PENGOLAHAN CITRA · MATERI 06

# Edge Detection, Contour dan Shape Detection

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

Letakkan tutup botol dan potongan kertas berbentuk persegi di atas meja polos, lalu bayangkan keduanya difoto dari atas. Kita mudah membedakan bentuknya karena melihat batas yang memisahkan benda dari meja. Komputer hanya menerima susunan piksel, yaitu titik-titik penyusun gambar beserta nilai warnanya. Agar bentuk dapat diperiksa, informasi gambar perlu diolah menjadi batas dan ukuran yang lebih sederhana.

Langkah awalnya adalah mencari **tepi**, tempat nilai terang-gelap berubah cukup kuat. Namun, tulisan pada tutup botol dan bayangan kertas juga dapat menghasilkan tepi. Karena itu, kumpulan garis pada gambar belum tentu sama dengan batas luar benda. Kita membutuhkan **mask**, gambar hitam-putih yang membedakan daerah objek dari latarnya. Dari mask tersebut, kita dapat mengambil **kontur**, yaitu urutan titik yang mengikuti batas daerah objek.

Setelah batasnya tersedia, kita dapat menghitung luas daerah yang dilingkupinya, panjang batas, serta kotak pembatas. Kotak pembatas adalah persegi panjang yang mengurung objek, bukan bentuk objek itu sendiri. Selanjutnya, batas yang berliku dapat disederhanakan menjadi beberapa ruas garis untuk membantu memperkirakan bentuk. Empat sudut saja belum cukup untuk menyatakan persegi; panjang sisi dan sudutnya juga perlu dipertimbangkan.

Bab ini menghubungkan langkah-langkah tersebut tanpa mencampur fungsi masing-masing. Canny dipakai untuk mempelajari tepi, sedangkan kontur siluet diperoleh dari mask yang telah diperiksa. Dari sana kita membaca ukuran dalam piksel dan memahami alasan munculnya dugaan bentuk. Tujuannya bukan membuat komputer langsung mengenali semua benda, melainkan memahami bagaimana keputusan geometris dibangun serta mengapa foto miring, bayangan, atau latar yang kurang sesuai dapat mengubah hasilnya.

---

## 2. Pahami konsep secara berurutan

### 2.1 Perubahan terang-gelap belum merupakan batas objek

Gambar grayscale menyimpan tingkat terang pada setiap piksel. Pada permukaan yang relatif seragam, nilai piksel bertetangga cenderung berdekatan. Di pertemuan benda gelap dan latar

terang, perbedaannya dapat menjadi besar. Perubahan seperti ini menjadi dasar pencarian tepi. Akan tetapi, komputer belum mengetahui penyebab perubahan tersebut. Urat daun, tulisan, pantulan, dan bayangan juga bisa menimbulkan garis yang kuat meskipun bukan batas luar benda.

Canny menghasilkan peta tepi berupa garis tipis melalui pengolahan perubahan intensitas dan pemilihan kandidat tepi. Gaussian blur diterapkan lebih dahulu untuk mengurangi gangguan kecil atau noise. Penghalusan ini membantu, tetapi juga dapat melemahkan detail yang memang ingin dipertahankan. Dua ambang Canny mempunyai peran berbeda: ambang atas menentukan kandidat kuat, sedangkan kandidat di antara kedua ambang dipertahankan jika terhubung dengan tepi kuat. Ambang rendah bukan perintah agar semua garis lemah diterima. Nilai 50 dan 150 pada kode hanyalah titik awal, bukan pasangan terbaik untuk semua foto.

## 2.2 Memisahkan daerah objek sebelum mengambil kontur

Jika tujuannya mengukur siluet, kita membutuhkan daerah objek yang terisi, bukan hanya garis-garis Canny. Mask biner menyatakan daerah itu dengan piksel putih dan latar dengan piksel hitam. Dalam praktikum, Otsu memilih ambang dari sebaran intensitas gambar. Pilihan tersebut tidak mengandung pemahaman tentang benda; jika terang-gelap objek dan latar saling bercampur, pemisahannya tetap dapat keliru.

Variabel `OBJEK_GELAP` mengatur polaritas, yaitu apakah bagian gelap atau bagian terang yang dijadikan daerah putih. Jika pilihan terbalik, latar dapat menjadi daerah utama yang terukur. Operasi opening membantu membuang bintik kecil, sedangkan closing membantu menutup celah kecil. Keduanya juga dapat menghapus bagian tipis atau menyatukan bagian berdekatan. Karena itu, mask sebelum dan sesudah pengolahan perlu dibaca sebagai hasil pemisahan objek, bukan sekadar gambar antara yang dilewati.

## 2.3 Dari mask menuju lintasan batas

Kontur menyimpan urutan titik batas daerah pada mask. `RETR_EXTERNAL` mengambil batas paling luar; lubang di dalam objek tidak diuraikan sebagai bagian analisis dasar ini. Misalnya, batas luar cincin saja tidak menjelaskan luas bahan cincin setelah lubangnya dikurangi. `CHAIN_APPROX_SIMPLE` mengurangi titik berulang pada segmen lurus agar representasinya lebih ringkas, tanpa mengubahnya menjadi pengenal bentuk.

Kode menyaring kontur yang terlalu kecil melalui `MIN_AREA` dan melewati kontur yang menyentuh bingkai gambar. Bintik kecil mungkin hanya gangguan, tetapi mungkin pula benda kecil yang sebenarnya ingin diamati. Benda terpotong tidak mempunyai siluet lengkap di dalam foto. Penanganan ini membuat batas contoh lebih jelas, sekaligus menjelaskan mengapa jumlah kontur valid dapat lebih sedikit daripada jumlah daerah yang awalnya ditemukan. Objek yang saling menempel juga dapat terbaca sebagai satu daerah, bukan dua benda terpisah.

## 2.4 Membaca luas, perimeter, dan kotak pembatas

`contourArea` menghitung luas geometris yang dibatasi kontur, sedangkan `arcLength` menghitung panjang lintasannya atau perimeter. Luas dinyatakan dalam piksel kuadrat dan perimeter dalam piksel. Nilainya tidak otomatis menjadi ukuran fisik. Benda yang sama dapat tampak lebih besar saat kamera mendekat. Pengukuran sentimeter memerlukan skala yang diketahui serta penanganan pengaruh perspektif.

Luas kontur juga tidak persis sama dengan jumlah piksel putih. Yang pertama memakai geometri batas, sedangkan yang kedua menghitung titik raster yang terisi. Sementara itu, `boundingRect` memberikan posisi kiri atas, lebar, dan tinggi kotak sejajar sumbu gambar. Sudut kotaknya dapat berisi latar, sehingga luas kotak bukan luas benda. Saat persegi diputar, kotak ini dapat membesar. `minAreaRect` menyediakan kotak yang boleh berotasi dan membantu memeriksa perbandingan sisi tanpa terlalu bergantung pada kemiringan di bidang gambar.

## 2.5 Menyederhanakan batas dan menduga bentuk

`approxPolyDP` mendekati kontur dengan poligon, yaitu bentuk yang dibatasi ruas-ruas garis. Toleransi `epsilon` menentukan seberapa jauh pendekatan boleh menyimpang dari kontur. Kode menggunakan sebagian kecil perimeter sebagai toleransi. Toleransi terlalu kecil mempertahankan lekukan dan gangguan; terlalu besar dapat menghilangkan sudut penting. Jadi, jumlah titik sudut atau vertex adalah hasil pendekatan dengan parameter tertentu, bukan sifat yang selalu tetap pada setiap foto.

Empat vertex dapat menggambarkan persegi, persegi panjang, atau segiempat lain. Kode mempertimbangkan kecembungan, kedekatan sudut terhadap siku-siku, dan rasio sisi kotak berotasi. Kecembungan berarti bentuk tidak memiliki lekukan ke dalam. Untuk kandidat lingkaran, kode juga memakai circularity, $4\pi A/P^2$, yang menggabungkan luas A dan perimeter P. Kontur bulat yang mulus cenderung mempunyai nilai mendekati satu. Gerigi batas dapat memperpanjang perimeter sehingga nilai ini berubah meskipun bendanya tetap bulat.

## 2.6 Menilai hasil sebagai dugaan geometris

Label bentuk pada contoh adalah heuristik: keputusan praktis berdasarkan beberapa ciri, bukan jaminan pengenalan benda. Lingkaran yang difoto miring dapat tampak lonjong; persegi dapat tampak seperti trapesium. Daun yang bertepi halus pun tidak otomatis merupakan lingkaran. Bentuk pada gambar harus dibedakan dari bentuk fisik benda dan kategori bendanya.

Urutan pemeriksaan yang masuk akal adalah foto sumber, mask, kontur, poligon pendekatan, lalu label dan kotak. Rekap luas, perimeter, vertex, dan circularity membantu menjelaskan keputusan. Jika mask salah, mengganti aturan label saja tidak memperbaiki akar masalah. Kontur yang diperlakukan tertutup untuk perhitungan juga bukan bukti bahwa seluruh benda

benar-benar terlihat. Membaca hasil dengan urutan ini membuat keberhasilan maupun kegagalan dapat dijelaskan dari prosesnya.




Panel geometri adalah ilustrasi sintetis; panel foto berasal dari foto nyata.

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

---

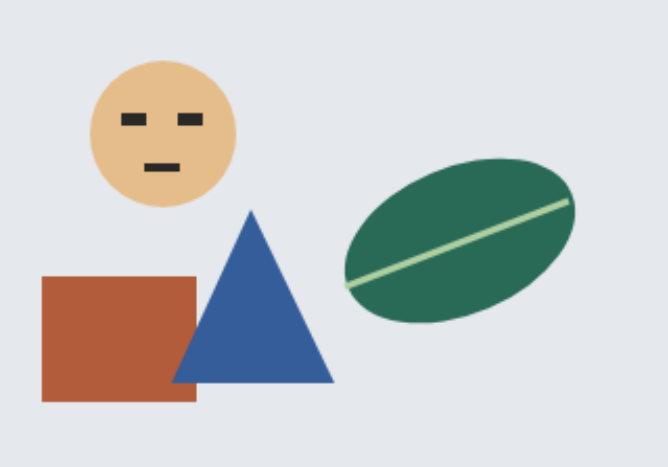
# 3. Ilustrasi interaktif sebelum praktikum

## Tepi Sobel dan komponen mask

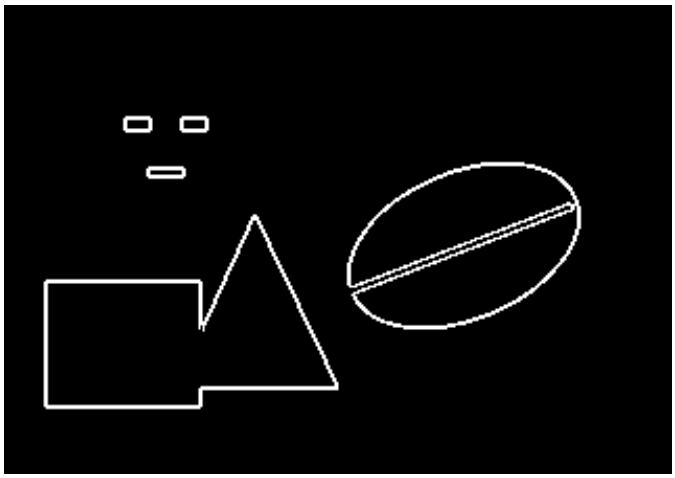
Hitung gradien Sobel nyata atau komponen terhubung dari mask ambang. Ini bukan Canny, findContours OpenCV, atau pengenalan bentuk poligon.

Operasi piksel dan pengukuran pada ilustrasi ini dihitung dari input. Perhatikan nama metodenya: ilustrasi Sobel atau komponen mask tidak sama dengan seluruh alur Canny, kontur OpenCV, atau pengenalan kategori benda.

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: Tampilan = sobel · Ambang (0–255) = 80 · Luas piksel minimum (mode mask) = 20 · Polaritas (mode mask) = light

Sobel: 1963 piksel tepi; ambang 80.

Magnitudo = min(255, √(Gx²+Gy²)/4); tepi jika > ambang. Bingkai 1 piksel diatur nol. Luas minimum dan polaritas hanya berlaku pada mode mask. Bukan Canny atau deteksi bentuk.

▼ **Angka dan parameter hasil perhitungan ilustrasi**


```
{
  "mode": "sobel",
  "threshold": 80,
  "edgePixels": 1963,
  "maxGradient": 163.3416664540925
}
```

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

**Real time Shape Detection using Contours [9] | OpenCV Python Tutorials for Beginners 2020** — Murtaza's Workshop - Robotics and AI.

**Fokus saat menonton:** Ikuti preprocessing, contour dan aproksimasi bentuk. Bedakan batas piksel, kontur dan label bentuk.

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 — Canny Edge Detection
- OpenCV — Contours: Getting Started
- OpenCV — Contour Features

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 sel impor sebelum sel unggah. Masukannya satu foto JPEG/PNG tanpa transparansi, maksimal 10 MiB dan 12 juta piksel, dengan sisi minimal 32 piksel. Foto objek terpisah pada latar polos memudahkan hubungan antara benda dan mask terbaca. Jalur ini menggunakan unggah foto, bukan webcam.

Sel Canny mengubah foto menjadi grayscale, melakukan blur, lalu menampilkan peta tepi. Sel berikutnya bekerja melalui jalur berbeda: Otsu membentuk mask, `OBJEK_GELAP` menentukan polaritas, dan pengolahan mask menyiapkan daerah untuk pencarian kontur. Baca mask sebelum menafsirkan jumlah kontur; kontur kecil dan kontur yang menyentuh bingkai dapat dilewati oleh kode.

Keluaran akhirnya berupa anotasi kontur, poligon pendekatan, kotak pembatas, dan rekap ukuran. Hubungkan setiap baris rekap dengan nomor pada gambar, lalu baca label sebagai dugaan berdasarkan geometri. Jika hasil kosong atau bentuknya tidak sesuai, periksa pemisahan objek terlebih dahulu sebelum mencurigai aturan bentuk. Ukuran yang tercetak tetap dalam piksel dan piksel kuadrat, bukan ukuran fisik benda.

## 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. Peta tepi Canny

Dua threshold Canny adalah parameter awal. Bandingkan tepi dengan foto sumber agar tekstur internal tidak dianggap batas objek.

```
gray = cv2.cvtColor(bgr, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (5, 5), 0)
LOW, HIGH = 50, 150
if not 0 <= LOW < HIGH:
    raise ValueError("Threshold Canny harus 0 <= LOW < HIGH.")
edges = cv2.Canny(blur, LOW, HIGH)
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
axes[0].imshow(cv2.cvtColor(bgr, cv2.COLOR_BGR2RGB));
axes[0].set_title("Foto sumber")
axes[1].imshow(edges, cmap="gray", vmin=0, vmax=255)
axes[1].set_title(f"Canny: {LOW}, {HIGH}")
for ax in axes: ax.axis("off")
plt.tight_layout(); plt.show()
```

## Sel 4 — 4. Mask dan kontur eksternal

Kontur dihitung dari mask siluet Otsu, bukan langsung dari tekstur Canny. Gunakan OBJEK_GELAP sesuai foto; morfologi kecil boleh dilewati bila merusak bentuk.

```
OBJEK_GELAP = True
mode = cv2.THRESH_BINARY_INV if OBJEK_GELAP else cv2.THRESH_BINARY
T_otsu, mask_awal = cv2.threshold(blur, 0, 255, mode |
cv2.THRESH_OTSU)
kernel = np.ones((3, 3), dtype=np.uint8)
mask = cv2.morphologyEx(mask_awal, cv2.MORPH_OPEN, kernel)
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
contours, hierarchy = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
MIN_AREA = max(50.0, 0.001 * mask.size)
kontur_valid = []
h, w = mask.shape
for cnt in contours:
    if cv2.contourArea(cnt) < MIN_AREA:
        continue
    x, y, bw, bh = cv2.boundingRect(cnt)
    if x <= 0 or y <= 0 or x + bw >= w or y + bh >= h:
        print("Dilewati: kontur menyentuh bingkai; cek polaritas
atau ambil foto lebih longgar.")
        continue
    kontur_valid.append(cnt)
kontur_valid.sort(key=lambda c: cv2.boundingRect(c)[0])
print("Threshold Otsu:", T_otsu, "MIN_AREA:", MIN_AREA,
      "kontur total:", len(contours), "kontur valid:",
len(kontur_valid))
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
for ax, judul, img in zip(axes, ["Mask awal", "Mask untuk kontur"],
[mask_awal, mask]):
    ax.imshow(img, cmap="gray", vmin=0, vmax=255);
ax.set_title(judul); ax.axis("off")
plt.tight_layout(); plt.show()
```

## Sel 5 — 5. Luas, bounding box, dan klasifikasi bentuk

Label adalah heuristik geometri, bukan pengenal kategori. Persegi diperiksa dengan sudut serta rasio sisi minAreaRect; lingkaran memerlukan circularity dan bentuk cukup bulat.

```
def dugaan_bentuk(cnt):
    area = float(cv2.contourArea(cnt))
    perimeter = float(cv2.arcLength(cnt, True))
    approx = cv2.approxPolyDP(cnt, 0.02 * perimeter, True)
```

```
    n = len(approx)
    circularity = 4 * np.pi * area / (perimeter ** 2) if perimeter >
0 else 0.0
    sisi = cv2.minAreaRect(cnt)[1]
    rasio = max(sisi) / min(sisi) if min(sisi) > 0 else float("inf")
    convex = bool(cv2.isContourConvex(approx))
    label = "objek tertutup lain"
    if n == 3:
        label = "segitiga"
    elif n == 4:
        pts = approx.reshape(-1, 2).astype(np.float32)
        cos_sudut = []
        for i in range(4):
            u = pts[(i - 1) % 4] - pts[i]
            v = pts[(i + 1) % 4] - pts[i]
            pembagi = float(np.linalg.norm(u) * np.linalg.norm(v))
            cos_sudut.append(abs(float(np.dot(u, v))) / pembagi if
pembagi else 1.0)
        siku = max(cos_sudut) < 0.25
        if convex and siku:
            label = "persegi" if rasio <= 1.10 else "persegi
panjang"
        else:
            label = "segiempat lain"
    elif n >= 5 and convex and circularity >= 0.80 and rasio <=
1.20:
        label = "kandidat lingkaran"
    return label, approx, area, perimeter, float(circularity)

anotasi = bgr.copy()
rekap = []
for nomor, cnt in enumerate(kontur_valid, start=1):
    label, approx, area, perimeter, circ = dugaan_bentuk(cnt)
    x, y, bw, bh = cv2.boundingRect(cnt)
    cv2.drawContours(anotasi, [cnt], -1, (0, 255, 0), 2)
    cv2.polylines(anotasi, [approx], True, (255, 0, 255), 2)
    cv2.rectangle(anotasi, (x, y), (x + bw - 1, y + bh - 1), (255,
0, 0), 2)
    cv2.putText(anotasi, f"{nomor}: {label}", (x, max(15, y - 8)),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1,
cv2.LINE_AA)
    item = {"nomor": nomor, "label": label, "luas_pixel2":
```

```
round(area, 2),
            "perimeter_pixel": round(perimeter, 2), "vertex":
len(approx),
            "circularity": round(circ, 3), "bbox_xywh": (x, y, bw,
bh)}
    rekap.append(item)
    print(item)
if not rekap:
    print("Tidak ada kontur valid. Periksa mask, polaritas,
MIN_AREA, dan posisi objek.")
plt.figure(figsize=(12, 8))
plt.imshow(cv2.cvtColor(anotasi, cv2.COLOR_BGR2RGB))
plt.title("Hijau: kontur; magenta: approx; biru: bounding box")
plt.axis("off"); plt.show()
```

## Jika terjadi error

**Banyak kontur kecil muncul.**

Periksa mask dan noise; atur prapemrosesan/MIN_AREA secara terdokumentasi, jangan hanya menaikkan batas tanpa melihat objek kecil.

**Tidak ada kontur valid.**

Periksa polaritas, mask kosong/penuh, objek menyentuh bingkai, dan apakah MIN_AREA terlalu besar.

**Persegi miring dilabeli bentuk lain.**

Periksa polygon, sudut, dan rasio minAreaRect; perspektif dapat mengubah persegi menjadi trapezoid, sehingga label harus dianggap heuristik.

**Luas dilaporkan sebagai cm².**

Gunakan pixel²; ukuran fisik memerlukan objek kalibrasi, skala, dan koreksi perspektif.

**Satu benda menghasilkan banyak garis Canny.**

Canny mendeteksi tekstur; gunakan mask foreground yang diperiksa untuk kontur siluet, bukan mengasumsikan semua tepi adalah batas 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.

---

Pengolahan Citra · Pendahuluan dan daftar seluruh materi