PENGOLAHAN CITRA · MATERI 03

Color Space dan Manipulasi Warna

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 sebuah foto berisi gelas merah, daun hijau, dan buku biru di atas meja putih. Bagi manusia, mengenali ketiga benda itu terasa mudah. Komputer mula-mula hanya menerima deretan angka pada setiap lokasi pixel. Sebelum menentukan bagian mana yang tampak merah, kita perlu mengetahui arti angka tersebut: apakah urutannya merah–hijau–biru, biru–hijau–merah, atau komponen ruang warna lain? Kesalahan membaca urutan dapat membuat gelas merah terlihat biru, padahal isi file dan posisi bendanya tidak berubah.

Ruang warna adalah cara menyusun keterangan warna itu. RGB dan BGR menyimpan tiga komponen dasar dengan urutan berbeda. Grayscale merangkum informasi menjadi satu nilai intensitas sehingga bentuk dan perbedaan terang masih terlihat, tetapi pembeda warna berkurang. HSV menyatakan warna melalui hue, saturation, dan value. Pemisahan ini membantu kita membuat aturan seperti memilih pixel dengan keluarga warna merah yang cukup jenuh dan tidak terlalu gelap. Aturan tersebut lebih mudah ditelusuri daripada sekadar melihat apakah komponen merah bernilai besar.

Alur materi bergerak dari membaca representasi menuju membuat keputusan. Pertama, kita memastikan foto tampil dengan urutan komponen yang benar. Berikutnya, kita membaca channel sebagai peta angka, memahami rentang HSV, lalu membentuk mask yang menandai pixel terpilih. Mask itu dipakai untuk menampilkan kembali bagian foto yang memenuhi aturan warna. Hasilnya perlu dibaca hati-hati: meja yang memantulkan warna merah dapat ikut terpilih, sementara bagian gelas dalam bayangan dapat hilang. Dengan demikian, keberhasilan bukan berarti komputer sudah mengenali sebuah gelas. Kita sedang mempelajari seleksi warna beserta alasan sebuah pixel masuk atau keluar dari pilihan.

---

2. Pahami konsep secara berurutan

2.1 Warna tersimpan sebagai komponen, bukan nama benda

Citra warna pada contoh berupa array tiga channel dengan nilai uint8, yaitu bilangan bulat 0–255. Setiap lokasi menyimpan tiga komponen. Dalam BGR, urutannya biru, hijau, merah; dalam

RGB, merah, hijau, biru. Pixel dengan komponen merah paling kuat mempunyai susunan angka berbeda pada kedua representasi itu, tetapi tetap menyatakan warna yang sama jika ditafsirkan dengan benar. BGR dan RGB bukan dua tingkat kualitas gambar.

Array `bgr` yang disiapkan OpenCV perlu dikonversi menjadi `rgb` sebelum ditampilkan sebagai foto warna melalui Matplotlib. Konversi ini menukar urutan komponen, bukan memindahkan pixel atau menambah detail. Simpan sumber dan hasil konversi dalam variabel berbeda supaya pemeriksaan warna tidak tercampur dengan perubahan data. Indeks array tetap mengikuti `[y, x]`: baris dahulu, kemudian kolom, meskipun koordinat biasa ditulis sebagai pasangan x dan y.

2.2 Grayscale menyederhanakan informasi warna

Konversi `cv2.COLOR_BGR2GRAY` menghasilkan satu nilai intensitas per pixel dengan kontribusi berbobot dari komponen warna, bukan sekadar mengambil salah satu channel. Penyederhanaan ini berguna saat perhatian tertuju pada pola terang, bentuk, atau tepi. Akan tetapi, dua permukaan dengan warna berbeda dapat menghasilkan intensitas yang mirip. Batas antara benda merah dan latar hijau mungkin jelas pada foto warna, tetapi kurang terlihat setelah konversi grayscale.

Karena informasi warna dirangkum, grayscale tidak dapat diperlakukan sebagai pengganti setara untuk seleksi berdasarkan hue. Pada foto grayscale yang diperluas menjadi BGR tiga channel, ketiga komponennya sama; penambahan jumlah channel tidak mengembalikan warna yang sudah tidak tersedia. Bedakan bentuk penyimpanan tiga channel dari keberadaan informasi warna yang sungguh berbeda di antara channel tersebut.

2.3 Channel adalah peta nilai pada lokasi yang sama

Pemisahan dengan `cv2.split` menghasilkan `B`, `G`, dan `R`. Ketiganya ditampilkan sebagai peta grayscale: terang berarti nilai komponen besar, gelap berarti kecil. Panel R bukan foto yang hanya menyisakan benda merah. Meja putih dapat tampak terang di panel R karena warna putih memiliki komponen merah, hijau, dan biru yang sama-sama tinggi. Sebaliknya, permukaan merah yang berada dalam bayangan dapat tampak gelap.

Prinsip peta nilai juga berlaku untuk `H`, `S`, dan `V`. Setiap panel menerangkan satu komponen pada lokasi foto yang sama. Terang pada panel H berarti angka hue tinggi, bukan warna lebih cerah atau lebih merah. Rentang tampilan perlu dinyatakan supaya pembacaan konsisten: H menggunakan maksimum 179, sedangkan channel lain pada contoh menggunakan maksimum 255. Array HSV tidak ditampilkan langsung sebagai foto RGB karena arti ketiga komponennya berbeda.

2.4 HSV memisahkan keluarga warna dari kejenuhan dan value

Hue atau H menyatakan posisi pada lingkaran warna. Saturation atau S menyatakan kejenuhan; nilai rendah menunjukkan warna yang mendekati abu-abu. Value atau V menggambarkan tingkat terang dalam representasi HSV. Pada konversi standar `cv2.COLOR_BGR2HSV` untuk input uint8, H berada pada 0–179, sedangkan S dan V berada pada 0–255. Ini bukan skala hue dalam derajat 0–360; secara pendekatan, satu langkah H mewakili dua derajat. Karena itu angka batas dari aplikasi lain tidak boleh disalin tanpa memeriksa skalanya.

Aturan warna memakai ketiga komponen bersama-sama. Contoh hijau menggunakan H 35–85, S minimal 60, dan V minimal 40. Batas tersebut merupakan titik awal, bukan definisi universal semua benda hijau. Saat S sangat rendah, hue kurang bermakna sebagai petunjuk warna. Saat V rendah, permukaan berada di daerah gelap dan pemisahan warna bisa sulit. Menetapkan batas S/V menolak sebagian gangguan, tetapi juga dapat membuang warna pucat dan bagian objek yang berbayang.

2.5 Mask menyatakan keputusan, ekstraksi menampilkan pilihannya

`cv2.inRange` menguji apakah setiap pixel memenuhi seluruh batas bawah dan atas H, S, serta V. Hasilnya adalah mask satu channel: nilai 255 berarti terpilih dan 0 berarti tidak terpilih. Jadi mask merupakan peta keputusan, bukan foto berwarna. Fungsi `cv2.bitwise_and` dengan argumen mask kemudian mempertahankan nilai BGR sumber hanya pada lokasi terpilih. Area lainnya menjadi hitam pada hasil, meskipun latar foto aslinya bukan hitam.

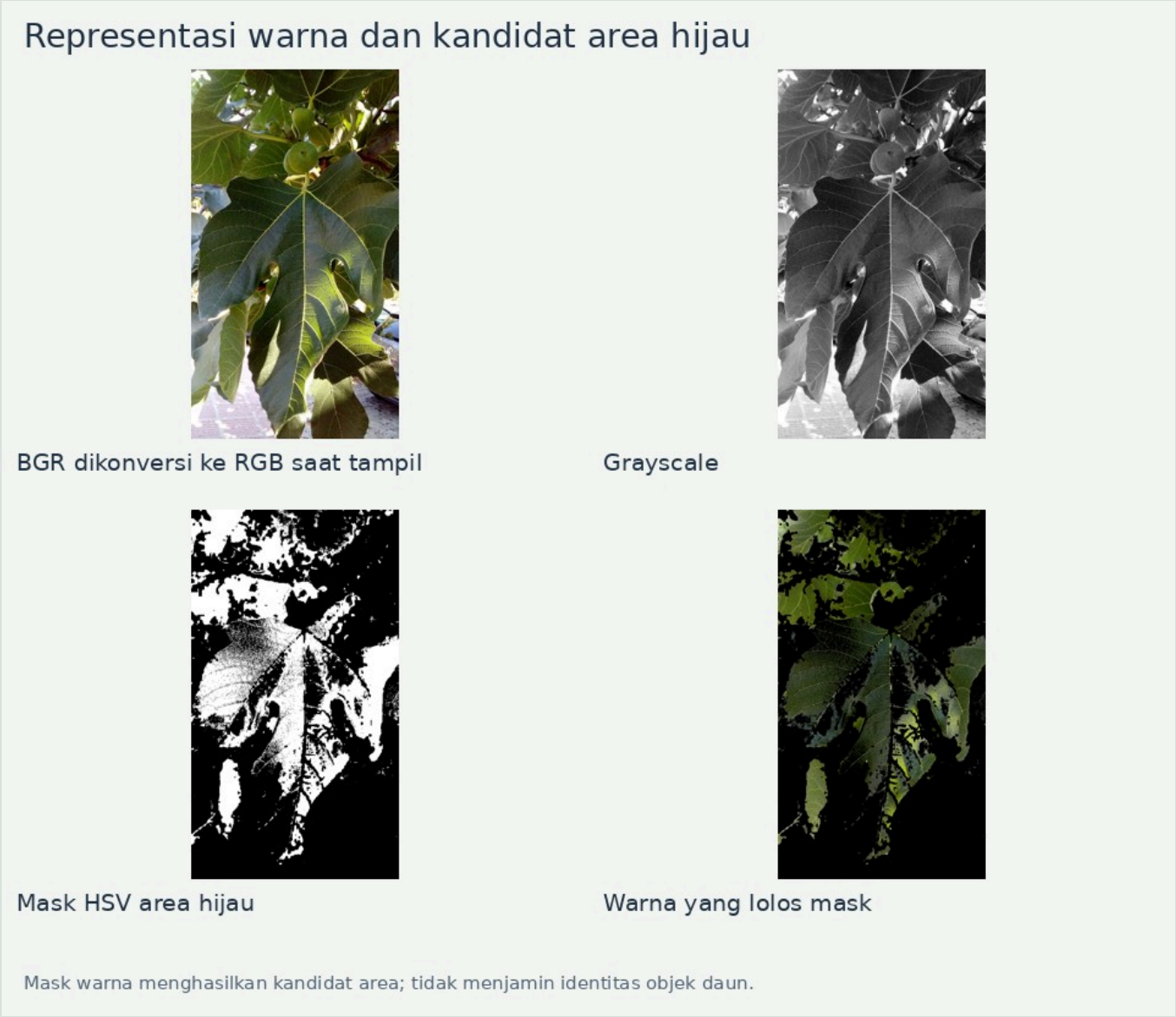
Merah memerlukan perhatian khusus karena lingkaran hue bertemu kembali di ujung rentang. Contoh memakai dua interval H, yaitu 0–10 dan 170–179, keduanya dengan S minimal 80 serta V minimal 50. Dua hasil seleksi digabung melalui `cv2.bitwise_or`. Menggunakan satu interval saja dapat menghilangkan sebagian merah. Jumlah pixel putih dan persentasenya menerangkan luas pilihan dalam gambar, bukan jumlah benda, tingkat keyakinan, atau akurasi pengenalan.

2.6 Pencahayaan membatasi kesimpulan dari warna

Cahaya, bayangan, pantulan, dan pengaturan kamera dapat mengubah nilai HSV permukaan yang sama. Kalibrasi berarti memeriksa beberapa pixel objek dan latar, lalu menyesuaikan interval berdasarkan nilai yang terlihat. Titik tengah gambar pada contoh hanya menunjukkan cara membaca array; titik itu tidak otomatis berada pada objek. Bandingkan bagian terang dan berbayang agar alasan perubahan batas jelas.

Seleksi warna tidak mengetahui makna benda. Daun hijau dan sampul buku hijau dapat masuk mask yang sama, sementara bagian daun kecokelatan tidak terpilih. Latar yang ikut terambil dan bagian sasaran yang hilang perlu dijelaskan terpisah. Mask kosong juga dapat benar ketika

warna tersebut memang tidak ada. Kesimpulan yang tepat menyebut area memenuhi rentang warna, bukan menyatakan sistem telah mengenali identitas objek.



Mask warna menghasilkan kandidat area; tidak menjamin identitas objek daun.

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

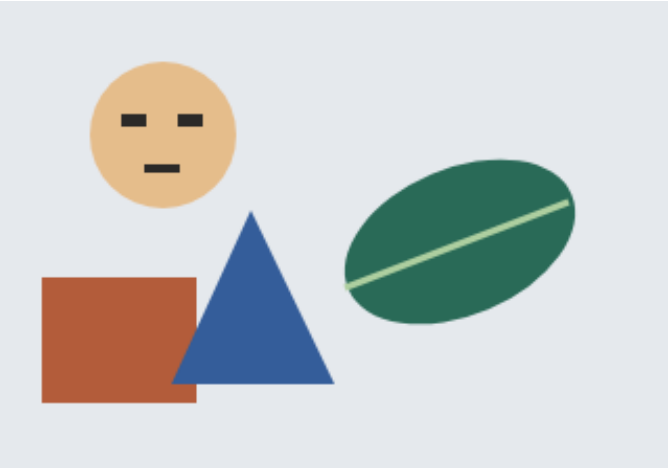
Ruang warna RGB/HSV dan mask warna

Konversi RGB → HSV numerik (H 0–179, S/V 0–255). Channel R/G/B dan H/S/V adalah peta skalar, bukan foto berwarna atau pengenalan objek. Mask memakai jarak hue melingkar, S

minimum, dan V minimum tetap 50.

> Mask memilih piksel berdasarkan rentang warna, bukan mengenali nama objek. Hue pada ilustrasi ditampilkan dalam skala 0–179 seperti contoh OpenCV; pembulatan perhitungan browser dapat berbeda. Warna serupa pada latar bisa ikut dipilih.

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: Tampilan = mask · Hue target H (0=merah, 60=hijau, 120=biru) = 0 · Toleransi hue melingkar = 10 · S minimum = 80

4280 dari 71680 pixel memenuhi mask (5.97%).

H target 0 ± 10 (melingkar: merah melintasi 179→0); S ≥ 80; V ≥ 50.

HSV pixel tengah: [105, 9, 237]. H divisualisasi skalar dengan 0–179 dipetakan ke 0–255.

Konversi float JavaScript dengan Math.round; batas pembulatan dapat berbeda ±1 dari implementasi integer OpenCV. Ini bukan klaim bit-identik.

Jumlah adalah pixel yang lolos ambang, bukan akurasi atau jumlah objek. Engine menghitung semua pixel RGB dan mempertahankan alpha buffer. Pada menu upload halaman ini, transparansi lebih dahulu diratakan di atas putih.

▼ **Angka dan parameter hasil perhitungan ilustrasi**


```
{
  "selectedPixels": 4280,
  "totalPixels": 71680,
  "selectedPercent": 5.970982142857143,
  "centerH": 105,
  "centerS": 9,
  "centerV": 237,
  "hue": 0,
  "tolerance": 10,
  "smin": 80,
  "vmin": 50
}
```


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 #5 - Colors and Color Detection — Tech With Tim.

Fokus saat menonton: Amati perbedaan RGB/BGR/HSV serta cara mask memilih rentang warna.

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 — Changing Colorspaces
- OpenCV — Basic Operations on Images
- Matplotlib — imshow

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 sel import dan upload terlebih dahulu agar `bgr` tersedia dalam runtime yang sama. Sel upload menerima satu JPEG/PNG uint8 tanpa alpha, maksimum 10 MiB dan 12 juta pixel, dengan sisi minimal 32 pixel. Sel konversi membentuk `rgb`, `gray`, dan `hsv`, kemudian mencetak nilai pixel tengah melalui `[y, x]`. Baca nilai tersebut sebagai contoh koordinat, bukan otomatis sebagai sampel benda sasaran. Foto warna ditampilkan dari `rgb`; hasil `cv2.split` dibaca sebagai peta nilai B/G/R dan H/S/V.

Pada sel seleksi warna, kamus `rentang` memuat dua interval merah serta masing-masing satu interval hijau dan biru. Perulangan membuat `bagian` dengan `cv2.inRange`, menggabungkannya ke `mask` melalui `cv2.bitwise_or`, lalu menyimpan hasil dalam `masks[warna]`. Baca setiap baris panel dari foto sumber, mask, hingga ekstraksi `hasil`. Angka `jumlah` dan persentase menunjukkan banyaknya pixel yang terpilih, bukan banyaknya objek. Jika mask kosong atau terlalu luas, bandingkan `hsv[y, x]` pada beberapa lokasi objek dan latar, kemudian sesuaikan batas H/S/V dengan tetap mempertahankan dua interval merah.

Periksa apakah bagian berbayang tertolak karena V rendah, permukaan pucat tertolak karena S rendah, atau latar berwarna serupa ikut terambil. Penurunan batas S/V dapat memasukkan bagian sasaran sekaligus menambah pilihan yang tidak diinginkan. Setelah perubahan `rentang`, jalankan kembali sel seleksi untuk melihat akibatnya pada foto yang sama. Bila runtime direset atau foto diganti, jalankan ulang sel berurutan agar mask tidak berasal dari array foto sebelumnya.

Sel 1 — 1. Import library

Ketik di notebook Colab baru pada bagian awal notebook. Modul ini mandiri; tidak menggunakan variabel dari modul lain.

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

Sel 2 — 2. Upload satu foto

Pilih satu foto JPEG/PNG uint8, maksimum 10 MiB dan 12 juta pixel. Pembacaan mempertahankan channel asli untuk validasi, lalu menyiapkan BGR tiga channel. PNG transparan tidak dipakai agar tidak ada background tersembunyi.

```
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()
```

Sel 3 — 3. Konversi dan inspeksi ruang warna

RGB dipakai untuk tampilan, grayscale untuk intensitas, dan HSV untuk aturan warna. Nilai pixel tengah hanya contoh pembacaan; pilih koordinat objek saat kalibrasi.

```
rgb = cv2.cvtColor(bgr, cv2.COLOR_BGR2RGB)
gray = cv2.cvtColor(bgr, cv2.COLOR_BGR2GRAY)
hsv = cv2.cvtColor(bgr, cv2.COLOR_BGR2HSV)
y, x = bgr.shape[0] // 2, bgr.shape[1] // 2
print("Koordinat (x,y):", (x, y))
print("BGR:", bgr[y, x].tolist(), "RGB:", rgb[y, x].tolist())
print("Gray:", int(gray[y, x]), "HSV:", hsv[y, x].tolist())
fig, ax = plt.subplots(1, 2, figsize=(12, 5))
ax[0].imshow(rgb); ax[0].set_title("RGB: warna asli")
ax[1].imshow(gray, cmap="gray", vmin=0, vmax=255);
ax[1].set_title("Grayscale")
for a in ax: a.axis("off")
plt.tight_layout(); plt.show()
```

Sel 4 — 4. Pisahkan BGR dan HSV

Channel ditampilkan sebagai scalar grayscale; panel R bukan hasil deteksi objek merah. H menggunakan rentang 0–179.

```
B, G, R = cv2.split(bgr)
H, S, V = cv2.split(hsv)
fig, axes = plt.subplots(2, 3, figsize=(14, 8))
for ax, judul, channel, maksimum in zip(
        axes.flat, ["B", "G", "R", "H", "S", "V"],
        [B, G, R, H, S, V], [255, 255, 255, 179, 255, 255]):
    ax.imshow(channel, cmap="gray", vmin=0, vmax=maksimum)
    ax.set_title(f"Channel {judul} (0–{maksimum})")
    ax.axis("off")
plt.tight_layout(); plt.show()
```

Sel 5 — 5. Deteksi merah, hijau, dan biru

Contoh lengkap memproses tiga warna sekaligus; fokuskan pembahasan awal pada satu warna objek. Parameter merupakan baseline yang boleh dikalibrasi, bukan jaminan semua foto.

```
rentang = {
    "merah": [((0, 80, 50), (10, 255, 255)),
              ((170, 80, 50), (179, 255, 255))],
    "hijau": [((35, 60, 40), (85, 255, 255))],
    "biru": [((90, 80, 50), (130, 255, 255))]
```

```
}
masks = {}
fig, axes = plt.subplots(3, 3, figsize=(14, 12))
for baris, (warna, interval) in enumerate(rentang.items()):
    mask = np.zeros(hsv.shape[:2], dtype=np.uint8)
    for bawah, atas in interval:
        bagian = cv2.inRange(hsv, np.array(bawah, np.uint8),
np.array(atas, np.uint8))
        mask = cv2.bitwise_or(mask, bagian)
    masks[warna] = mask
    hasil = cv2.bitwise_and(bgr, bgr, mask=mask)
    jumlah = int(np.count_nonzero(mask))
    print(warna, "batas:", interval, "pixel terpilih:", jumlah,
          "persen:", round(100 * jumlah / mask.size, 2))
    axes[baris, 0].imshow(rgb); axes[baris, 0].set_title("Foto
sumber")
    axes[baris, 1].imshow(mask, cmap="gray", vmin=0, vmax=255)
    axes[baris, 1].set_title(f"Mask {warna}")
    axes[baris, 2].imshow(cv2.cvtColor(hasil, cv2.COLOR_BGR2RGB))
    axes[baris, 2].set_title(f"Ekstraksi {warna}")
    for ax in axes[baris]: ax.axis("off")
    if jumlah == 0:
        print(f"Tidak ada pixel memenuhi rentang {warna}; cek objek
dan parameter.")
plt.tight_layout(); plt.show()
```

Jika terjadi error

Mask kosong meski objek terlihat.

Periksa urutan konversi BGR2HSV, range H 0–179, serta batas S/V; sampel pixel objek dan sesuaikan parameter.

Sebagian objek merah hilang.

Gunakan kedua interval hue merah dan gabungkan mask dengan bitwise_or.

HSV ditampilkan sebagai foto warna yang aneh.

Tampilkan foto dari RGB; pisahkan H/S/V dan gunakan peta scalar untuk memeriksa HSV.

Latar berwarna sama ikut terdeteksi.

Akui batas deteksi warna; ubah latar/pencahayaan atau tambahkan pembatas lokasi dalam pengembangan, bukan mengklaim pengenal objek.

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.
