PENGOLAHAN CITRA · MATERI 08

Pengenalan Video 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

Sebuah video orang berjalan tampak bergerak mulus, padahal layar menampilkan banyak gambar secara berurutan. Satu gambar dalam urutan itu disebut **frame**. Setiap frame dapat diperlakukan seperti foto: dibaca sebagai susunan piksel, diubah ukurannya, atau diberi tulisan. Perbedaannya, pada video pekerjaan tersebut dilakukan berulang sambil menjaga urutan dan informasi waktu.

Bayangkan kita ingin memberi nomor dan penanda waktu pada klip pendek kegiatan di halaman rumah. Nomor frame menunjukkan posisi gambar dalam urutan, sedangkan penanda waktu menunjukkan perkiraan posisinya sejak awal klip. Agar keduanya tidak tertukar, kita perlu memahami **FPS**, kependekan dari frames per second, yaitu banyaknya frame per detik pada alur berkecepatan frame konstan. Angka ini berbeda dari kecepatan komputer memproses video. Komputer boleh bekerja lebih lambat tanpa harus membuat hasil akhirnya diputar lebih lambat, selama pengaturan waktu keluarannya sesuai.

Ukuran gambar juga mempunyai peran tersendiri. Mengecilkan lebar dan tinggi frame membuat tampilannya lebih ringan, tetapi tidak sama dengan mengubah kecepatan pemutaran. Selain itu, video bukan sekadar gambar yang dibungkus dalam nama berkas MP4. Di dalamnya ada cara penyandian gambar, informasi waktu, dan mungkin suara. Jalur pengolahan pada bab ini menulis ulang bagian gambar tanpa mempertahankan audio.

Kita akan mengikuti perjalanan satu klip dari unggah, pemeriksaan informasi berkas, pembacaan frame, perubahan ukuran, penambahan tulisan, hingga pemutaran hasil di Colab. Sepanjang alur itu, waktu relatif dibedakan dari jam perekaman, nilai perkiraan dibedakan dari metadata sumber, dan berkas yang terbentuk dibedakan dari hasil yang benar-benar dapat diputar. Dengan begitu, hasil tidak hanya terlihat selesai, tetapi juga dapat dijelaskan batas dan maknanya.

2. Pahami konsep secara berurutan

2.1 Menghubungkan frame, resolusi, dan kecepatan pemutaran

Frame adalah satu citra dalam urutan video. Resolusi menyatakan lebar dan tinggi citra tersebut, bukan lamanya video. Pada alur frame rate konstan, FPS menyatakan banyak frame yang dipresentasikan setiap detik. Jika jumlah frame tetap tetapi FPS keluaran diubah, lama pemutaran ikut berubah. Sebaliknya, resize mengubah ukuran setiap gambar tanpa harus mengubah waktu pemutarannya.

Kecepatan pemrosesan adalah hal lain lagi: berapa cepat komputer membaca, mengolah, dan menulis frame. Proses yang berlangsung lama tidak otomatis menghasilkan video gerak lambat. Waktu pemutaran keluaran ditentukan oleh urutan frame dan pengaturan waktunya, bukan lama menunggu sel notebook selesai. Pembedaan ini membantu membaca laporan durasi video dan batas waktu kerja program secara terpisah.

2.2 Membuka berkas dan membaca informasi awal

`VideoCapture` pada praktikum menerima jalur berkas yang diunggah, bukan nomor kamera. `isOpened()` memeriksa apakah pembukaan awal berhasil. Setelah itu, `read()` mengembalikan penanda keberhasilan dan citra frame. Penanda perlu diperiksa sebelum citra digunakan. Kegagalan membaca dapat menandai akhir berkas, tetapi juga dapat terjadi karena masalah decode, yaitu penerjemahan data video menjadi gambar.

Metadata adalah informasi yang menyertai berkas, misalnya FPS dan jumlah frame. Informasi ini berguna untuk memperkirakan durasi nominal, tetapi tidak menjamin semua frame benar-benar dapat dibaca. Jumlah frame atau FPS yang tidak tersedia tidak boleh ditafsirkan sebagai video berdurasi nol. Contoh memeriksa frame pertama untuk melihat ukuran gambar yang berhasil didekode, lalu melepaskan pembaca sebelum berkas dibuka lagi untuk pengolahan.

Batas input menjaga praktik tetap ringan: klip pendek, ukuran berkas terbatas, dan sisi gambar tidak terlalu besar. Selain pemeriksaan metadata, perulangan juga mempunyai batas frame, durasi nominal, dan waktu kerja. Ketiganya bukan hal yang sama. Batas waktu kerja dapat menghentikan proses sebelum seluruh klip selesai, sedangkan batas frame mengendalikan banyaknya gambar yang ditulis.

2.3 Menghitung waktu relatif dengan asumsi yang jelas

Indeks internal dimulai dari nol. Waktu relatif dihitung dengan membagi indeks tersebut oleh FPS yang dipakai. Nomor frame pada tulisan dimulai dari satu agar mudah dibaca manusia. Jadi, frame yang bertuliskan nomor satu memiliki waktu relatif nol, bukan waktu satu detik. Indeks, nomor tampilan, dan satuan waktu perlu dibedakan agar anotasi tidak menyesatkan.

Waktu relatif menyatakan posisi dalam urutan yang diproses. Ia bukan tanggal atau jam ketika kamera merekam peristiwa. Perhitungan indeks dibagi FPS juga mengasumsikan selang waktu antarframe seragam. Pada video dengan frame rate bervariasi, selang presentasi dapat berbeda-beda sehingga satu nilai FPS tidak cukup untuk merekonstruksi seluruh waktunya secara presisi.

Kode menerima metadata FPS dalam rentang demo tertentu. Jika metadata tidak valid atau berada di luar rentang itu, nilai pengganti 25 FPS dipakai dan ditandai sebagai estimasi. Pengganti ini membantu alur berjalan, tetapi bukan hasil pengukuran kecepatan sumber. Waktu relatif serta durasi yang diturunkan darinya ikut bersifat perkiraan. Kebutuhan analisis waktu yang lebih teliti memerlukan informasi timestamp presentasi sumber, bukan hanya nomor urut frame.

2.4 Mengolah dan menulis setiap frame secara konsisten

`VideoWriter` dibuat dengan codec, FPS, dan ukuran keluaran yang ditetapkan. Setiap frame harus sesuai ukuran dan format tersebut; writer tidak boleh diasumsikan memperbaiki ketidakcocokan secara otomatis. Kode melakukan resize sebelum menulis frame dan mempertahankan representasi BGR untuk jalur OpenCV. Konversi ke RGB hanya diperlukan ketika cuplikan ditampilkan dengan Matplotlib.

Resize mengikuti skala yang sama untuk lebar dan tinggi agar benda tidak sengaja dipipihkan. Dimensi keluaran dibuat genap untuk memenuhi kebutuhan codec tertentu. Pembulatan tersebut dapat menimbulkan selisih kecil pada rasio gambar, tetapi berbeda dari sengaja mengubah lebar dan tinggi tanpa menjaga proporsi. Karena ukuran tampilan mengecil, tulisan juga disesuaikan agar tetap terbaca.

Nomor frame dan waktu digambar di atas latar gelap sehingga tidak mudah hilang pada pemandangan terang. Frame pertama yang sudah dianotasi disimpan sebagai cuplikan pemeriksaan. Pembaca dan penulis dilepaskan melalui `finally`, termasuk jika terjadi kesalahan. Pelepasan writer penting untuk menyelesaikan penulisan struktur berkas; sekadar melihat nama file muncul belum membuktikan video selesai ditulis.

2.5 Membedakan kontainer, codec, dan audio

MP4 adalah kontainer, yaitu wadah bagi aliran data dan informasi yang berkaitan. Codec menentukan cara gambar disandikan dan dibaca kembali. Dua berkas berakhiran MP4 dapat memakai codec berbeda, sehingga ekstensi saja tidak menjamin keduanya dapat diputar browser. Mengganti nama ekstensi juga tidak mengubah penyandian di dalamnya.

Contoh membuat berkas kerja memakai `mp4v`, kemudian melakukan transkode dengan FFmpeg ke H.264 dan format piksel `yuv420p` untuk jalur tampilan browser. Transkode berarti menyandikan ulang, bukan hanya menyalin nama berkas. Jika encoder tidak tersedia atau proses gagal, pesan kesalahannya harus dibaca. Jalur ini menulis gambar tanpa audio; suara

sumber tidak otomatis ikut karena tiap frame gambar telah dibaca dan ditulis ulang secara terpisah.

2.6 Memeriksa hasil sebelum menyatakan pemrosesan lengkap

Pemeriksaan hasil dilakukan bertingkat. Berkas harus ada dan tidak kosong, lalu dapat dibuka ulang, frame pertamanya dapat didekode, dan ukurannya sesuai konfigurasi. Itu membuktikan sebagian sifat berkas, bukan bahwa seluruh isinya sempurna. Player notebook tetap perlu diputar untuk memeriksa tampilan, keterbacaan tulisan, dan perilaku pemutaran pada browser yang digunakan.

Jumlah frame yang diproses dan alasan berhenti membantu menafsirkan cakupan hasil. Berhenti karena batas kerja berarti hasilnya mungkin hanya cuplikan; berhenti saat pembacaan gagal belum selalu membedakan akhir normal dan gangguan decode. Metadata keluaran pun dibaca sebagai informasi pendukung, bukan pengganti pemeriksaan. Kesimpulan akhirnya menyebut ukuran, waktu yang dipakai, status estimasi, serta hilangnya audio sesuai jalur pengolahan yang benar-benar dijalankan.

Frame, FPS dan timestamp



Ilustrasi frame 0



Ilustrasi frame 5



Ilustrasi frame 10



Sumber foto untuk simulasi frame

Ilustrasi frame dibuat dari satu foto; bukan rekaman perjalanan bus nyata.

Ilustrasi frame dibuat dari satu foto; bukan rekaman perjalanan bus nyata.

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

---

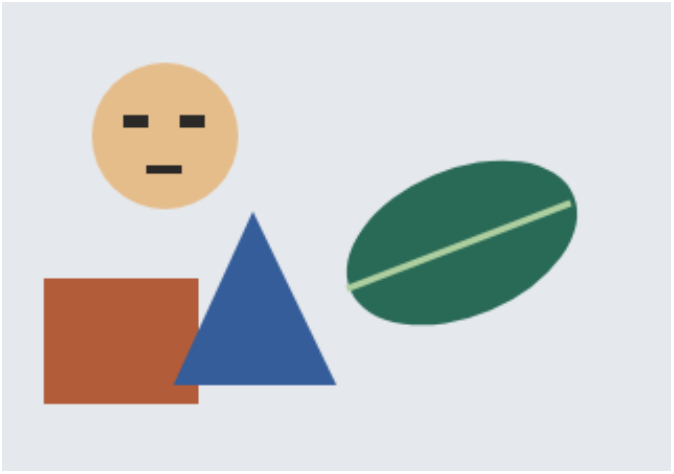
3. Ilustrasi interaktif sebelum praktikum

Frame buatan, FPS, dan waktu

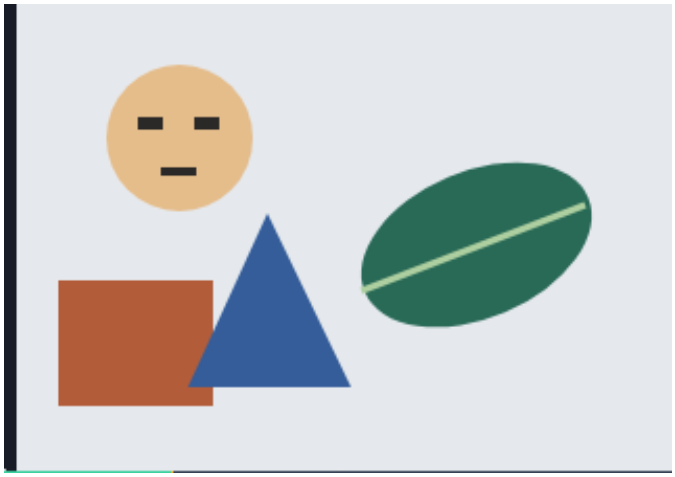
Bangun urutan frame simulasi dari satu gambar diam dengan translasi horizontal. Waktu sumber simulasi = indeks/FPS; durasi = jumlah frame/FPS. Tidak membaca berkas video dan tidak mendeteksi gerakan.

Frame di ilustrasi ini dibuat dari satu gambar diam untuk memahami waktu dan urutan. Ini bukan pembacaan file video asli. Pembacaan, penulisan, dan verifikasi codec dilakukan pada praktikum.

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: Indeks frame (mulai 0; dibatasi jumlah) = 12 · FPS simulasi = 24 · Jumlah frame buatan = 48 · Kecepatan translasi (piksel/detik) = 12

Frame buatan indeks 12/47; waktu sumber simulasi 12/24 = 0.500 dtk; durasi 48/24 = 2.000 dtk.

Seluruh frame dihasilkan dari satu gambar diam, bukan decoding video atau deteksi gerakan. FPS 24; translasi 12 piksel/detik, offset 6 piksel (dibulatkan); bagian di luar sumber diisi gelap. Garis bawah: progres waktu, penanda jingga = frame saat ini, takik putih tiap detik. Timestamp frame terakhir 1.958 dtk; durasi mencakup satu interval tambahan hingga akhir frame terakhir.

▼ **Angka dan parameter hasil perhitungan ilustrasi**


```
{
  "frame": 12,
  "fps": 24,
  "count": 48,
  "translation": 12,
  "timestamp": 0.5,
  "duration": 2,
```

```
  "lastTimestamp": 1.9583333333333333,
  "offsetX": 6,
  "generatedFromStill": 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

OpenCV Python Tutorial #3 - Cameras and VideoCapture — Tech With Tim.

Fokus saat menonton: Ikuti pembacaan frame dan loop VideoCapture. Pahami FPS, resize per-frame serta pelepasan resource.

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 Videos
- OpenCV — VideoCapture dan VideoWriter
- FFmpeg — dokumentasi resmi opsi perintah
- IPython — Video display

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 sebelum impor, lalu unggah satu klip MP4/AVI/MOV/WebM berizin dengan durasi maksimal 20 detik, ukuran maksimal 50 MiB, dan sisi maksimal 1920 piksel. Praktikum membaca berkas video, bukan webcam. Sel inspeksi memperlihatkan frame pertama, ukuran sumber, FPS metadata, FPS yang dipakai, dan status estimasinya; informasi ini menjadi dasar membaca keluaran berikutnya.

Pada sel pengolahan, frame dibaca berurutan, diperkecil secara proporsional, diberi nomor dan waktu relatif, kemudian ditulis ke berkas kerja. Indeks waktu dimulai dari nol meskipun nomor tampilan dimulai dari satu. Kode membatasi jumlah frame dan waktu kerja, sehingga perhatikan alasan berhenti serta jumlah frame yang benar-benar ditulis. Cuplikan pertama membantu memeriksa warna dan keterbacaan anotasi sebelum melihat video lengkap.

Sel transkode memakai FFmpeg untuk membuat versi H.264 yang ditampilkan dalam notebook. Setelah pemeriksaan pembukaan ulang dan ukuran frame berhasil, putar player untuk melihat hasil sebenarnya. Jika player tidak berjalan, baca pesan transkode dan jangan menganggap ekstensi MP4 sudah menjamin kompatibilitas. Video keluaran tidak memuat audio; apabila FPS pengganti dipakai, waktu dan durasi turunannya tetap dibaca sebagai estimasi. Berkas pada runtime bersifat sementara, sehingga simpan hasil yang ingin dipertahankan sebelum sesi berakhir.

Sel 1 — 0. Setup baseline OpenCV 4.x

Jalankan sebelum import cv2 untuk menyamakan baseline dengan modul sebelumnya. Jika cv2 sudah diimport dan versi berubah, restart runtime lalu jalankan dari awal. Tidak perlu menaikkan NumPy secara terpisah.

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

Sel 2 — 1. Import dan batas demo

Gunakan klip maksimum 20 detik, 50 MB, maksimal 1920×1920 piksel. FFmpeg diperiksa pada sel transkode.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import math, time, pathlib, shutil, subprocess
from google.colab import files
from IPython.display import display, Video
MAX_SECONDS = 20
MAX_BYTES = 50 * 1024 * 1024
```

```
MAX_FRAMES = 1200
MAX_WALL_SECONDS = 60
```

Sel 3 — 2. Unggah dan inspeksi

Unggah tepat satu MP4/AVI/MOV/WebM pendek berizin. Durasi nominal yang tidak diketahui tetap dibatasi loop pada sel berikutnya.

```
uploaded = files.upload()
if len(uploaded) != 1:
    raise ValueError('Unggah tepat satu video pendek.')
nama, data = next(iter(uploaded.items()))
if len(data) > MAX_BYTES or not nama.lower().endswith(('.mp4',
'.avi', '.mov', '.webm')):
    raise ValueError('Video harus MP4/AVI/MOV/WebM, maksimal 50
MB.')
input_path = pathlib.Path('input_clip' +
pathlib.Path(nama).suffix.lower())
input_path.write_bytes(data)
cap = cv2.VideoCapture(str(input_path))
try:
    if not cap.isOpened():
        raise RuntimeError('Video tidak dapat dibuka.')
    fps_meta = float(cap.get(cv2.CAP_PROP_FPS))
    count_meta = float(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    fps_estimate = not (math.isfinite(fps_meta) and 1 <= fps_meta <=
60)
    fps = 25.0 if fps_estimate else fps_meta
    ok, first = cap.read()
    if not ok or first is None:
        raise RuntimeError('Frame pertama tidak dapat didekode.')
    in_h, in_w = first.shape[:2]
    if max(in_h, in_w) > 1920:
        raise ValueError('Resolusi input maksimal 1920 piksel per
sisi untuk demo.')
    duration_nominal = (count_meta / fps_meta if
math.isfinite(count_meta)
                        and count_meta > 0 and
math.isfinite(fps_meta)
                        and fps_meta > 0 else None)
    if duration_nominal is not None and duration_nominal >
MAX_SECONDS + 0.1:
```

```
        raise ValueError('Potong klip menjadi maksimal 20 detik
sebelum unggah.')
finally:
    cap.release()
print('Resolusi input:', (in_w, in_h))
print('FPS metadata:', fps_meta, '| FPS dipakai:', fps)
print('FPS fallback/estimasi:', fps_estimate)
print('Durasi nominal metadata:', duration_nominal)
plt.figure(figsize=(8, 5))
plt.imshow(cv2.cvtColor(first, cv2.COLOR_BGR2RGB))
plt.axis('off'); plt.show()
```

Sel 4 — 3. Resize, teks, dan simpan video

Timestamp berasal dari indeks/FPS. Pemrosesan dibatasi 20 detik nominal, 1200 frame, dan 60 detik kerja; output ini tidak menyimpan audio.

```
scale = min(1.0, 640 / in_w, 640 / in_h)
out_w = max(2, int(in_w * scale) // 2 * 2)
out_h = max(2, int(in_h * scale) // 2 * 2)
raw_path = pathlib.Path('video_anotasi_mp4v.mp4')
cap = cv2.VideoCapture(str(input_path))
writer = None
processed = 0
preview = None
frame_limit = min(MAX_FRAMES, max(1, int(math.floor(MAX_SECONDS *
fps))))
start = time.monotonic()
stop_reason = 'batas frame/durasi nominal'
try:
    if not cap.isOpened():
        raise RuntimeError('Video gagal dibuka untuk pemrosesan.')
    writer = cv2.VideoWriter(str(raw_path),
cv2.VideoWriter_fourcc(*'mp4v'),
                             fps, (out_w, out_h))
    if not writer.isOpened():
        raise RuntimeError('VideoWriter tidak tersedia untuk codec
mp4v.')
    while processed < frame_limit:
        if time.monotonic() - start > MAX_WALL_SECONDS:
            stop_reason = 'batas waktu kerja; hasil hanya cuplikan'
            break
```

```
        ok, frame = cap.read()
        if not ok or frame is None:
            stop_reason = 'akhir input atau read/decode gagal'
            break
        frame = cv2.resize(frame, (out_w, out_h))
        label = f'Frame {processed+1} | t={processed/fps:.2f}s'
        if fps_estimate:
            label += ' (est.)'
        font_scale = max(0.25, min(0.6, out_w / 700))
        text_size, baseline = cv2.getTextSize(label,
cv2.FONT_HERSHEY_SIMPLEX, font_scale, 1)
        cv2.rectangle(frame, (0, 0), (min(out_w-1, text_size[0]+12),
                      min(out_h-1, text_size[1]+baseline+12)), (0,
0, 0), -1)
        cv2.putText(frame, label, (6, text_size[1]+6),
cv2.FONT_HERSHEY_SIMPLEX,
                    font_scale, (255, 255, 255), 1, cv2.LINE_AA)
        writer.write(frame)
        if preview is None:
            preview = frame.copy()
        processed += 1
finally:
    cap.release()
    if writer is not None:
        writer.release()
if processed == 0 or not raw_path.exists() or
raw_path.stat().st_size == 0:
    raise RuntimeError('Tidak ada video hasil yang valid.')
print('Frame ditulis:', processed, '| ukuran:', (out_w, out_h))
print('Durasi keluaran nominal:', processed/fps, 'detik; estimasi:',
fps_estimate)
print('Alasan berhenti:', stop_reason)
plt.figure(figsize=(8, 5))
plt.imshow(cv2.cvtColor(preview, cv2.COLOR_BGR2RGB))
plt.axis('off'); plt.show()
```

Sel 5 — 4. Transkode H.264 untuk browser

Gunakan FFmpeg yang tersedia di runtime Colab. Sel ini tidak memasang paket lokal dan tidak mengubah NumPy.

```
ffmpeg = shutil.which('ffmpeg')
if ffmpeg is None:
    raise RuntimeError('FFmpeg tidak tersedia di runtime; gunakan
runtime Colab standar dan jalankan ulang.')
browser_path = pathlib.Path('video_anotasi_browser.mp4')
command = [ffmpeg, '-y', '-i', str(raw_path), '-an', '-c:v',
'libx264',
           '-preset', 'veryfast', '-crf', '23', '-pix_fmt',
'yuv420p',
           '-movflags', '+faststart', str(browser_path)]
result = subprocess.run(command, capture_output=True, text=True,
timeout=120)
if result.returncode != 0:
    raise RuntimeError('Transkode gagal: ' + result.stderr[-1500:])
if not browser_path.exists() or browser_path.stat().st_size == 0:
    raise RuntimeError('Berkas H.264 tidak terbentuk.')
print('Ukuran video browser (MB):',
round(browser_path.stat().st_size / (1024*1024), 2))
```

Sel 6 — 5. Verifikasi dan tampilkan hasil

Player ditanam dalam output notebook. Jalankan play dan pastikan tulisan terbaca serta nomor frame berubah secara berurutan.

```
check = cv2.VideoCapture(str(browser_path))
try:
    if not check.isOpened():
        raise RuntimeError('Video hasil tidak dapat dibuka ulang.')
    ok, check_frame = check.read()
    if not ok or check_frame is None:
        raise RuntimeError('Frame video hasil gagal didekode.')
    if check_frame.shape[:2] != (out_h, out_w):
        raise RuntimeError('Ukuran video hasil tidak sesuai.')
    print('Frame pertama output berhasil didekode.')
    print('FPS output:', check.get(cv2.CAP_PROP_FPS))
    print('Jumlah frame output menurut metadata:',
check.get(cv2.CAP_PROP_FRAME_COUNT))
finally:
    check.release()
if browser_path.stat().st_size > 20 * 1024 * 1024:
    raise RuntimeError('Output terlalu besar untuk embed demo;
```

```
gunakan klip yang lebih pendek.')
display(Video(str(browser_path), embed=True, mimetype='video/mp4'))
```

Jika terjadi error

cap.isOpened() bernilai False.

Periksa berkas dan format; gunakan MP4 pendek yang dapat diputar, bukan hanya mengganti ekstensi.

Metadata FPS nol/NaN atau melampaui batas demo.

Kode memakai 25 FPS sebagai estimasi. Jika pengukuran waktu membutuhkan ketelitian, gunakan klip dengan metadata yang jelas dan catat penggunaan fallback.

VideoWriter gagal atau video kosong.

Periksa isOpened(), frame count, ukuran genap, dan kesesuaian setiap frame dengan ukuran writer.

MP4 tidak diputar browser.

Jalankan transkode H.264/yuv420p dan cek pesan FFmpeg; ekstensi MP4 saja tidak menentukan codec.

Video hasil tanpa suara.

Ini batas jalur pengolahan frame OpenCV dan opsi -an, bukan kerusakan tak terduga. Jelaskan bahwa audio tidak dipertahankan.

> 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.
