeda. `scaleFactor` mengatur perubahan skala antar tingkat pencarian dan nilainya harus lebih besar dari satu. Langkah skala yang lebih rapat dapat menambah pekerjaan pencarian.

`minSize` membatasi ukuran minimum kandidat dalam piksel, sehingga wajah yang lebih kecil daripada batas itu tidak menjadi sasaran pencarian contoh.

`minNeighbors` berhubungan dengan dukungan kelompok kandidat agar sebuah deteksi dipertahankan. Ini bukan jumlah orang di sekitar wajah dan bukan persentase keyakinan. Nilai lebih besar cenderung membuang kandidat lemah, tetapi juga dapat membuang wajah yang sebenarnya ada. Kode membandingkan nilai 3 dan 7 pada foto yang sama. Perbandingan demikian lebih jelas daripada mengganti foto sekaligus mengganti parameter, sebab penyebab perubahan hasil menjadi lebih mudah dilacak.

2.4 Mencari mata di bagian wajah dan mengembalikan koordinat

ROI, singkatan dari region of interest, berarti bagian gambar yang menjadi daerah perhatian. Setelah kotak wajah diperoleh, potongan grayscale di dalam kotak itu menjadi ROI untuk pencarian mata. Pencarian lebih terarah karena model mata tidak dijalankan ke seluruh latar foto.

Koordinat di dalam ROI dimulai dari sudut kiri atas potongan, bukan dari sudut kiri atas foto utuh. Karena itu, posisi mata `ex, ey` perlu ditambah posisi wajah `x, y` ketika digambar kembali. Tanpa penambahan tersebut, kotak mata dapat bergeser ke bagian lain foto. Deteksi mata tetap merupakan dugaan: kacamata, arah pandang, bayangan, dan ketajaman gambar dapat memengaruhinya. Tidak ditemukannya dua mata tidak langsung membatalkan keberadaan wajah.

2.5 Mengikuti benda melalui warna dan titik pusat

Pelacakan sederhana pada bagian opsional memakai prinsip berbeda dari Haar cascade. Setiap tangkapan diubah ke HSV, representasi warna yang memisahkan rona, kejenuhan, dan kecerahan. Batas nilai HSV memilih daerah yang dianggap hijau menjadi mask. Opening mengurangi bintik kecil, lalu kontur terbesar dipilih sebagai kandidat benda yang akan diikuti apabila luasnya cukup.

Centroid adalah pusat geometris daerah kontur. Kode menghitungnya dengan moments, ringkasan distribusi geometri daerah, dan hanya membagi jika luas pada perhitungannya tidak nol. Pusat-pusat yang diperoleh berurutan dihubungkan menjadi jejak. Ketika tidak ada kandidat yang cukup luas, kode mengosongkan jejak agar garis tidak menyambung melalui masa hilangnya target.

Asumsi utamanya adalah satu benda hijau dominan dengan latar yang tidak membingungkan. Benda hijau lain bisa menjadi kontur terbesar dan mengambil alih jejak tanpa diketahui program. Perubahan cahaya juga dapat mengubah daerah mask. Jadi, garis yang terlihat tidak membuktikan identitas target tetap sama. Contoh ini mengajarkan perubahan posisi, bukan pelacak banyak objek atau pengukur kecepatan fisik yang telah dikalibrasi.

2.6 Memahami kamera Colab dan batas penggunaan hasil

Python di Colab berjalan pada runtime jarak jauh, sedangkan kamera pengguna berada pada perangkat yang membuka browser. Karena itu, contoh meminta izin kamera melalui JavaScript di browser, kemudian mengirimkan tangkapan ke Python. Kamera tidak dibuka dengan menganggap runtime memiliki kamera laptop pengguna. Pengambilan dibatasi paling banyak dua belas frame dalam batas sepuluh detik setelah kamera siap; ini bukan tayangan deteksi langsung yang terus berjalan. Jalur foto tetap dapat dipakai ketika kamera tidak tersedia.

Gunakan gambar sendiri atau gambar orang lain dengan izin yang sesuai. Izin menyalakan kamera tidak sama dengan izin menyebarluaskan tangkapan. Hindari foto sensitif dan hentikan penggunaan ketika persetujuan tidak ada. Beberapa contoh keberhasilan tidak membuktikan model bekerja sama baik pada semua orang atau kondisi. Kesimpulan yang wajar tetap terbatas pada kandidat yang terlihat, kondisi masukan, dan kesalahan yang dapat diperiksa.



Hasil deteksi benar-benar dijalankan; bukan pengenalan identitas orang.

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

---

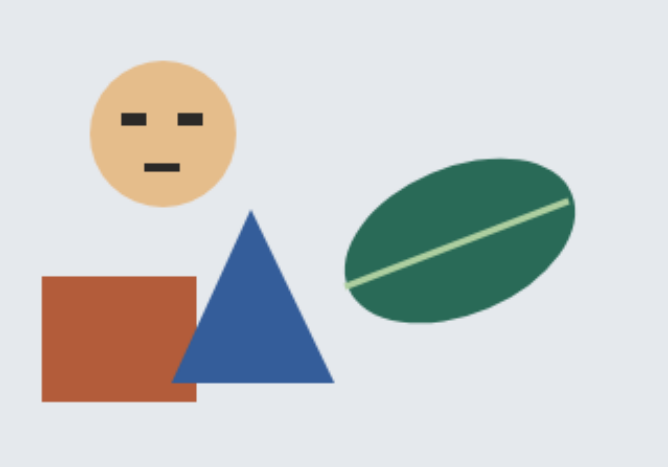
3. Ilustrasi interaktif sebelum praktikum

Jendela pindai dan kontras ala Haar

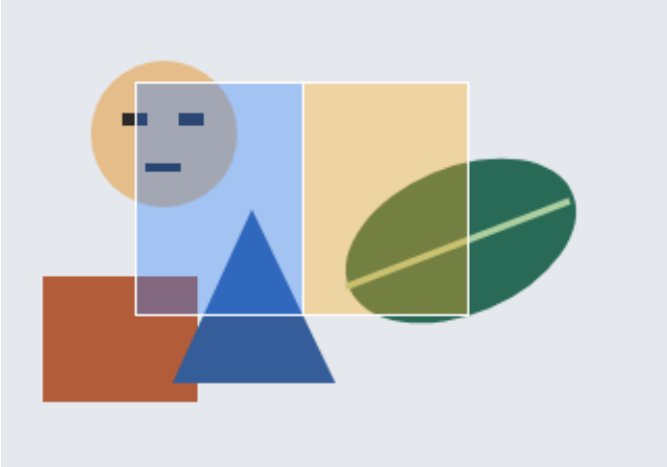
Pindahkan ROI untuk menghitung beda rerata dua wilayah. Ini hanya ilustrasi fitur kontras ala Haar: bukan cascade terlatih, deteksi wajah, pengenalan identitas, atau klasifikasi.

Ilustrasi ini memperlihatkan jendela dan perbedaan kontras lokal. Ini bukan model Haar yang dilatih dan bukan pendeteksi atau pengenal identitas wajah. Deteksi cascade yang sebenarnya dijalankan pada praktikum Colab.

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 ilustrasi · baca jenis operasi dan batasannya

Parameter: Posisi horizontal ROI (%) = 40 · Posisi vertikal ROI (%) = 35 · Ukuran ROI (% sisi gambar) = 50 · Pola dua wilayah = vertical

ROI (64,39), 160×112; rerata A 191.21, B 178.42; B−A = -12.79.

A biru, B jingga; pola kiri/kanan. Posisi persen dihitung dalam sisa ruang agar ROI tetap di gambar. Nilai dari grayscale asli sebelum pewarnaan. Beda rerata ternormalisasi, bukan skor model Haar terlatih. Kotak adalah jendela pilihan pengguna, bukan wajah terdeteksi; tidak ada hitungan wajah.

▼ **Angka dan parameter hasil perhitungan ilustrasi**


```
{
  "x": 64,
  "y": 39,
  "width": 160,
  "height": 112,
  "pattern": "vertical",
  "meanA": 191.20558035714285,
```

```
  "meanB": 178.4171875,
  "difference": -12.788392857142838,
  "countA": 8960,
  "countB": 8960,
  "sampleCount": 17920,
  "valid": true
}
```


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

Haar Cascade Object Detection Face & Eye - OpenCV with Python for Image and Video Analysis 16 — sentdex.

Fokus saat menonton: Amati ROI wajah dan mata serta hasil Haar cascade. Kotak deteksi tidak membuktikan identitas seseorang.

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 — Cascade Classifier
- Google Colab — Advanced outputs (kamera browser)
- 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.

Jalankan setup OpenCV sebelum impor, kemudian muat model wajah dan mata sampai pemeriksaan pemuatannya berhasil. Jalur utama memakai satu foto JPG/PNG berizin, maksimal 10 MiB dengan sisi maksimal 6000 piksel. Foto yang besar diperkecil oleh kode; pembacaan hasil dilakukan pada citra yang benar-benar ditampilkan, bukan dengan menganggap ukurannya selalu sama dengan berkas awal.

Foto masuk ke fungsi `deteksi_wajah`, diubah menjadi grayscale, lalu menghasilkan kotak wajah. Pencarian mata berlangsung di ROI setiap wajah, dan kotaknya dikembalikan ke koordinat gambar penuh. Keluaran memuat gambar beranotasi dan jumlah kandidat. Tampilan perbandingan `minNeighbors` memperlihatkan pengaruh parameter pada masukan yang sama; jumlah yang lebih banyak belum tentu lebih benar.

Sel webcam dan pelacakan benda hijau bersifat opsional. Jika dipakai, jalankan tangkapan browser terlebih dahulu agar `frames_webcam` tersedia, baru jalankan pengolahan yang menggunakannya. Tangkapan terakhir dapat diperiksa dengan detektor wajah, sedangkan rangkaian tangkapan menjadi masukan pelacakan warna. Jika izin kamera ditolak, lewati kedua bagian tersebut dan gunakan foto; hasil tangkapan terbatas ini bukan video real-time atau pengenalan identitas.

Sel 1 — 0. Setup OpenCV 4.x

Jalankan sebelum import cv2: baseline praktikum memakai OpenCV 4.x karena API CascadeClassifier tidak tersedia pada sebagian instalasi 5.x. Jika cv2 sudah diimport, restart runtime melalui menu Runtime, lalu jalankan sel dari awal. Jangan menaikkan versi NumPy secara terpisah.

```
%pip install "opencv-python-headless>=4.10,<5"
```

Sel 2 — 1. Import dan model cascade

Jalankan sebelum mengunggah foto. Model bawaan harus terbaca; pesan kesalahan menghentikan praktik.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```
from google.colab import files, output
from IPython.display import display, Javascript
import base64
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_eye.xml')
if face_cascade.empty() or eye_cascade.empty():
    raise RuntimeError('Berkas cascade tidak berhasil dimuat.')
print('Cascade wajah dan mata siap.')
```

Sel 3 — 2. Unggah satu foto

Gunakan JPG/PNG maksimal 10 MB dan sisi maksimal 6000 piksel. Foto besar diperkecil untuk demo; hitungan manual mengikuti citra yang ditampilkan.

```
uploaded = files.upload()
if len(uploaded) != 1:
    raise ValueError('Unggah tepat satu foto JPG/PNG.')
nama, data = next(iter(uploaded.items()))
if not nama.lower().endswith(('.jpg', '.jpeg', '.png')) or len(data)
> 10 * 1024 * 1024:
    raise ValueError('Format harus JPG/PNG dan ukuran maksimal 10
MB.')
img = cv2.imdecode(np.frombuffer(data, np.uint8), cv2.IMREAD_COLOR)
if img is None:
    raise ValueError('Isi berkas tidak dapat dibaca sebagai citra.')
h, w = img.shape[:2]
if max(h, w) > 6000:
    raise ValueError('Sisi gambar maksimal 6000 piksel.')
scale = min(1.0, 1200 / max(h, w))
if scale < 1:
    img = cv2.resize(img, (round(w * scale), round(h * scale)))
plt.figure(figsize=(10, 6))
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```

Sel 4 — 3. Deteksi wajah dan mata

Kotak mata dikembalikan ke koordinat citra penuh. Jumlah wajah ditampilkan, termasuk jika hasilnya nol.

```
def deteksi_wajah(citra, tetangga=5):
    gray = cv2.cvtColor(citra, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1,
                                          minNeighbors=tetangga,
minSize=(30, 30))
    hasil = citra.copy()
    for x, y, w, h in faces:
        cv2.rectangle(hasil, (x, y), (x+w, y+h), (0, 255, 0), 2)
        roi = gray[y:y+h, x:x+w]
        eyes = eye_cascade.detectMultiScale(roi, scaleFactor=1.1,
                                            minNeighbors=5, minSize=
(10, 10))
        for ex, ey, ew, eh in eyes:
            cv2.rectangle(hasil, (x+ex, y+ey), (x+ex+ew, y+ey+eh),
(255, 0, 0), 2)
    cv2.putText(hasil, f'Wajah: {len(faces)}', (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255), 2)
    return hasil, faces
hasil, faces = deteksi_wajah(img)
print('Jumlah kandidat wajah:', len(faces))
plt.figure(figsize=(10, 6))
plt.imshow(cv2.cvtColor(hasil, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```

Sel 5 — 4. Bandingkan parameter

Bandingkan kandidat dengan hitungan manual pada gambar yang sama. Jangan memilih parameter hanya karena jumlahnya lebih tinggi.

```
fig, axes = plt.subplots(1, 2, figsize=(14, 6))
for ax, tetangga in zip(axes, [3, 7]):
    anotasi, kandidat = deteksi_wajah(img, tetangga)
    ax.imshow(cv2.cvtColor(anotasi, cv2.COLOR_BGR2RGB))
    ax.set_title(f'minNeighbors={tetangga}; kandidat=
{len(kandidat)}')
```

```
    ax.axis('off')
plt.show()
```

Sel 6 — 5. Opsional: tangkapan webcam browser

Lewati bila izin kamera tidak tersedia. Jalankan di Colab melalui browser; tangkapan hanya 12 frame/10 detik setelah kamera siap, bukan video real-time.

```
display(Javascript(r"""
window.captureLimited = async function() {
  const video = document.createElement('video');
  video.muted = true; video.playsInline = true;
  const canvas = document.createElement('canvas');
  let stream = null, timer = null, cancelled = false;
  document.body.appendChild(video);
  try {
    const request = navigator.mediaDevices.getUserMedia({video:
true, audio: false});
    request.then(s => { if (cancelled) s.getTracks().forEach(t =>
t.stop()); }, () => {});
    stream = await Promise.race([request, new Promise((_, reject) =>
{
      timer = setTimeout(() => {cancelled = true; reject(new
Error('Izin kamera timeout'));}, 15000);
    })]);
    clearTimeout(timer);
    video.srcObject = stream;
    await Promise.race([video.play(), new Promise((_, reject) => {
      timer = setTimeout(() => reject(new Error('Kamera tidak
siap')), 5000);
    })]);
    clearTimeout(timer);
    canvas.width = 640; canvas.height = 480;
    const ctx = canvas.getContext('2d'), frames = [], start =
performance.now();
    while (frames.length < 12 && performance.now() - start < 10000)
{
      ctx.drawImage(video, 0, 0, 640, 480);
      frames.push(canvas.toDataURL('image/jpeg', 0.8));
      await new Promise(r => setTimeout(r, 400));
    }
    return frames;
```

```
  } finally {
    cancelled = true; clearTimeout(timer);
    if (stream) stream.getTracks().forEach(t => t.stop());
    video.srcObject = null; video.remove(); canvas.remove();
  }
};
"""))
encoded_frames = output.eval_js('captureLimited()')
frames_webcam = []
for item in encoded_frames:
    raw = base64.b64decode(item.split(',', 1)[1])
    frame = cv2.imdecode(np.frombuffer(raw, np.uint8),
cv2.IMREAD_COLOR)
    if frame is None:
        raise ValueError('Tangkapan kamera gagal didekode.')
    frames_webcam.append(frame)
if not frames_webcam:
    raise RuntimeError('Tidak ada frame webcam.')
print('Frame diterima:', len(frames_webcam))
hasil_webcam, faces_webcam = deteksi_wajah(frames_webcam[-1])
print('Jumlah kandidat wajah pada tangkapan terakhir:',
len(faces_webcam))
plt.figure(figsize=(8, 5))
plt.imshow(cv2.cvtColor(hasil_webcam, cv2.COLOR_BGR2RGB))
plt.axis('off'); plt.show()
```

Sel 7 — 6. Opsional: ikuti benda hijau

Gerakkan satu benda hijau saat tangkapan. Kontur terbesar adalah target sementara, bukan identitas tetap; sesuaikan batas HSV dengan cahaya.

```
jejak = []
anotasi_frames = []
for frame in frames_webcam:
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsv, np.array([35, 60, 50]), np.array([85,
255, 255]))
    mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, np.ones((5, 5),
np.uint8))
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    hasil_track = frame.copy()
```

```
    kandidat = max(contours, key=cv2.contourArea) if contours else
None
    if kandidat is not None and cv2.contourArea(kandidat) > 300:
        M = cv2.moments(kandidat)
        if M['m00'] > 0:
            pusat = (int(M['m10']/M['m00']), int(M['m01']/M['m00']))
            jejak.append(pusat)
            cv2.circle(hasil_track, pusat, 7, (0, 0, 255), -1)
    else:
        jejak = []
    for a, b in zip(jejak, jejak[1:]):
        cv2.line(hasil_track, a, b, (0, 255, 255), 2)
    anotasi_frames.append(hasil_track)
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
for ax, indeks in zip(axes, np.linspace(0, len(anotasi_frames)-1, 3,
dtype=int)):
    ax.imshow(cv2.cvtColor(anotasi_frames[indeks],
cv2.COLOR_BGR2RGB))
    ax.set_title(f'Tangkapan {indeks+1}'); ax.axis('off')
plt.show()
```

Jika terjadi error

Tidak ada wajah terdeteksi.

Periksa citra, ukuran wajah, pose frontal, dan pencahayaan; ubah satu parameter lalu bandingkan dengan hitungan manual.

Kotak mata bergeser keluar wajah.

Tambahkan offset x,y wajah pada koordinat lokal ROI; gunakan fungsi yang disediakan.

Kamera ditolak atau eval_js gagal.

Periksa izin browser dan gunakan upload foto sebagai jalur utama. Tidak perlu memaksa akses webcam.

Objek hijau lain mengambil alih jejak.

Gunakan latar berbeda warna dan satu objek dominan; jelaskan keterbatasan metode kontur terbesar.

> 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.
