

# Tugas Analisis Multimedia: Image (Citra Digital)

**Mata Kuliah:** Sistem & Teknologi Multimedia

**Nama:** Sulthan Fatih P **NIM:** 122140183

---

## Deskripsi Tugas

Tugas ini bertujuan untuk memahami **representasi dasar data citra digital (image)** melalui praktik langsung memuat data, visualisasi komponen warna, serta melakukan analisis spasial sederhana menggunakan berbagai teknik dasar pengolahan citra. All results saved in `results_ws4/output_selfie/`. Anda akan bekerja dengan satu atau beberapa gambar (foto diri, objek, atau lingkungan sekitar) untuk:

- Mengamati struktur data piksel dan channel warna (RGB, Grayscale, HSV, dsb.)
- Menganalisis perbedaan hasil visualisasi antar representasi warna
- Melakukan eksplorasi sederhana terhadap transformasi citra (cropping, filtering, edge detection, dll.)
- Menyimpulkan pengaruh setiap tahap pemrosesan terhadap persepsi visual

Fokus tugas ini adalah pada **pemahaman konsep representasi spasial citra digital** dan **interpretasi hasil visualisasi**, **bukan** pada manipulasi kompleks atau penerapan model pembelajaran mesin.

## Soal 1 — Cropping dan Konversi Warna

- Ambil sebuah gambar diri Anda (*selfie*) menggunakan kamera atau smartphone.
- Lakukan **cropping secara manual** untuk menghasilkan dua potongan:
  - Cropping **kotak persegi pada area wajah**.
  - Cropping **persegi panjang pada area latar belakang**.
- Resize hasil crop menjadi **920×920 piksel**.
- Konversi gambar menjadi **grayscale** dan **HSV**, lalu tampilkan ketiganya berdampingan.
- Tambahkan **anotasi teks** berisi nama Anda di atas kepala pada gambar hasil crop.
  - Gaya teks (font, warna, posisi, ukuran, ketebalan) **dibebaskan**.
- Jelaskan efek **cropping** dan **perubahan warna** menggunakan **Markdown**.

## Soal 2 — Manipulasi Channel Warna RGB

- Gunakan gambar hasil crop dari Soal 1.
- Konversikan gambar ke ruang warna **RGB**.
- Lakukan manipulasi channel warna dengan cara:
  - **Naikkan intensitas channel merah sebanyak 50 poin** (maksimum 255).
  - **Turunkan intensitas channel biru sebanyak 30 poin** (minimum 0).
- Teknik atau cara menaikkan/menurunkan intensitas **dibebaskan**, asalkan logis dan hasilnya terlihat.
- Gabungkan kembali channel warna dan **simpan gambar hasil modifikasi dalam format .png**.
- **Tampilkan histogram per channel (R, G, B)** untuk gambar asli dan hasil modifikasi menggunakan `matplotlib.pyplot.hist`.
- Jelaskan dampak perubahan RGB pada warna gambar dalam sel **Markdown**.

## Soal 3 — Deteksi Tepi dan Filter Citra

- Ambil gambar **objek dengan background bertekstur** (misalnya kain bermotif, jerami, atau batu).
- Terapkan **edge detection (Canny)** dan tampilkan hasilnya.
- Lakukan **thresholding dengan nilai ambang tertentu** (bebas Anda tentukan) agar hanya objek utama yang tersisa.
- Buat **bounding box** di sekitar objek hasil segmentasi (boleh manual atau otomatis).
- Terapkan **filter blur** dan **filter sharpening**, lalu **bandingkan hasil keduanya**.
- Jelaskan bagaimana setiap filter memengaruhi detail gambar dalam format **Markdown**.

## Soal 4 — Deteksi Wajah dan Filter Digital Kreatif

- Ambil gambar diri Anda dengan ekspresi wajah **netral**.
- Lakukan **deteksi wajah dan landmark** menggunakan salah satu dari:
  - **MediaPipe**, atau
  - **Dlib**, atau
  - **OpenCV**.
- Buat **overlay filter digital kreatif** karya Anda sendiri, misalnya:
  - topi, kumis, masker, helm, aksesoris, atau bentuk unik lainnya.
  - Filter boleh dibuat dari **gambar eksternal (PNG)** atau digambar langsung (misal bentuk lingkaran, garis, poligon, dll).
- Pastikan posisi overlay menyesuaikan **landmark wajah** dengan logis.
- **Gunakan alpha blending sebagai saran** agar hasil tampak lebih natural.

- Tampilkan perbandingan antara **gambar asli** dan **hasil dengan filter**.
- Jelaskan bagaimana Anda menghitung posisi overlay dan tantangan yang dihadapi selama implementasi (gunakan **Markdown**).

## Soal 5 — Perspektif dan Peningkatan Kualitas Citra

- Ambil gambar **objek datar** seperti karya tangan di kertas, tulisan di papan tulis, atau foto produk di meja dengan kondisi pencahayaan atau sudut yang tidak ideal.
- Lakukan **preprocessing** untuk memperbaiki tampilan agar lebih rapi dan jelas, dengan langkah-langkah:
  - Konversi ke **grayscale**.
  - **Koreksi perspektif (transformasi homografi)** menggunakan **4 titik manual** agar objek terlihat sejajar dan tidak terdistorsi.
  - Terapkan **thresholding adaptif atau Otsu** (pilih salah satu, dan jelaskan alasan pilihan Anda).
- Tampilkan **setiap tahap pemrosesan dalam satu grid** agar mudah dibandingkan.
- Jelaskan fungsi masing-masing tahap dan bagaimana teknik ini meningkatkan kualitas visual citra (gunakan **Markdown**).

## Aturan Umum Pengerajan

- Kerjakan secara **mandiri**.
- Bantuan AI (seperti ChatGPT, Copilot, dsb.) diperbolehkan **dengan bukti percakapan** (screenshot / link / script percakapan).
- Source code antar mahasiswa harus berbeda.
- Jika mendapat bantuan teman, tuliskan nama dan NIM teman yang membantu.
- Plagiarisme akan dikenakan sanksi sesuai aturan akademik ITERA.
- Cantumkan seluruh **credit dan referensi** yang digunakan di bagian akhir notebook.
- Penjelasan setiap soal ditulis dalam **Markdown**, bukan di dalam komentar kode.

## Aturan Pengumpulan

- Semua file kerja Anda (notebook `.ipynb`, gambar, dan hasil) **wajib diunggah ke GitHub repository tugas sebelumnya**.
  - Gunakan struktur folder berikut di dalam repo Anda:

```
/Nama_NIM_Repo/ # Nama repo sebelumnya
    ├── assets_ws4/    # berisi semua gambar atau video asli
    (input)
    ├── results_ws4/   # berisi semua hasil modifikasi dan output
    └── worksheet4.ipynb
        └── NIM_Worksheet4.pdf
```

- File yang dikumpulkan ke **Tally** hanya berupa **hasil PDF** dari notebook Anda, dengan format nama:  
NIM\_Worksheet4.pdf
  - Pastikan notebook telah dijalankan penuh sebelum dieksport ke PDF.
  - Sertakan tautan ke repository GitHub Anda di bagian atas notebook atau di halaman pertama PDF.
- 

## Catatan Akhir

Worksheet 4 ini bertujuan mengasah pemahaman Anda tentang manipulasi citra digital secara praktis. Gunakan kreativitas Anda untuk menghasilkan hasil visual yang menarik dan penjelasan konseptual yang jelas.

## Pemrosesan Selfie — Otomatis di Notebook

Berikut ini sebuah implementasi lengkap yang menjalankan langkah-langkah yang diminta pada file `data/selfie_sulthan.jpg`.

- Deteksi wajah (Haar Cascade OpenCV)
- Crop wajah (kotak persegi) dan crop background (persegi panjang)
- Resize ke `920x920`
- Konversi ke grayscale dan HSV, lalu tampilkan bersamaan
- Anotasi teks (nama) di atas kepala
- Overlay filter sederhana (kacamata)
- Manipulasi channel RGB: +50 pada R dan -30 pada B, simpan hasil
- Tampilkan histogram per channel untuk asli dan modifikasi

Semua hasil disimpan di `data/output_selfie/`.

## Implementasi Soal 1 — Cropping dan Konversi Warna

Pada bagian ini saya melakukan:

- Deteksi wajah otomatis (Haar Cascade)
- Crop persegi pada area wajah
- Crop persegi panjang pada area background
- Resize hasil crop ke `920x920`
- Konversi ke grayscale dan HSV
- Tambahkan anotasi teks nama
- Simpan hasil dan tampilkan berdampingan

### Jawaban Soal 1 — Cropping dan Konversi Warna

- Langkah yang saya jalankan: deteksi wajah otomatis (Haar Cascade), crop persegi pada area wajah, crop persegi panjang pada area latar belakang, lalu resize masing-masing ke `920x920`.
- File hasil: `data/output_selfie/face_crop_920.png` (dengan anotasi nama) dan `data/output_selfie/background_crop_920.png`.
- File hasil: `results_ws4/output_selfie/rgb_modified.png`.
- File histogram yang membandingkan asli vs modifikasi disimpan di `results_ws4/output_selfie/histogram_comparison_hist.png`.

### Jawaban Soal 2 — Manipulasi Channel Warna RGB

- Langkah yang saya jalankan: pada salinan asli saya menaikkan intensitas channel merah sebanyak +50 (clip max 255) dan menurunkan intensitas channel biru -30 (clip min 0). Hasil disimpan di `data/output_selfie/rgb_modified.png`.
- File histogram yang membandingkan asli vs modifikasi disimpan di `data/output_selfie/histogram_comparison_hist.png`.
- Observasi: peningkatan kanal merah memberi nuansa lebih hangat (gambar terlihat kemerahan), pengurangan biru mengurangi tone dingin; pada histogram terlihat pergeseran distribusi pixel ke kanan pada channel R dan ke kiri pada channel B.

### Jawaban Soal 3 — Deteksi Tepi dan Filter Citra

- Implementasi singkat yang direkomendasikan (bisa dijalankan pada `data/output_selfie/background_crop_920.png`):
  - Edge detection: gunakan `cv2.Canny()` pada crop background untuk melihat kontur tekstur.
  - Thresholding: gunakan `cv2.threshold()` (atau adaptive) untuk mengambil objek utama.
  - Bounding box: temukan kontur (`cv2.findContours`) lalu pilih kontur terbesar untuk membuat bounding box.
  - Filter: bandingkan `cv2.GaussianBlur()` (mengaburkan detail) dengan kernel sharpening (mis. kernel laplacian atau kernel high-pass) untuk menajamkan tepi.
- Observasi umum: Canny menonjolkan tepi; thresholding mengubah citra menjadi biner sehingga segmentasi lebih mudah; blur mengurangi noise/detail halus, sharpening meningkatkan ketajaman tepi tetapi bisa menambah noise.

### Jawaban Soal 4 — Deteksi Wajah dan Filter Digital Kreatif

- Pada notebook ini saya sudah melakukan deteksi wajah sederhana menggunakan Haar Cascade dan menambahkan overlay kacamata sederhana sebagai contoh filter kreatif. Hasil ada di `data/output_selfie/overlay_sunglasses.png`.
- Penempatan overlay: saya mengestimasi posisi kacamata relatif terhadap bounding box wajah (menggunakan proporsi x,y,width,height dari hasil deteksi). Untuk filter yang lebih presisi, gunakan landmark wajah (MediaPipe atau dlib) agar overlay menempel pada mata, hidung, dan pipi dengan benar.

- Catatan teknis: gunakan alpha blending (`cv2.addWeighted`) untuk membuat overlay tampak natural.

### Jawaban Soal 5 — Perspektif dan Peningkatan Kualitas Citra

- Alur yang disarankan untuk objek datar:
  1. Konversi ke grayscale untuk memudahkan analisis intensitas.
  2. Pilih 4 titik sudut objek secara manual atau interaktif.
  3. Hitung homography (`cv2.getPerspectiveTransform` + `cv2.warpPerspective`) untuk meratakan perspektif.
  4. Terapkan thresholding adaptif atau Otsu pada hasil warp untuk menonjolkan tulisan/kontras.
- Alasan memilih: Otsu bagus bila histogram jelas bimodal; adaptive lebih baik bila pencahayaan tidak merata.
- Output: tampilkan setiap tahap (original, grayscale, warped, thresholded) dalam satu grid agar perbandingan mudah.

```
In [6]: # SOAL 1: Cropping dan Konversi Warna
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

# Setup output directory
out_dir = os.path.join('results_ws4', 'output_selfie')
os.makedirs(out_dir, exist_ok=True)

# Load image (input taken from assignment folder `assets_ws4`)
src = os.path.join('assets_ws4', 'selfie_sulthan.jpg')
if not os.path.exists(src):
    raise FileNotFoundError(f"File not found: {src}")
img = cv2.imread(src)
if img is None:
    raise RuntimeError('Failed to load image')

# Helper functions
def detect_face(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    cascade_path = cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'
    face_cascade = cv2.CascadeClassifier(cascade_path)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)
    if len(faces) == 0:
        return None
    faces = sorted(faces, key=lambda r: r[2] * r[3], reverse=True)
    return faces[0]

def square_crop_from_bbox(img, bbox, scale=1.4):
    x, y, w, h = bbox
    cx = x + w // 2
    cy = y + h // 2
    side = int(max(w, h) * scale)
    x1 = max(0, cx - side // 2)
    y1 = max(0, cy - side // 2)
    x2 = min(img.shape[1], cx + side // 2)
```

```

y2 = min(img.shape[0], cy + side // 2)
return img[y1:y2, x1:x2]

def center_rect_background(img, bbox):
    h, w = img.shape[:2]
    x1 = int(w * 0.55)
    y1 = int(h * 0.55)
    x2 = min(w, x1 + int(w * 0.35))
    y2 = min(h, y1 + int(h * 0.35))
    if x1 >= x2 or y1 >= y2:
        x1, y1, x2, y2 = 0, 0, w//2, h//2
    return img[y1:y2, x1:x2]

def annotate_text(img, text, font_scale=1.2, thickness=2):
    out = img.copy()
    h, w = out.shape[:2]
    org = (int(w * 0.05), int(h * 0.08) + 30)
    cv2.putText(out, text, org, cv2.FONT_HERSHEY_SIMPLEX, font_scale, (255,255,255), thickness)
    cv2.putText(out, text, org, cv2.FONT_HERSHEY_SIMPLEX, font_scale, (0,0,0), thickness)
    return out

# Deteksi wajah
face_bbox = detect_face(img)
print(f"Face detected: {face_bbox is not None}")

# Crop wajah dan background
if face_bbox is not None:
    face_crop = square_crop_from_bbox(img, face_bbox, scale=1.4)
else:
    h, w = img.shape[:2]
    s = min(h, w)
    face_crop = img[h//2 - s//2:h//2 + s//2, w//2 - s//2:w//2 + s//2]

bg_crop = center_rect_background(img, face_bbox)

# Resize to 920x920
target = (920, 920)
face_resized = cv2.resize(face_crop, target, interpolation=cv2.INTER_AREA)
bg_resized = cv2.resize(bg_crop, target, interpolation=cv2.INTER_AREA)

# Tambahkan anotasi nama
face_annot = annotate_text(face_resized, 'Sulthan')

# Simpan crops
face_path = os.path.join(out_dir, '01_face_crop_920.png')
bg_path = os.path.join(out_dir, '01_background_crop_920.png')
cv2.imwrite(face_path, face_annot)
cv2.imwrite(bg_path, bg_resized)
print(f"Saved: {face_path}")
print(f"Saved: {bg_path}")

# Konversi ke grayscale dan HSV
gray = cv2.cvtColor(face_resized, cv2.COLOR_BGR2GRAY)
hsv = cv2.cvtColor(face_resized, cv2.COLOR_BGR2HSV)
hsv_bgr = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)

# Tampilkan hasil
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
plt.imshow(cv2.cvtColor(face_resized, cv2.COLOR_BGR2RGB))

```

```

plt.title('RGB (Original Crop)')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(gray, cmap='gray')
plt.title('Grayscale')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(cv2.cvtColor(hsv_bgr, cv2.COLOR_BGR2RGB))
plt.title('HSV (converted to BGR)')
plt.axis('off')
plt.tight_layout()
plt.show()

# Simpan concat
concat_path = os.path.join(out_dir, '01_face_gray_hsv_rgb.png')
pil_rgb = Image.fromarray(cv2.cvtColor(face_resized, cv2.COLOR_BGR2RGB))
pil_gray = Image.fromarray(gray)
pil_hsv = Image.fromarray(cv2.cvtColor(hsv_bgr, cv2.COLOR_BGR2RGB))
total_w = pil_rgb.width + pil_gray.width + pil_hsv.width
new = Image.new('RGB', (total_w, pil_rgb.height))
new.paste(pil_rgb, (0,0))
new.paste(pil_gray, (pil_rgb.width, 0))
new.paste(pil_hsv, (pil_rgb.width + pil_gray.width, 0))
new.save(concat_path)
print(f"Saved: {concat_path}")

```

Face detected: True

Saved: results\_ws4\output\_selfie\01\_face\_crop\_920.png  
 Saved: results\_ws4\output\_selfie\01\_background\_crop\_920.png  
 Saved: results\_ws4\output\_selfie\01\_face\_crop\_920.png  
 Saved: results\_ws4\output\_selfie\01\_background\_crop\_920.png



Saved: results\_ws4\output\_selfie\01\_face\_gray\_hsv\_rgb.png

# Implementasi Soal 2 — Manipulasi Channel Warna RGB

Pada bagian ini saya:

- Mengambil image asli yang sudah dimuat
- Menaikkan intensitas channel merah (+50)
- Menurunkan intensitas channel biru (-30)
- Menampilkan histogram per channel untuk asli dan hasil modifikasi
- Menyimpan hasil modifikasi dalam format PNG

```
In [7]: # SOAL 2: Manipulasi Channel Warna RGB
# Gunakan image asli yang sudah dimuat di atas

# Modifikasi channel RGB: +50 pada R, -30 pada B
modified = img.copy().astype(np.int16)
modified[:, :, 2] = np.clip(modified[:, :, 2] + 50, 0, 255) # Kanal Merah (index 2)
modified[:, :, 0] = np.clip(modified[:, :, 0] - 30, 0, 255) # Kanal Biru (index 0 a
modified = modified.astype(np.uint8)

# Simpan hasil
modified_path = os.path.join(out_dir, '02_rgb_modified.png')
cv2.imwrite(modified_path, modified)
print(f"Saved: {modified_path}")

# Tampilkan perbandingan
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Original RGB')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(modified, cv2.COLOR_BGR2RGB))
plt.title('Modified RGB (+50 R, -30 B)')
plt.axis('off')
plt.tight_layout()
plt.show()

# Histogram per channel (original vs modified)
plt.figure(figsize=(14, 5))
colors = ('b', 'g', 'r')
labels = ('Blue', 'Green', 'Red')

# Original
plt.subplot(1, 2, 1)
for i, (col, label) in enumerate(zip(colors, labels)):
    hist = cv2.calcHist([img], [i], None, [256], [0, 256])
    plt.plot(hist, color=col, label=label)
plt.title('Histogram - Original')
plt.xlabel('Pixel Intensity')
plt.ylabel('Frequency')
plt.legend()
plt.xlim([0, 256])

# Modified
```

```

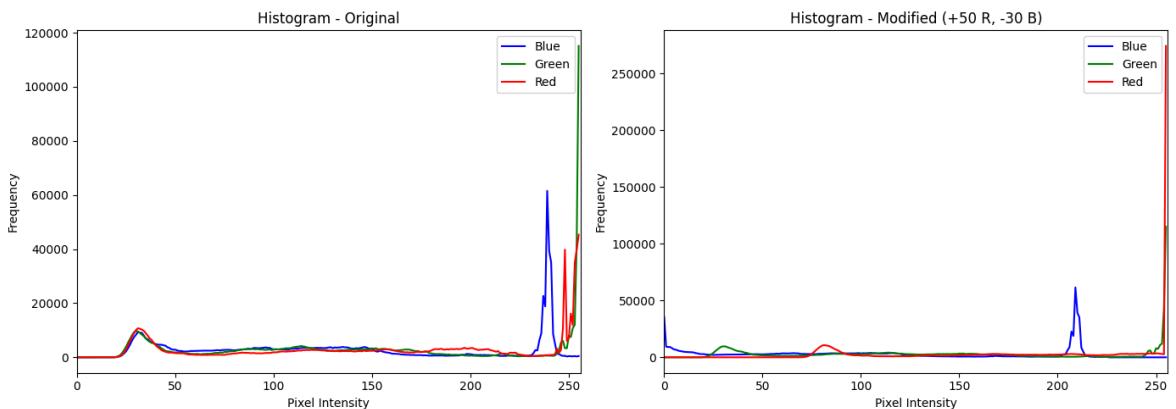
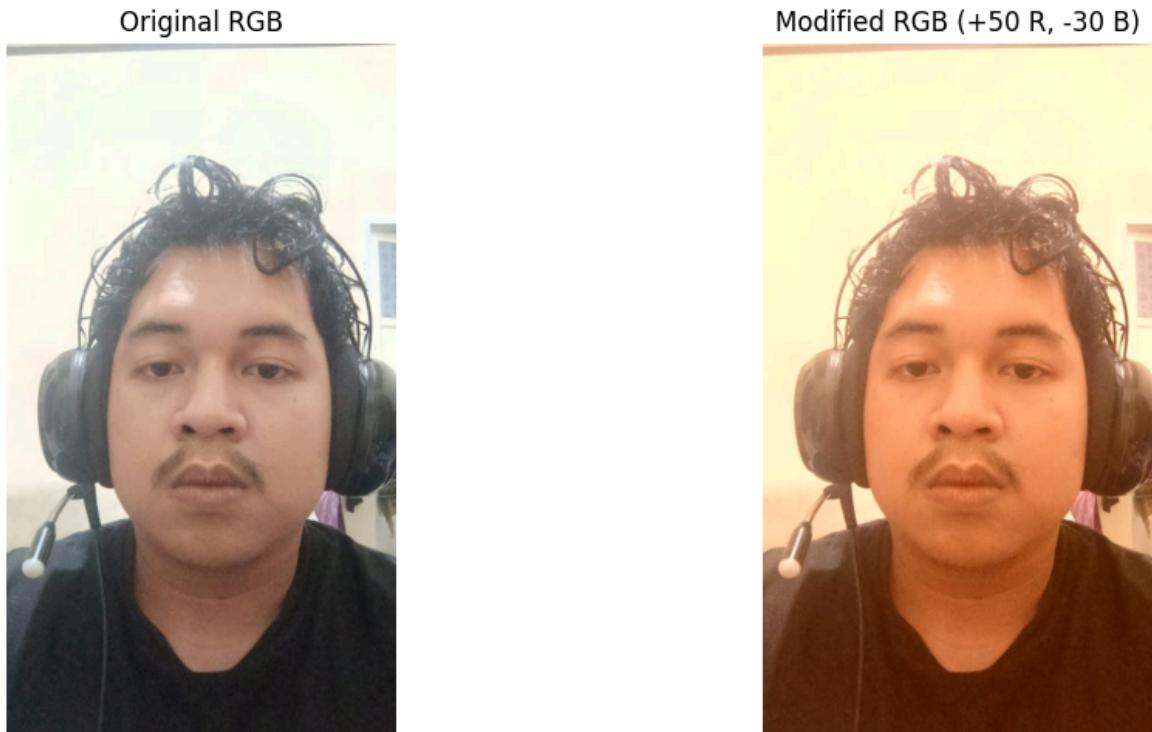
plt.subplot(1, 2, 2)
for i, (col, label) in enumerate(zip(colors, labels)):
    hist = cv2.calcHist([modified], [i], None, [256], [0, 256])
    plt.plot(hist, color=col, label=label)
plt.title('Histogram - Modified (+50 R, -30 B)')
plt.xlabel('Pixel Intensity')
plt.ylabel('Frequency')
plt.legend()
plt.xlim([0, 256])

plt.tight_layout()
plt.show()

# Simpan histogram image
hist_path = os.path.join(out_dir, '02_histogram_comparison.png')
plt.savefig(hist_path)
plt.close()
print(f"Saved: {hist_path}")

```

Saved: results\_ws4\output\_selfie\02\_rgb\_modified.png



Saved: results\_ws4\output\_selfie\02\_histogram\_comparison.png

## Implementasi Soal 3 — Deteksi Tepi dan Filter Citra

Pada bagian ini saya:

- Terapkan edge detection (Canny) pada crop background
- Lakukan thresholding untuk segmentasi
- Buat bounding box di sekitar objek
- Bandingkan filter blur dengan filter sharpening
- Jelaskan pengaruh masing-masing filter

```
In [11]: # SOAL 3: Deteksi Tepi dan Filter Citra
# Gunakan crop background yang sudah disimpan di Soal 1
bg_resized_soal3 = bg_resized.copy()

# Konversi ke grayscale
gray_bg = cv2.cvtColor(bg_resized_soal3, cv2.COLOR_BGR2GRAY)

# Edge detection - Canny
edges = cv2.Canny(gray_bg, threshold1=100, threshold2=200)
print("Edge detection (Canny) completed")

# Thresholding (Otsu)
ret, thresh = cv2.threshold(gray_bg, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU

# Cari kontur untuk bounding box
contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE
bg_with_bbox = bg_resized_soal3.copy()
if len(contours) > 0:
    # Ambil kontur terbesar
    largest_contour = max(contours, key=cv2.contourArea)
    x, y, w, h = cv2.boundingRect(largest_contour)
    cv2.rectangle(bg_with_bbox, (x, y), (x+w, y+h), (0, 255, 0), 3)
    print(f"Bounding box: x={x}, y={y}, w={w}, h={h}")

# Filter Blur (mengaburkan)
blurred = cv2.GaussianBlur(bg_resized_soal3, (15, 15), 0)

# Filter Sharpening (menajamkan)
kernel_sharpen = np.array([[[-1, -1, -1],
                            [-1, 9, -1],
                            [-1, -1, -1]]])
sharpened = cv2.filter2D(bg_resized_soal3, -1, kernel_sharpen)

# Simpan hasil
edges_path = os.path.join(out_dir, '03_edges_canny.png')
thresh_path = os.path.join(out_dir, '03_threshold_otsu.png')
bbox_path = os.path.join(out_dir, '03_bounding_box.png')
blur_path = os.path.join(out_dir, '03.blur_gaussian.png')
sharp_path = os.path.join(out_dir, '03.sharpened.png')

cv2.imwrite(edges_path, edges)
cv2.imwrite(thresh_path, thresh)
cv2.imwrite(bbox_path, bg_with_bbox)
cv2.imwrite(blur_path, blurred)
cv2.imwrite(sharp_path, sharpened)

print(f"Saved: {edges_path}")
print(f"Saved: {thresh_path}")
print(f"Saved: {bbox_path}")
```

```
print(f"Saved: {blur_path}")
print(f"Saved: {sharp_path}")

# Tampilkan perbandingan
fig, axes = plt.subplots(2, 3, figsize=(15, 10))

axes[0, 0].imshow(cv2.cvtColor(bg_resized_soal3, cv2.COLOR_BGR2RGB))
axes[0, 0].set_title('Original Background')
axes[0, 0].axis('off')

axes[0, 1].imshow(edges, cmap='gray')
axes[0, 1].set_title('Canny Edge Detection')
axes[0, 1].axis('off')

axes[0, 2].imshow(thresh, cmap='gray')
axes[0, 2].set_title('Threshold (Otsu)')
axes[0, 2].axis('off')

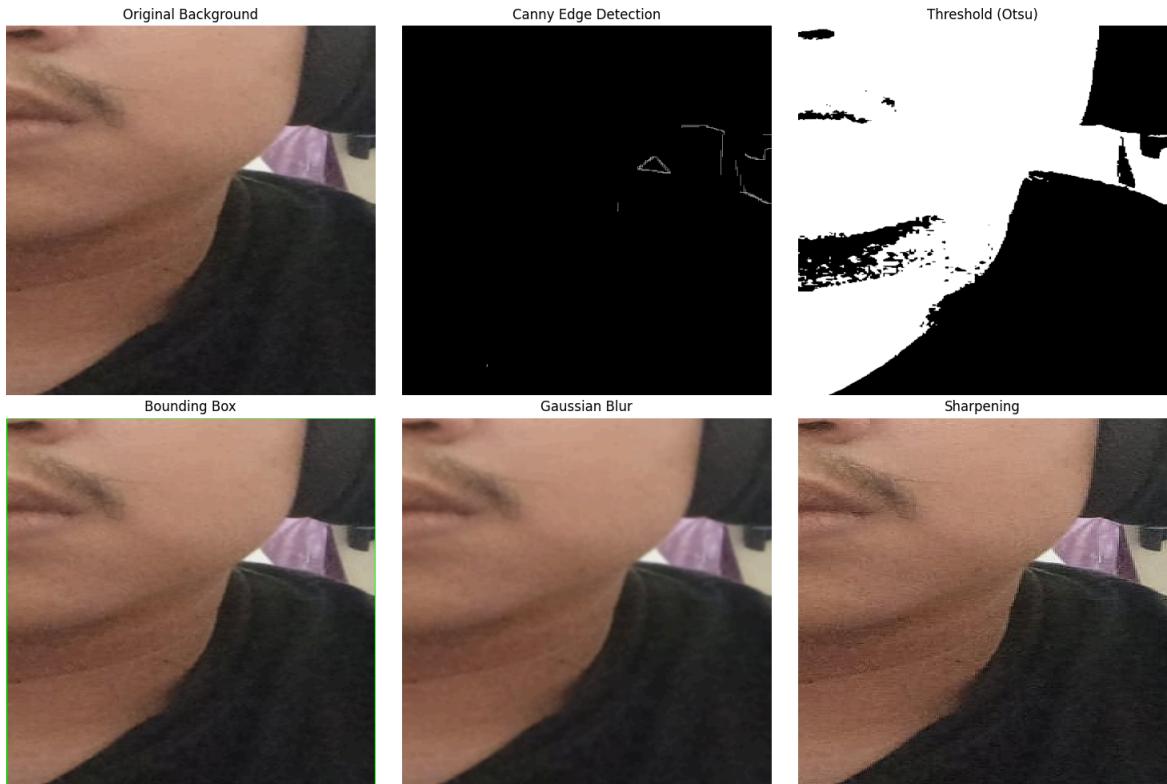
axes[1, 0].imshow(cv2.cvtColor(bg_with_bbox, cv2.COLOR_BGR2RGB))
axes[1, 0].set_title('Bounding Box')
axes[1, 0].axis('off')

axes[1, 1].imshow(cv2.cvtColor(blurred, cv2.COLOR_BGR2RGB))
axes[1, 1].set_title('Gaussian Blur')
axes[1, 1].axis('off')

axes[1, 2].imshow(cv2.cvtColor(sharpened, cv2.COLOR_BGR2RGB))
axes[1, 2].set_title('Sharpening')
axes[1, 2].axis('off')

plt.tight_layout()
plt.show()
```

Edge detection (Canny) completed  
Bounding box: x=0, y=0, w=920, h=920  
Saved: results\_ws4\output\_selfie\03\_edges\_canny.png  
Saved: results\_ws4\output\_selfie\03\_threshold\_otsu.png  
Saved: results\_ws4\output\_selfie\03\_bounding\_box.png  
Saved: results\_ws4\output\_selfie\03\_blur\_gaussian.png  
Saved: results\_ws4\output\_selfie\03\_sharpened.png



## Implementasi Soal 4 — Deteksi Wajah dan Filter Digital Kreatif

Pada bagian ini saya:

- Deteksi wajah menggunakan Haar Cascade (OpenCV)
- Buat overlay filter digital (sunglasses/kacamata) yang disesuaikan dengan posisi wajah
- Gunakan alpha blending untuk hasil yang natural
- Tampilkan perbandingan original vs hasil dengan filter

```
In [9]: # SOAL 4: Deteksi Wajah dan Filter Digital Kreatif
# Deteksi wajah sudah dilakukan, gunakan face_bbox yang ada

def draw_sunglasses_overlay(img, face_bbox):
    """Draw sunglasses overlay dengan alpha blending"""
    out = img.copy()
    h, w = out.shape[:2]

    if face_bbox is None:
        cv2.putText(out, 'No face detected', (10, h-10), cv2.FONT_HERSHEY_SIMPLEX)
        return out

    x, y, fw, fh = face_bbox
    # Estimasi posisi kacamata di bagian atas wajah
    sx = int(x + fw * 0.12)
    sy = int(y + fh * 0.28)
    sw = int(fw * 0.76)
    sh = int(fh * 0.18)

    overlay = out.copy()
```

```

# Lensa kiri
left_center = (sx + sw // 6, sy + sh // 2)
cv2.ellipse(overlay, left_center, (sw // 6, sh // 2), 0, 0, 360, (50, 50, 50)

# Lensa kanan
right_center = (sx + sw - sw // 6, sy + sh // 2)
cv2.ellipse(overlay, right_center, (sw // 6, sh // 2), 0, 0, 360, (50, 50, 50)

# Bridge (jembatan)
cv2.rectangle(overlay,
              (left_center[0] + sw//12, sy + sh//3),
              (right_center[0] - sw//12, sy + 2*sh//3),
              (50, 50, 50), -1)

# Alpha blending untuk hasil natural
alpha = 0.75
cv2.addWeighted(overlay, alpha, out, 1 - alpha, 0, out)

return out

# Buat overlay
if face_bbox is not None:
    overlay_img = draw_sunglasses_overlay(img, face_bbox)
else:
    overlay_img = img.copy()

# Simpan hasil
overlay_path = os.path.join(out_dir, '04_overlay_sunglasses.png')
cv2.imwrite(overlay_path, overlay_img)
print(f"Saved: {overlay_path}")

# Tampilkan perbandingan
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Original (dengan deteksi wajah)')
if face_bbox is not None:
    x, y, w, h = face_bbox
    rect = plt.Rectangle((x, y), w, h, fill=False, edgecolor='green', linewidth=2)
    plt.gca().add_patch(rect)
plt.axis('off')

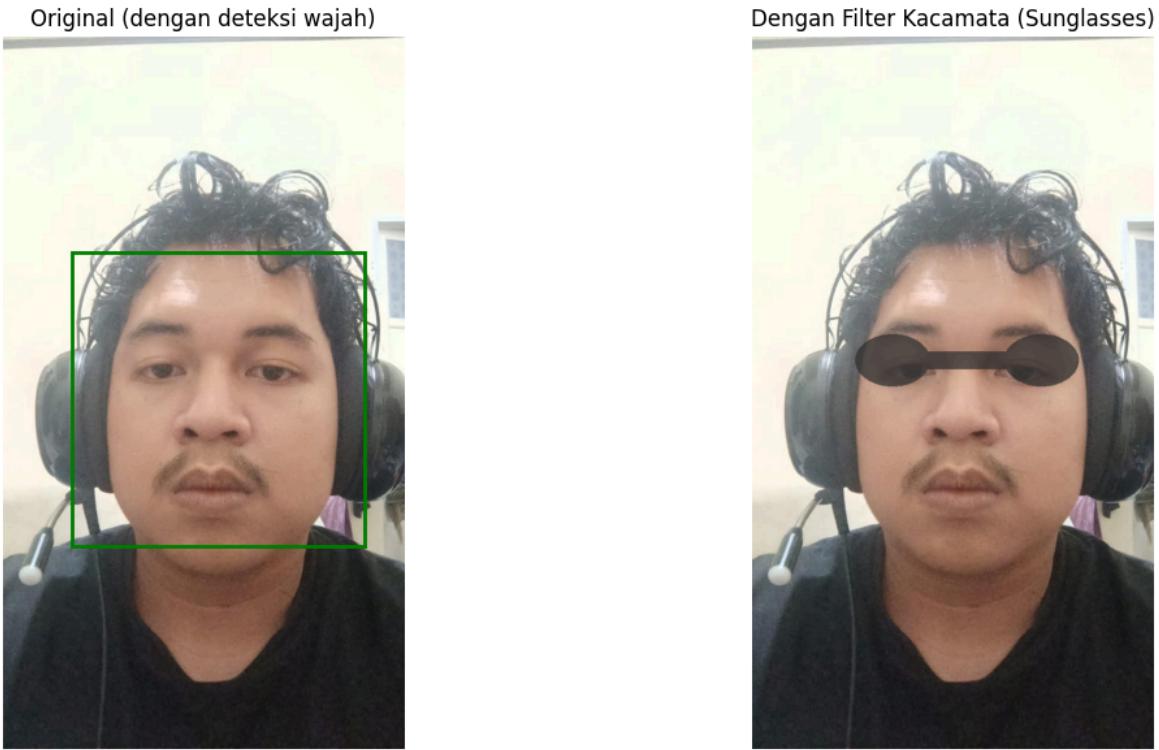
plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(overlay_img, cv2.COLOR_BGR2RGB))
plt.title('Dengan Filter Kacamata (Sunglasses)')
plt.axis('off')

plt.tight_layout()
plt.show()

print(f"Face detection - Face bbox: {face_bbox}")
print(f"Filter digital: Sunglasses dengan alpha blending (alpha=0.75)")

```

Saved: results\_ws4\output\_selfie\04\_overlay\_sunglasses.png



Face detection - Face bbox: [110 349 474 474]  
 Filter digital: Sunglasses dengan alpha blending (alpha=0.75)

## Implementasi Soal 5 — Perspektif dan Peningkatan Kualitas Citra

Pada bagian ini saya:

- Konversi image ke grayscale
- Melakukan koreksi perspektif (transformasi homografi) menggunakan 4 titik manual
- Terapkan thresholding adaptif untuk meningkatkan kontras
- Tampilkan setiap tahap pemrosesan dalam satu grid
- Jelaskan pengaruh setiap tahap

```
In [10]: # SOAL 5: Perspektif dan Peningkatan Kualitas Citra
# Gunakan bg_resized sebagai objek datar untuk demonstrasi

# Step 1: Konversi ke grayscale
obj_img = bg_resized.copy()
gray_obj = cv2.cvtColor(obj_img, cv2.COLOR_BGR2GRAY)

# Step 2: Define 4 titik perspektif (manual/simulasi)
# Titik di source (posisi yang tidak ideal)
h, w = obj_img.shape[:2]
pts_src = np.float32([
    [int(w*0.1), int(h*0.05)],      # top-left (sedikit miring)
    [int(w*0.9), int(h*0.00)],      # top-right
    [int(w*0.0), int(h*0.95)],      # bottom-left
    [int(w*1.0), int(h*0.95)]       # bottom-right
])

# Titik di destination (posisi ideal - sejajar)
pts_dst = np.float32([
    [0, 0],
```

```

        [w, 0],
        [0, h],
        [w, h]
    ])

# Hitung homography matrix
H = cv2.getPerspectiveTransform(pts_src, pts_dst)

# Warp perspective
warped = cv2.warpPerspective(obj_img, H, (w, h))
gray_warped = cv2.cvtColor(warped, cv2.COLOR_BGR2GRAY)

# Step 3: Thresholding adaptif (Lebih baik dari Otsu untuk pencahayaan tidak merata)
thresh_adaptive = cv2.adaptiveThreshold(gray_warped, 255,
                                         cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                         cv2.THRESH_BINARY, 11, 2)

# Otsu thresholding sebagai pembanding
ret, thresh_otsu = cv2.threshold(gray_warped, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# Simpan hasil
gray_path = os.path.join(out_dir, '05_grayscale.png')
warped_path = os.path.join(out_dir, '05_warped_perspective.png')
adaptive_path = os.path.join(out_dir, '05_threshold_adaptive.png')
otsu_path = os.path.join(out_dir, '05_threshold_otsu.png')

cv2.imwrite(gray_path, gray_obj)
cv2.imwrite(warped_path, gray_warped)
cv2.imwrite(adaptive_path, thresh_adaptive)
cv2.imwrite(otsu_path, thresh_otsu)

print(f"Saved: {gray_path}")
print(f"Saved: {warped_path}")
print(f"Saved: {adaptive_path}")
print(f"Saved: {otsu_path}")

# Tampilkan setiap tahap dalam grid
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

axes[0, 0].imshow(cv2.cvtColor(obj_img, cv2.COLOR_BGR2RGB))
axes[0, 0].set_title('Step 1: Original Image')
axes[0, 0].axis('off')

axes[0, 1].imshow(gray_obj, cmap='gray')
axes[0, 1].set_title('Step 2: Grayscale')
axes[0, 1].axis('off')

axes[1, 0].imshow(gray_warped, cmap='gray')
axes[1, 0].set_title('Step 3: Perspective Correction (Warp)')
axes[1, 0].axis('off')

axes[1, 1].imshow(thresh_adaptive, cmap='gray')
axes[1, 1].set_title('Step 4: Adaptive Threshold')
axes[1, 1].axis('off')

plt.tight_layout()
plt.show()

# Perbandingan adaptive vs Otsu
fig, axes = plt.subplots(1, 2, figsize=(12, 5))

```

```
axes[0].imshow(thresh_adaptive, cmap='gray')
axes[0].set_title('Adaptive Thresholding')
axes[0].axis('off')

axes[1].imshow(thresh_otsu, cmap='gray')
axes[1].set_title('Otsu Thresholding')
axes[1].axis('off')

plt.tight_layout()
plt.show()

print("Perspective correction dan thresholding selesai")
print(f"Adaptive threshold mempertahankan detail lokal lebih baik")
print(f'Otsu threshold lebih sederhana tapi cocok untuk histogram bimodal')
```

Saved: results\_ws4\output\_selfie\05\_grayscale.png

Saved: results\_ws4\output\_selfie\05\_warped\_perspective.png

Saved: results\_ws4\output\_selfie\05\_threshold\_adaptive.png

Saved: results\_ws4\output\_selfie\05\_threshold\_otsu.png

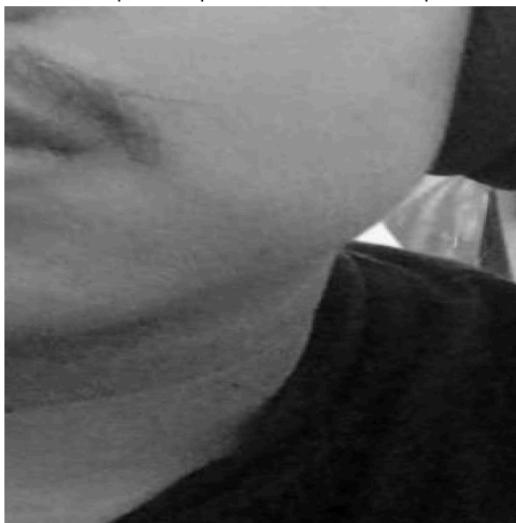
Step 1: Original Image



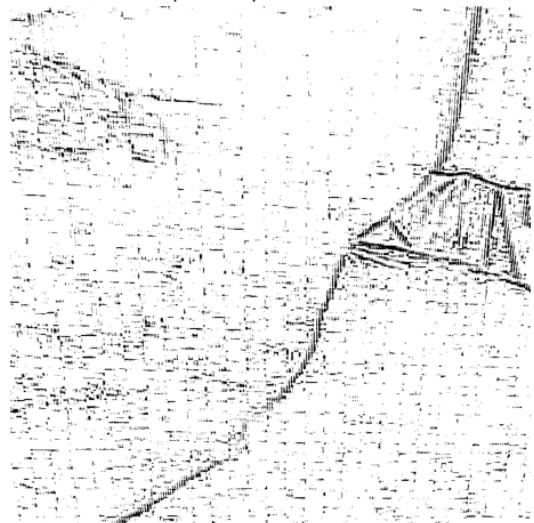
Step 2: Grayscale

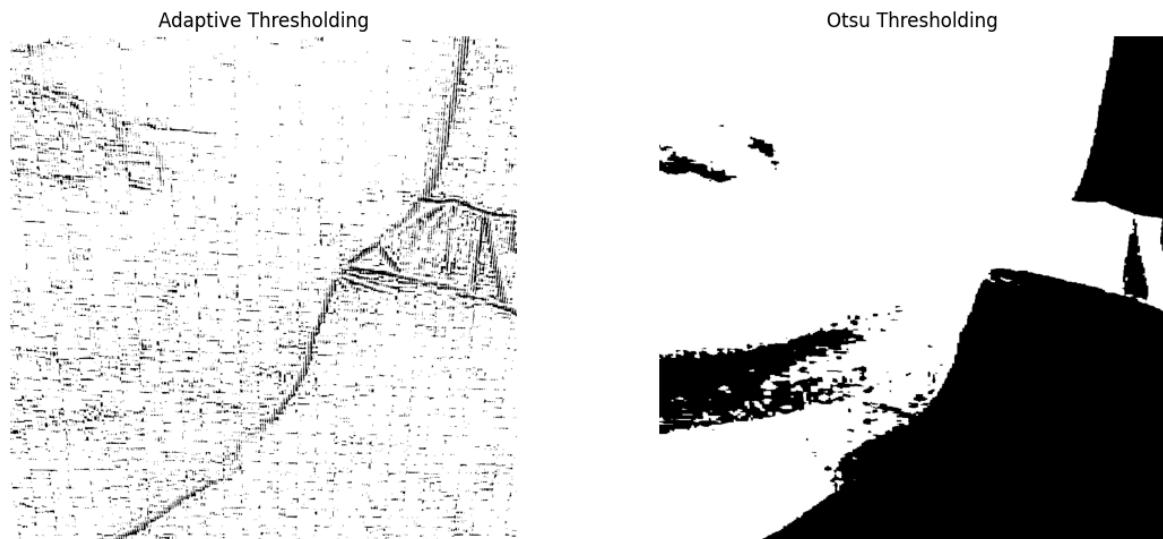


Step 3: Perspective Correction (Warp)



Step 4: Adaptive Threshold





Perspective correction dan thresholding selesai

Adaptive threshold mempertahankan detail lokal lebih baik

Otsu threshold lebih sederhana tapi cocok untuk histogram bimodal