

# Tugas Analisis Multimedia: Image (Citra Digital)

**Mata Kuliah:** Sistem & Teknologi Multimedia

**Nama:** Rachel Olivia Manullang

**NIM:** 122140181

---

## 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.

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**.

# Import library dan baca gambar

```
In [1]: import cv2
import numpy as np
import matplotlib.pyplot as plt

img_bgr = cv2.imread("assets_ws4/selfie_soal1.jpg")
img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)

print("Ukuran gambar:", img_rgb.shape)

plt.imshow(img_rgb)
plt.axis("off")
```

Ukuran gambar: (2048, 1152, 3)

Out[1]: (-0.5, 1151.5, 2047.5, -0.5)



- Mengimpor OpenCV, NumPy, dan Matplotlib
- Membaca gambar selfie dari file
- Mengubah BGR ke RGB supaya warna benar saat ditampilkan
- Menampilkan gambar asli dan mencetak ukurannya (tinggi, lebar, channel)

## Tentukan area cropping wajah dan latar belakang

```
In [9]: # koordinat wajah persegi
x1 = 200
y1 = 300
s = 700

face_crop = img_rgb[y1:y1+s, x1:x1+s]

# koordinat latar persegi panjang
x_bg1 = 1000
y_bg1 = 200
w_bg = 700
h_bg = 400

bg_crop = img_rgb[y_bg1:y_bg1+h_bg, x_bg1:x_bg1+w_bg]

plt.figure(figsize=(8,4))
plt.subplot(1,2,1)
plt.imshow(face_crop)
plt.title("Crop Wajah Persegi")
plt.axis("off")

plt.subplot(1,2,2)
plt.imshow(bg_crop)
plt.title("Crop Latar Persegi Panjang")
plt.axis("off")
```

Out[9]: (-0.5, 151.5, 399.5, -0.5)

Crop Wajah Persegi



Crop Latar Persegi Panjang



- Kode ini memotong bagian wajah dengan bentuk persegi menggunakan panjang sisi s.
- Kode juga memotong area latar belakang dengan lebar dan tinggi berbeda sehingga berbentuk persegi panjang.
- Kedua hasil crop ditampilkan berdampingan.

## Resize menjadi 920 x 920 untuk wajah dan 920 x 920 untuk latar

In [10]:

```
size = (920, 920)

face_resized = cv2.resize(face_crop, size)
bg_resized = cv2.resize(bg_crop, size)

plt.figure(figsize=(8,4))
plt.subplot(1,2,1)
plt.imshow(face_resized)
plt.title("Wajah 920x920")
plt.axis("off")

plt.subplot(1,2,2)
plt.imshow(bg_resized)
plt.title("Latar 920x920")
plt.axis("off")
```

Out[10]: (-0.5, 919.5, 919.5, -0.5)

Wajah 920x920



Latar 920x920



- Kode ini mengubah ukuran crop wajah dan latar ke 920x920.
- Hasil ditampilkan agar bisa memastikan ukurannya sudah benar.

# Konversi wajah ke grayscale dan HSV.

```
In [11]: face_gray = cv2.cvtColor(face_resized, cv2.COLOR_RGB2GRAY)
face_hsv = cv2.cvtColor(face_resized, cv2.COLOR_RGB2HSV)

face_hsv_to_rgb = cv2.cvtColor(face_hsv, cv2.COLOR_HSV2RGB)

plt.figure(figsize=(12,4))
plt.subplot(1,3,1)
plt.imshow(face_resized)
plt.title("RGB")
plt.axis("off")

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

plt.subplot(1,3,3)
plt.imshow(face_hsv_to_rgb)
plt.title("HSV")
plt.axis("off")
```

Out[11]: (-0.5, 919.5, 919.5, -0.5)



- Kode ini mengubah gambar wajah ke grayscale dan HSV lalu menampilkan kedua hasilnya bersama gambar asli.
- Ini membantu melihat perbedaan informasi warna pada tiap ruang warna.

## Tambahkan teks nama di atas kepala.

```
In [13]: face_annot_bgr = cv2.cvtColor(face_resized, cv2.COLOR_RGB2BGR)

x_text = 200
y_text = 100

cv2.putText(
face_annot_bgr,
"RACHELLL",
(x_text, y_text),
cv2.FONT_HERSHEY_SIMPLEX,
2,
(0, 255, 0),
4,
cv2.LINE_AA
)

face_annot_rgb = cv2.cvtColor(face_annot_bgr, cv2.COLOR_BGR2RGB)

plt.imshow(face_annot_rgb)
plt.axis("off")
```

```
Out[13]: (-0.5, 919.5, 919.5, -0.5)
```



- Kode ini menulis teks nama pada area atas kepala.
- Kamu dapat mengubah posisi teks dengan mengubah x\_text dan y\_text.

### Efek Cropping

- Cropping membatasi area gambar hanya pada bagian yang dianggap penting.
- Teknik ini menghilangkan latar yang tidak perlu sehingga fokus tertuju pada objek utama.
- Cropping persegi pada wajah membuat fitur wajah lebih jelas.
- Cropping persegi panjang pada latar belakang membantu membandingkan tekstur atau warna tanpa gangguan elemen lain.

### Efek Perubahan Warna

- Perubahan nilai RGB menggeser keseimbangan warna pada gambar.
- Penambahan nilai pada kanal tertentu membuat warna lebih kuat pada kanal tersebut.
- Pengurangan nilai menurunkan dominasi kanal tertentu sehingga gambar bergeser ke warna lain.
- Efek ini mengubah nuansa visual sehingga lebih hangat atau lebih dingin tergantung kanal yang dimodifikasi.

## Soal 2 — Manipulasi Channel Warna RGB

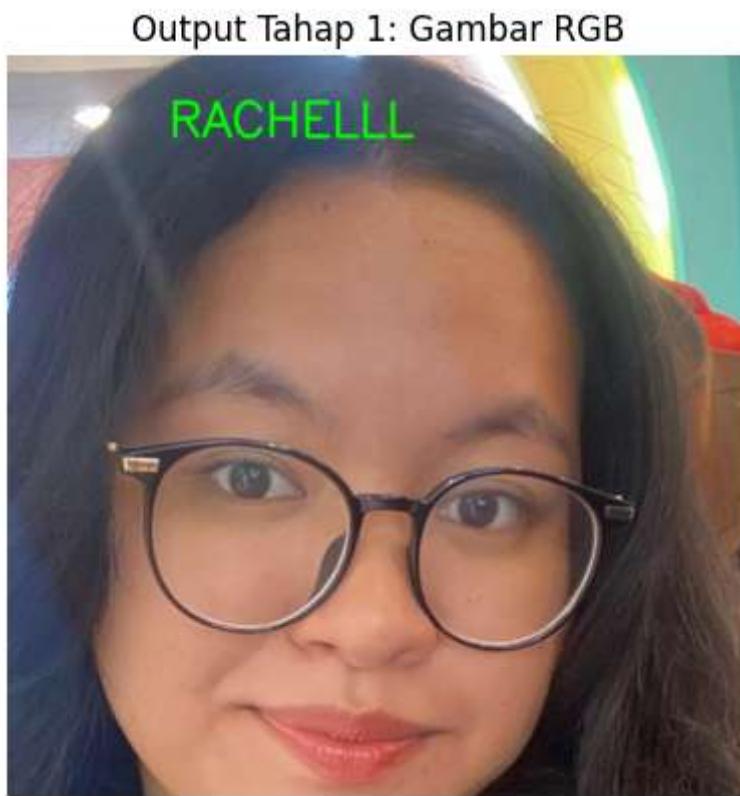
- Gunakan gambar hasil crop dari Soal 1.
- Konversikan gambar ke ruang warna **RGB**.
- Lakukan manipulasi channel warna dengan cara:
  - **Naikkan intensitas channel merah sebanyak 50 poin** (maksimum 255).
  - **Turunkan intensitas channel biru sebanyak 30 poin** (minimum 0).
- Teknik atau cara menaikkan/menurunkan 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**.

## Konversi gambar hasil nomor 1 ke RGB

```
In [17]: img = face_annot_rgb.copy()  
plt.imshow(img)
```

```
plt.title("Output Tahap 1: Gambar RGB")
plt.axis("off")
```

Out[17]: (-0.5, 919.5, 919.5, -0.5)



Tahap ini menyalin gambar hasil nomor 1 agar siap diproses tanpa mengubah aslinya.

## Pisahkan channel R, G, B

```
In [20]: R = img[:, :, 0]
          G = img[:, :, 1]
          B = img[:, :, 2]
```

Tahap ini memisahkan gambar ke tiga channel warna agar bisa dimanipulasi secara terpisah.

## Manipulasi channel warna

```
In [22]: R_mod = np.clip(R + 50, 0, 255)
          B_mod = np.clip(B - 30, 0, 255)

          R_mod = np.clip(R + 50, 0, 255)
          B_mod = np.clip(B - 30, 0, 255)

          plt.figure(figsize=(10, 4))
          plt.subplot(1, 2, 1)
```

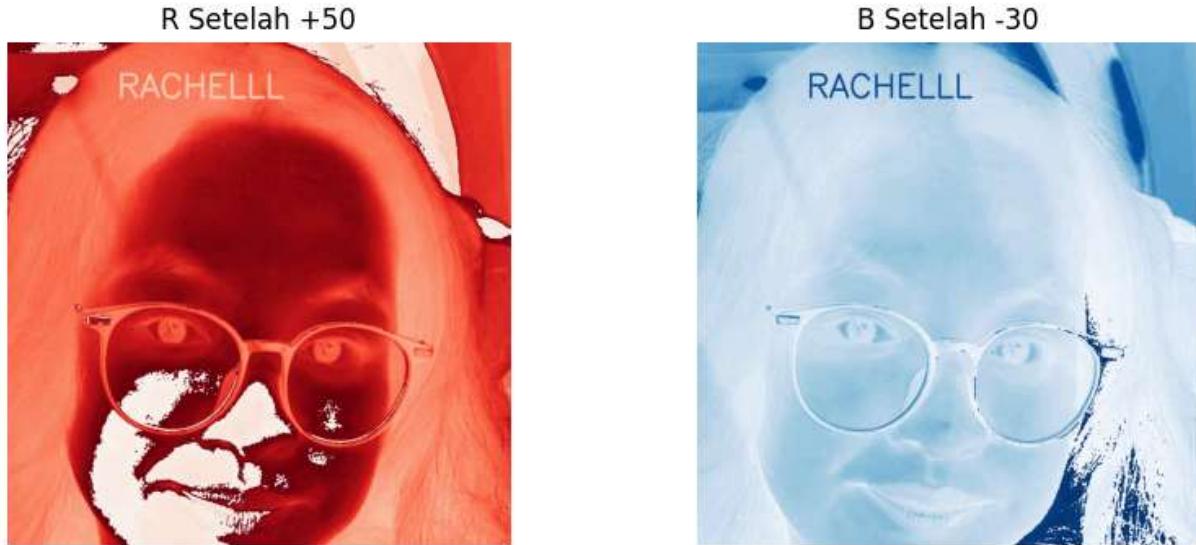
```

plt.imshow(R_mod, cmap="Reds")
plt.title("R Setelah +50")
plt.axis("off")

plt.subplot(1,2,2)
plt.imshow(B_mod, cmap="Blues")
plt.title("B Setelah -30")
plt.axis("off")

```

Out[22]: (-0.5, 919.5, 919.5, -0.5)



Tahap ini menaikkan intensitas merah dan menurunkan intensitas biru untuk mengubah karakter warna gambar.

## Gabungkan kembali channel

```

In [23]: img_mod = np.stack([R_mod, G, B_mod], axis=2).astype(np.uint8)

# Output Tahap 4
plt.imshow(img_mod)
plt.title("Output Tahap 4: Gambar RGB Baru")
plt.axis("off")

```

Out[23]: (-0.5, 919.5, 919.5, -0.5)

### Output Tahap 4: Gambar RGB Baru



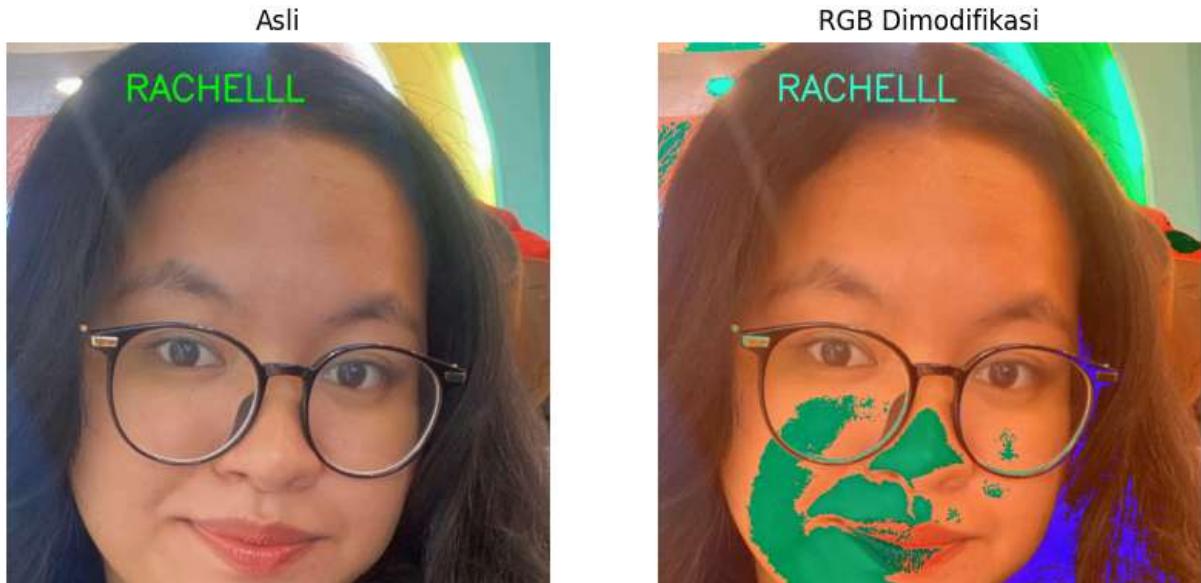
Tahap ini menggabungkan channel yang sudah dimodifikasi menjadi satu gambar RGB baru.

## Tampilkan gambar asli dan hasil modifikasi

```
In [24]: plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
plt.imshow(img)
plt.title("Asli")
plt.axis("off")

plt.subplot(1,2,2)
plt.imshow(img_mod)
plt.title("RGB Dimodifikasi")
plt.axis("off")
```

Out[24]: (-0.5, 919.5, 919.5, -0.5)



Tahap ini menampilkan perbedaan visual antara gambar asli dan gambar yang sudah dimodifikasi.

## Simpan hasil modifikasi

```
In [25]: cv2.imwrite("results_ws4/soal2_modifikasi.png", cv2.cvtColor(img_mod, cv2.COLOR_RGB
```

Out[25]: False

## Histogram per channel

```
In [44]: colors = ['red', 'green', 'blue']
channels_asli = [R, G, B]
channels_mod = [R_mod, G, B_mod]

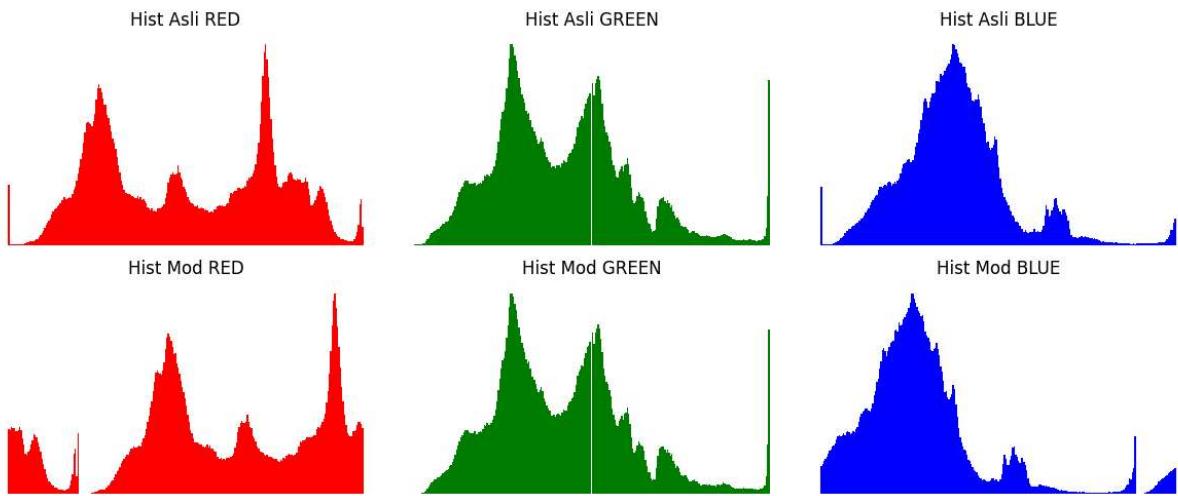
plt.figure(figsize=(12,5))

for i, (ch_asli, ch_mod, c) in enumerate(zip(channels_asli, channels_mod, colors)):

    plt.subplot(2,3,i+1)
    plt.hist(ch_asli.flatten(), bins=256, color=c)
    plt.title("Hist Asli " + c.upper())
    plt.axis("off")

    plt.subplot(2,3,i+4)
    plt.hist(ch_mod.flatten(), bins=256, color=c)
    plt.title("Hist Mod " + c.upper())
    plt.axis("off")

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



## Dampak Perubahan RGB pada Warna Gambar

- Penambahan nilai merah membuat warna kulit dan objek tampak lebih hangat.
- Pengurangan nilai biru mengurangi tone kebiruan sehingga warna bergeser ke arah kuning dan merah.
- Gambar akhir terlihat lebih hangat dan kurang dingin.
- Histogram menunjukkan pergeseran intensitas. Merah naik dan biru turun.

### 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**.

## Baca gambar dan konversi ke RGB

Tahap ini membaca file gambar soal 3 dan mengubahnya ke format RGB agar siap diproses.

```
In [45]: img_bgr_3 = cv2.imread("assets_ws4/soal3.jpg") # ganti nama file jika berbeda
img_rgb_3 = cv2.cvtColor(img_bgr_3, cv2.COLOR_BGR2RGB)

plt.figure(figsize=(6,8))
plt.imshow(img_rgb_3)
```

```
plt.title("Tahap 1 - Gambar RGB Asli")
plt.axis("off")
plt.show()
```

Tahap 1 - Gambar RGB Asli



## Deteksi tepi dengan Canny

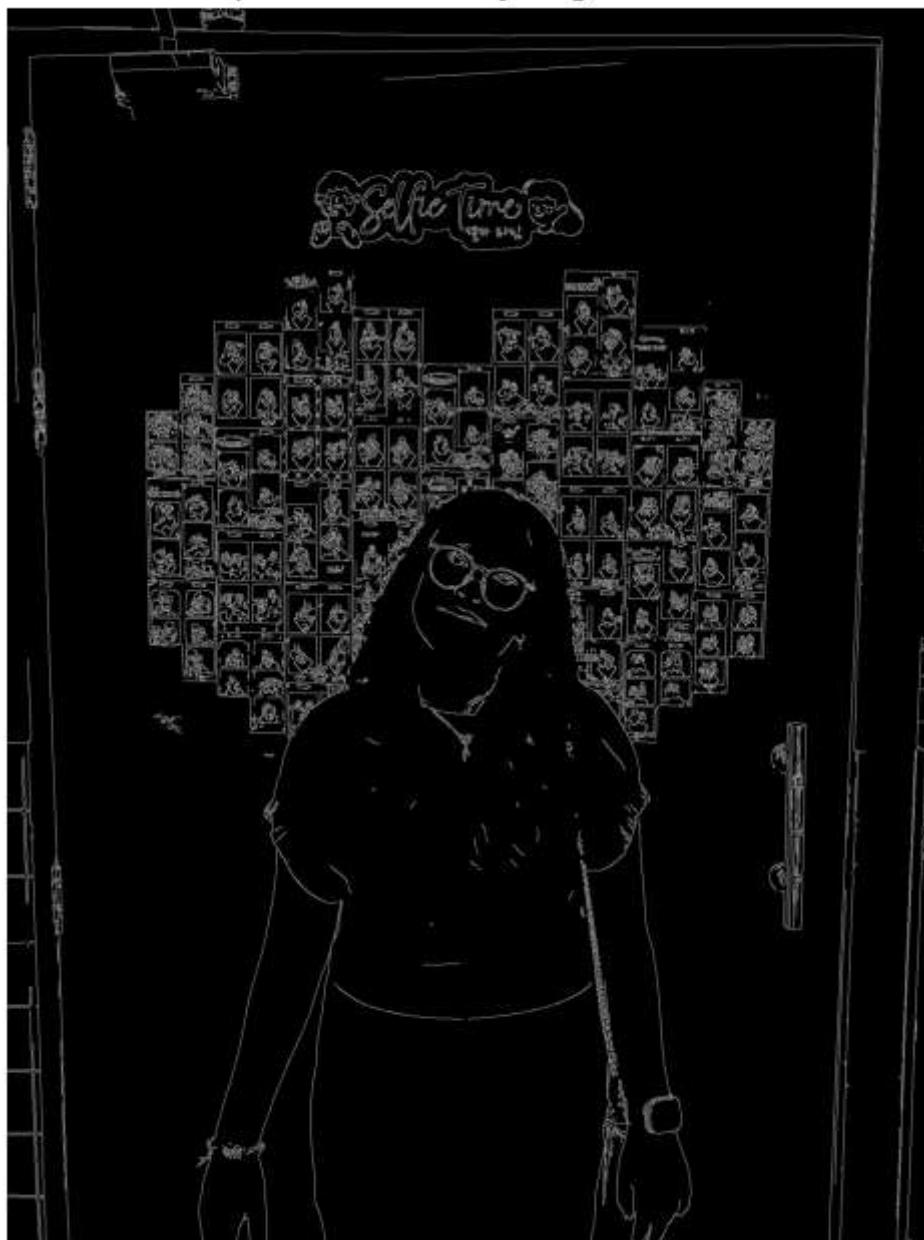
Tahap ini mengubah gambar ke grayscale lalu menerapkan algoritma Canny untuk menonjolkan tepi objek.

```
In [46]: img_gray_3 = cv2.cvtColor(img_rgb_3, cv2.COLOR_RGB2GRAY)
edges_3 = cv2.Canny(img_gray_3, 100, 200)

plt.figure(figsize=(6,8))
```

```
plt.imshow(edges_3, cmap="gray")
plt.title("Tahap 2 - Hasil Canny Edge Detection")
plt.axis("off")
plt.show()
```

Tahap 2 - Hasil Canny Edge Detection



## Thresholding untuk menyisakan objek utama

Tahap ini memberi ambang intensitas sehingga objek utama menjadi putih dan background menjadi hitam.

```
In [47]: _, thresh_3 = cv2.threshold(img_gray_3, 140, 255, cv2.THRESH_BINARY_INV)
```

```
plt.figure(figsize=(6,8))
plt.imshow(thresh_3, cmap="gray")
plt.title("Tahap 3 - Hasil Thresholding")
plt.axis("off")
plt.show()
```

Tahap 3 - Hasil Thresholding



## Cari kontur terbesar dan buat bounding box

Tahap ini mencari kontur dengan area terbesar sebagai objek utama lalu menggambar kotak di sekelilingnya.

```
In [48]: contours_3, _ = cv2.findContours(thresh_3, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

largest_contour_3 = max(contours_3, key=cv2.contourArea)
x3, y3, w3, h3 = cv2.boundingRect(largest_contour_3)

img_bbox_3 = img_rgb_3.copy()
cv2.rectangle(img_bbox_3, (x3, y3), (x3 + w3, y3 + h3), (0, 255, 0), 3)

plt.figure(figsize=(6,8))
plt.imshow(img_bbox_3)
plt.title("Tahap 4 - Bounding Box Objek Utama")
plt.axis("off")
plt.show()
```

## Tahap 4 - Bounding Box Objek Utama



## Terapkan filter blur

Tahap ini mengaburkan detail halus dengan Gaussian blur sehingga gambar menjadi lebih lembut.

```
In [49]: blur_3 = cv2.GaussianBlur(img_rgb_3, (11, 11), 0)
```

```
plt.figure(figsize=(6,8))
plt.imshow(blur_3)
plt.title("Tahap 5 - Hasil Gaussian Blur")
plt.axis("off")
plt.show()
```

## Tahap 5 - Hasil Gaussian Blur



## Terapkan filter sharpening

Tahap ini menajamkan tepi dan detail dengan kernel sharpening sehingga kontras lokal meningkat.

```
In [50]: kernel_sharp_3 = np.array([[0, -1, 0],  
[-1, 5, -1],  
[0, -1, 0]])  
  
sharp_3 = cv2.filter2D(img_rgb_3, -1, kernel_sharp_3)
```

```
plt.figure(figsize=(6,8))
plt.imshow(sharp_3)
plt.title("Tahap 6 - Hasil Sharpening")
plt.axis("off")
plt.show()
```

Tahap 6 - Hasil Sharpening



## Bandingkan asli, blur, dan sharpen

Tahap ini menampilkan tiga gambar sekaligus untuk memudahkan perbandingan efek filter.

In [51]:

```
plt.figure(figsize=(15,5))
plt.subplot(1,3,1)
plt.imshow(img_rgb_3)
plt.title("Asli")
```

```

plt.axis("off")

plt.subplot(1,3,2)
plt.imshow(blur_3)
plt.title("Blur")
plt.axis("off")

plt.subplot(1,3,3)
plt.imshow(sharp_3)
plt.title("Sharpen")
plt.axis("off")

plt.show()

```



```
In [53]: cv2.imwrite("results_ws4/soal3_edges.png", edges_3)
cv2.imwrite("results_ws4/soal3_thresh.png", thresh_3)
cv2.imwrite("results_ws4/soal3_bbox.png", cv2.cvtColor(img_bbox_3, cv2.COLOR_RGB2BG
cv2.imwrite("results_ws4/soal3_blur.png", cv2.cvtColor(blur_3, cv2.COLOR_RGB2BGR))
cv2.imwrite("results_ws4/soal3_sharp.png", cv2.cvtColor(sharp_3, cv2.COLOR_RGB2BGR))
```

Out[53]: True

## Dampak Filter

- Canny menampilkan garis tepi objek sehingga bentuk tubuh dan kontur rambut terlihat lebih jelas.
- Thresholding memisahkan objek dari background dengan mengubah warna menjadi hitam putih sehingga objek utama lebih mudah diambil.
- Blur menghaluskan gambar. Detail kecil seperti tekstur dinding dan rambut menjadi berkurang.
- Sharpen meningkatkan ketajaman. Tepi dan detail objek menjadi lebih jelas.

## 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**).

In [3]:

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

img_4 = cv2.imread("assets_ws4/soal4.jpg") # sesuaikan nama file
img_4_rgb = cv2.cvtColor(img_4, cv2.COLOR_BGR2RGB)

plt.figure(figsize=(6,8))
plt.imshow(img_4_rgb)
plt.title("Tahap 1 - Gambar Asli RGB")
plt.axis("off")
plt.show()
```

## Tahap 1 - Gambar Asli RGB



## Deteksi wajah dengan Haar Cascade

Tahap ini mendeteksi wajah dan memberi kotak di sekelilingnya.

```
In [5]: face_cascade = cv2.CascadeClassifier(  
    cv2.data.haarcascades + "haarcascade_frontalface_default.xml"  
)  
  
gray_4 = cv2.cvtColor(img_4, cv2.COLOR_BGR2GRAY)  
faces = face_cascade.detectMultiScale(gray_4, scaleFactor=1.2, minNeighbors=6)  
  
img_face_box = img_4_rgb.copy()  
  
for (x, y, w, h) in faces:      # baris for
```

```
cv2.rectangle(          # HARUS indent
    img_face_box,
    (x, y),
    (x+w, y+h),
    (0, 255, 0),
    2
)

plt.figure(figsize=(6,8))
plt.imshow(img_face_box)
plt.title("Tahap 2 - Deteksi Wajah")
plt.axis("off")
plt.show()
```

Tahap 2 - Deteksi Wajah



## Hitung posisi overlay (topi sederhana)

Tahap ini memakai posisi kotak wajah untuk menentukan posisi topi di atas kepala.

```
In [9]: landmark_overlay = img_4_rgb.copy()

for (x, y, w, h) in faces:
    hat_h = int(0.4 * h)
    hat_y1 = max(0, y - hat_h)
    hat_y2 = y
    hat_x1 = x
    hat_x2 = x + w

    cv2.rectangle(
        landmark_overlay,
        (hat_x1, hat_y1),
        (hat_x2, hat_y2),
        (200, 0, 0),
        -1
    )

plt.figure(figsize=(6,8))
plt.imshow(landmark_overlay)
plt.title("Tahap 3 - Overlay Topi")
plt.axis("off")
plt.show()
```

### Tahap 3 - Overlay Topi



## Alpha blending topi dan gambar asli

Tahap ini menggabungkan layer topi dengan gambar wajah agar tampak lebih natural.

```
In [11]: result_4 = img_4_rgb.copy()
alpha = 0.6

for (x, y, w, h) in faces:
    hat_h = int(0.4 * h)
    hat_y1 = max(0, y - hat_h)
    hat_y2 = y
    hat_x1 = x
    hat_x2 = x + w
```

```
roi_asli = result_4[hat_y1:hat_y2, hat_x1:hat_x2]
roi_overlay = landmark_overlay[hat_y1:hat_y2, hat_x1:hat_x2]

blended = cv2.addWeighted(roi_overlay, alpha, roi_asli, 1 - alpha, 0)
result_4[hat_y1:hat_y2, hat_x1:hat_x2] = blended

plt.figure(figsize=(6,8))
plt.imshow(result_4)
plt.title("Tahap 4 - Hasil Alpha Blending")
plt.axis("off")
plt.show()
```

Tahap 4 - Hasil Alpha Blending



# Perbandingan asli dan hasil filter

Tahap ini menampilkan perbandingan langsung sebelum dan sesudah diberi filter.

```
In [12]: plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
plt.imshow(img_4_rgb)
plt.title("Asli")
plt.axis("off")

plt.subplot(1,2,2)
plt.imshow(result_4)
plt.title("Dengan Filter Topi")
plt.axis("off")

plt.show()
```

Asli



Dengan Filter Topi



```
In [13]: import os
os.makedirs("results_ws4", exist_ok=True)
cv2.imwrite("results_ws4/soal4_filter_topi.png", cv2.cvtColor(result_4, cv2.COLOR_R
```

```
Out[13]: True
```

## Bagaimana menghitung posisi overlay dan tantangan yang dihadapi selama

# implementasi:

- Posisi topi dihitung dari bounding box wajah (x, y, w, h) hasil deteksi Haar Cascade.
- Topi ditempatkan di atas kepala dengan tinggi sekitar 40 persen dari tinggi wajah dan lebar sama dengan lebar wajah.
- Tantangannya adalah menyesuaikan ukuran topi supaya tidak terlalu besar atau terlalu kecil. Jika deteksi wajah meleset, posisi topi juga ikut bergeser.
- Alpha blending digunakan supaya warna topi menyatu dengan gambar. Nilai alpha 0.6 dipilih agar topi cukup terlihat tapi tidak menutupi detail wajah.

## 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**).

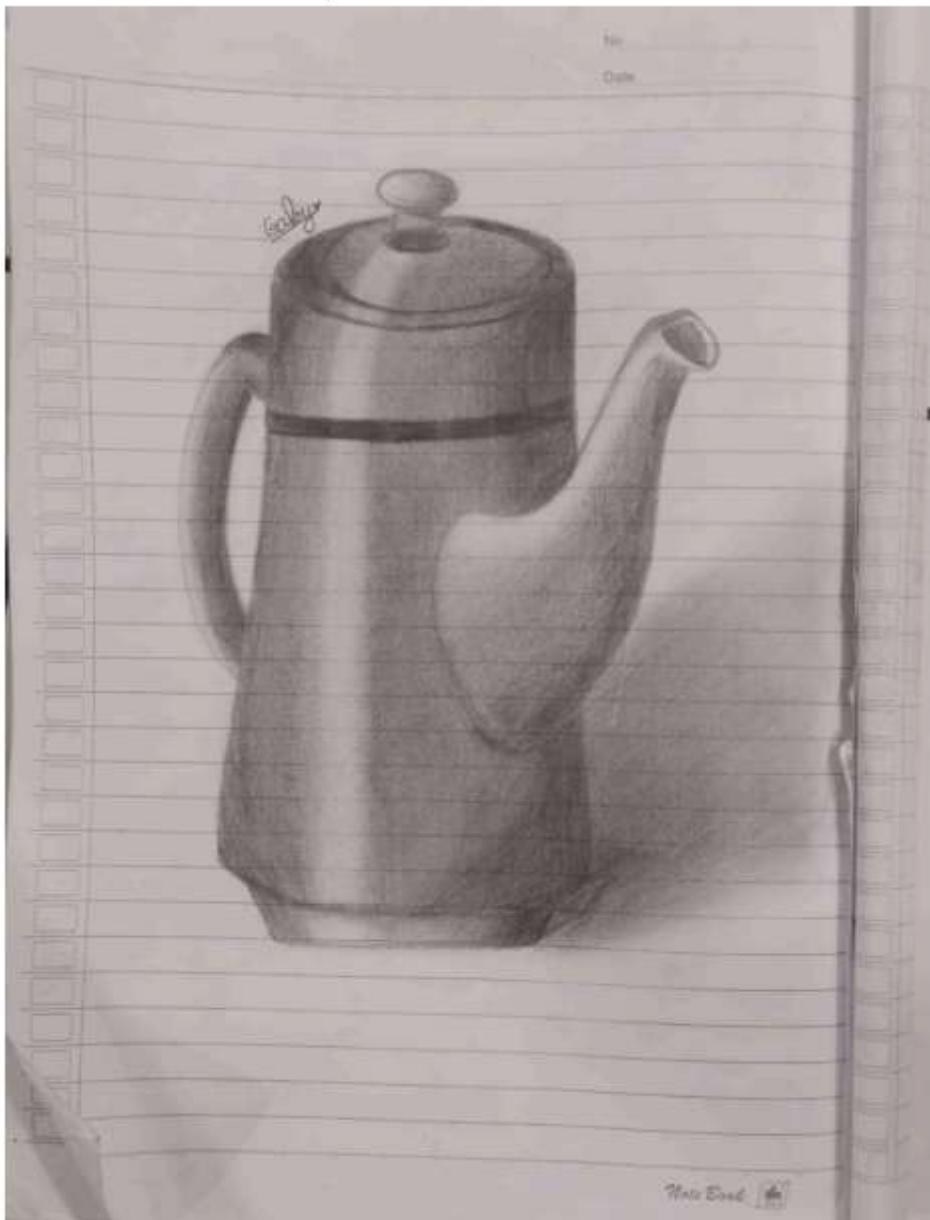
In [2]:

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

# Tahap 1: Baca gambar dan konversi ke RGB
# Caption: Tahap ini membaca gambar karya tangan dan mengubahnya ke RGB agar mudah
img5_bgr = cv2.imread("assets_ws4/soal5.jpg") # ganti nama file jika berbeda
img5_rgb = cv2.cvtColor(img5_bgr, cv2.COLOR_BGR2RGB)

plt.figure(figsize=(6,8))
plt.imshow(img5_rgb)
plt.title("Tahap 1 - Gambar Asli RGB")
plt.axis("off")
plt.show()
```

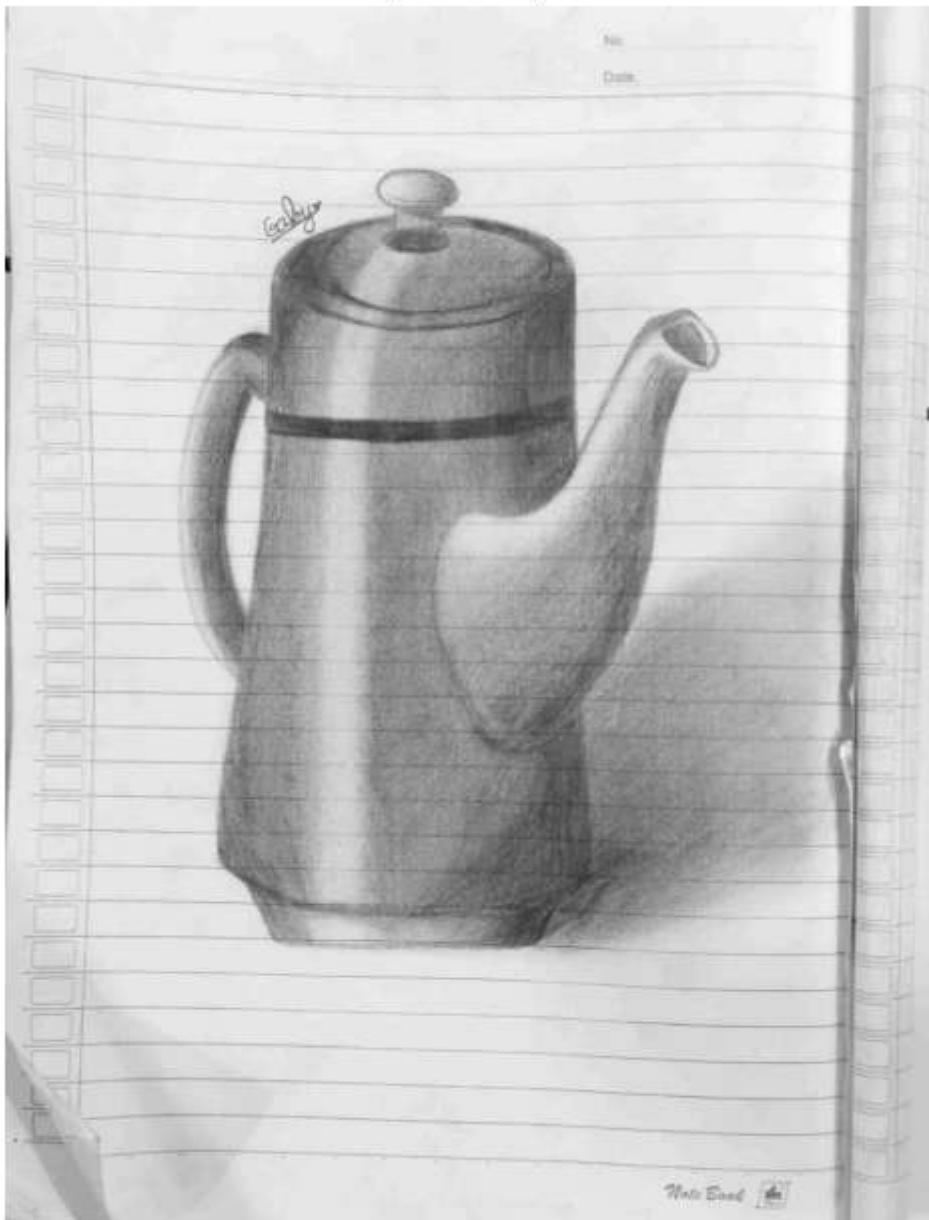
## Tahap 1 - Gambar Asli RGB



```
In [4]: img5_gray = cv2.cvtColor(img5_rgb, cv2.COLOR_RGB2GRAY)

plt.figure(figsize=(6,8))
plt.imshow(img5_gray, cmap="gray")
plt.title("Tahap 2 - Grayscale")
plt.axis("off")
plt.show()
```

## Tahap 2 - Grayscale



## Koreksi perspektif dengan 4 titik manual

- Tahap ini memperbaiki sudut pandang kertas agar terlihat rata dan tidak miring.
- Urutan titik: kiri-atas, kanan-atas, kanan-bawah, kiri-bawah.

```
In [5]: manual_points = np.float32([
    [120, 220],    # kiri atas
    [860, 210],    # kanan atas
    [900, 1120],   # kanan bawah
    [140, 1140]    # kiri bawah
])

width_out = 600
height_out = 900
```

```
dst_pts = np.float32([
    [0, 0],
    [width_out - 1, 0],
    [width_out - 1, height_out - 1],
    [0, height_out - 1]
])

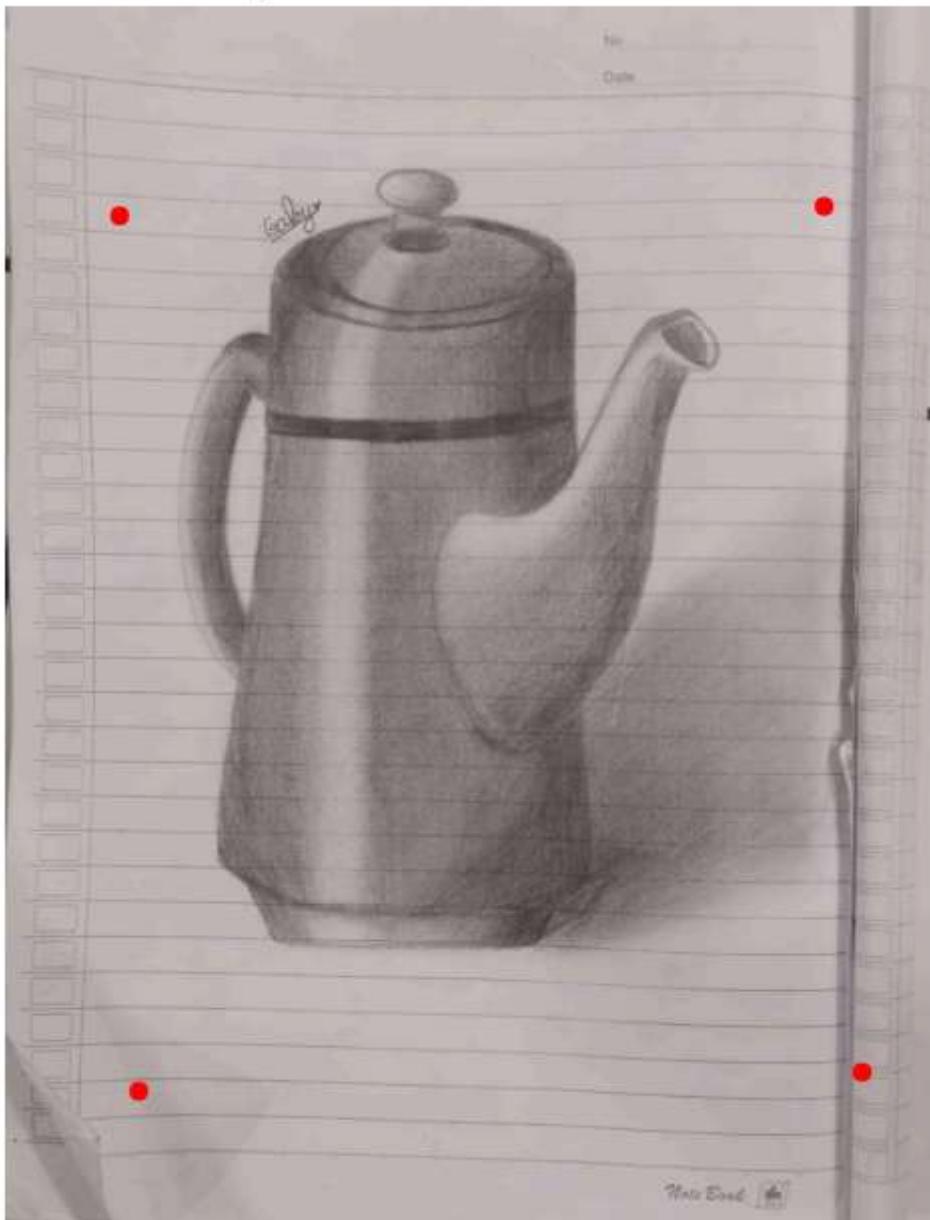
# Visualisasi empat titik manual di gambar asli
img5_pts = img5_rgb.copy()
for (px, py) in manual_points:
    px, py = int(px), int(py)
    cv2.circle(img5_pts, (px, py), 10, (255, 0, 0), -1)

plt.figure(figsize=(6,8))
plt.imshow(img5_pts)
plt.title("Tahap 3 - Titik Manual Sudut Kertas")
plt.axis("off")
plt.show()

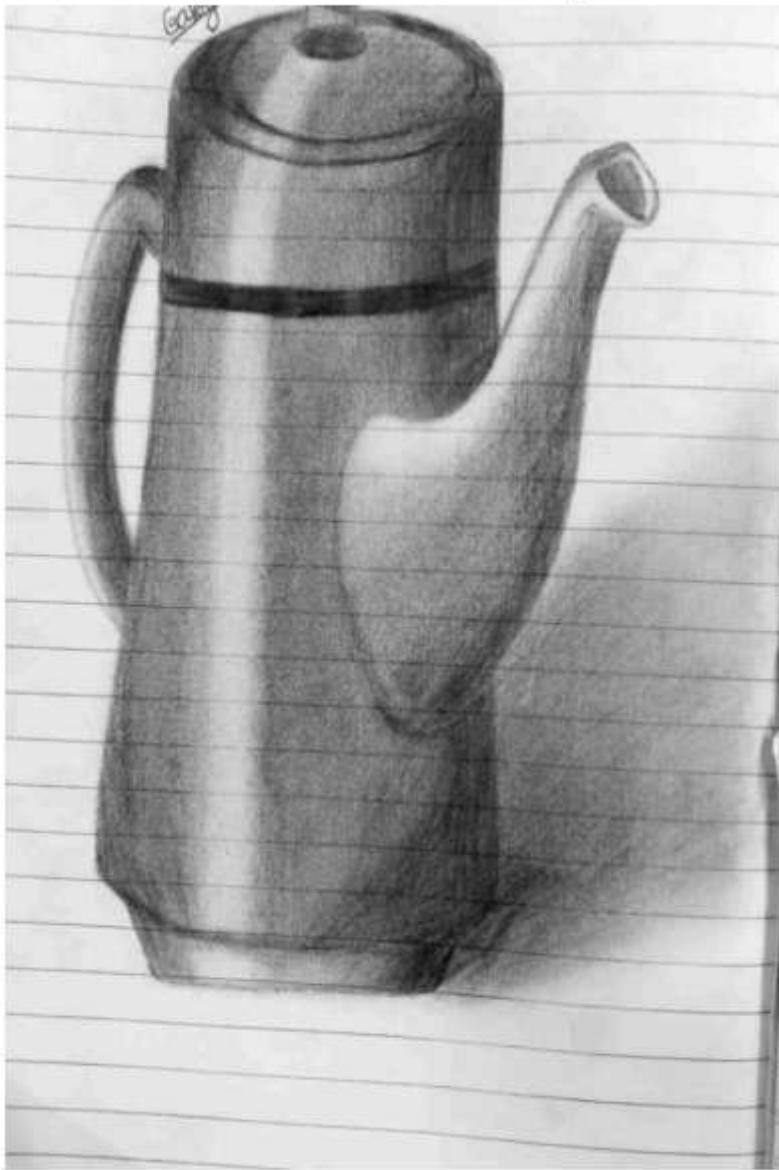
# Transformasi homografi dengan manual_points
M5 = cv2.getPerspectiveTransform(manual_points, dst_pts)
img5_warp = cv2.warpPerspective(img5_gray, M5, (width_out, height_out))

plt.figure(figsize=(5,8))
plt.imshow(img5_warp, cmap="gray")
plt.title("Tahap 3 - Hasil Koreksi Perspektif")
plt.axis("off")
plt.show()
```

### Tahap 3 - Titik Manual Sudut Kertas



### Tahap 3 - Hasil Koreksi Perspektif



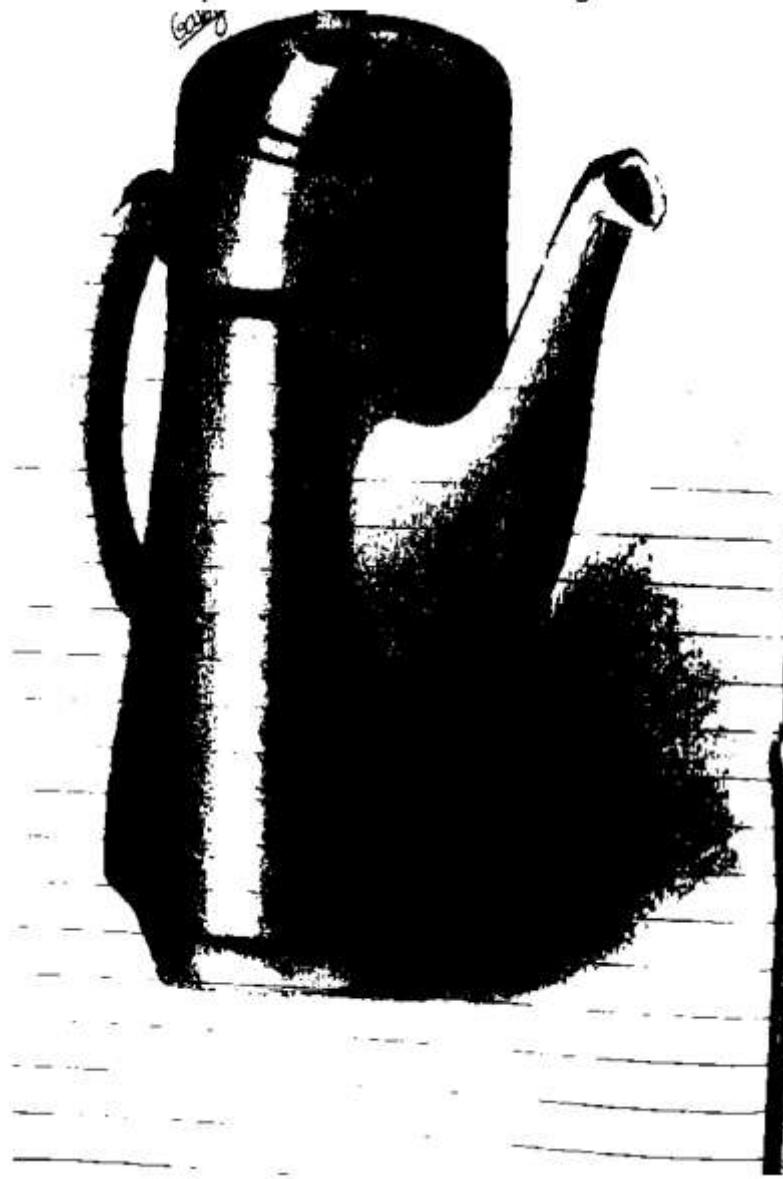
## Thresholding Otsu

Tahap ini menerapkan threshold otomatis Otsu untuk menegaskan objek dan mengurangi noise latar.

```
In [6]: __, img5_otsu = cv2.threshold(img5_warp, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU

plt.figure(figsize=(5,8))
plt.imshow(img5_otsu, cmap="gray")
plt.title("Tahap 4 - Hasil Thresholding Otsu")
plt.axis("off")
plt.show()
```

#### Tahap 4 - Hasil Thresholding Otsu



## Grid perbandingan semua tahap

Tahap ini menampilkan semua hasil dalam satu grid untuk memudahkan perbandingan visual.

```
In [7]: plt.figure(figsize=(12,8))

plt.subplot(2,2,1)
plt.imshow(img5_rgb)
plt.title("Asli RGB")
plt.axis("off")

plt.subplot(2,2,2)
plt.imshow(img5_gray, cmap="gray")
plt.title("Grayscale")
```

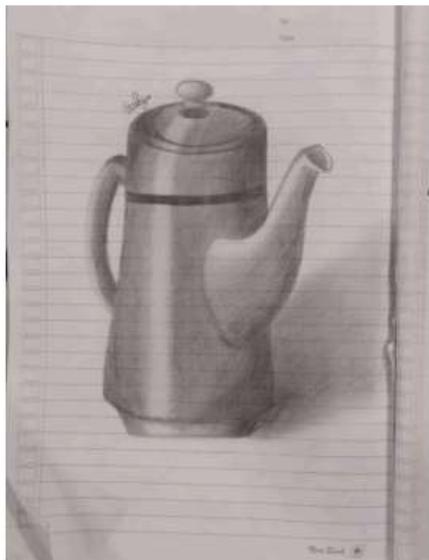
```
plt.axis("off")

plt.subplot(2,2,3)
plt.imshow(img5_warp, cmap="gray")
plt.title("Koreksi Perspektif")
plt.axis("off")

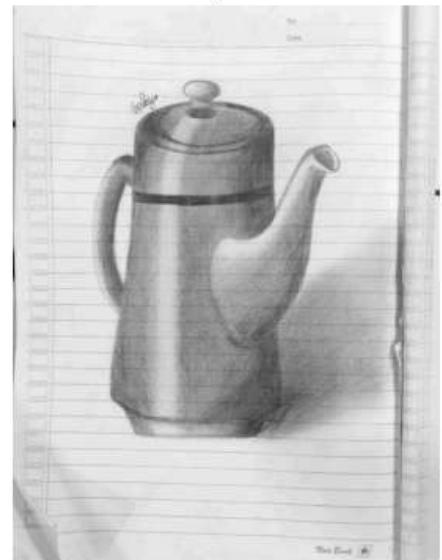
plt.subplot(2,2,4)
plt.imshow(img5_otsu, cmap="gray")
plt.title("Threshold Otsu")
plt.axis("off")

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

Asli RGB



Grayscale



Koreksi Perspektif



Threshold Otsu



```
In [8]: import os
os.makedirs("results_ws4", exist_ok=True)
cv2.imwrite("results_ws4/soal5_gray.png", img5_gray)
```

```
cv2.imwrite("results_ws4/soal5_warp.png", img5_warp)
cv2.imwrite("results_ws4/soal5_otsu.png", img5_otsu)
```

Out[8]: True

# Fungsi Setiap Tahap dan Dampaknya pada Kualitas Citra

## Grayscale

- Tahap ini mengubah citra dari RGB ke grayscale.
- Tujuannya untuk menyederhanakan citra menjadi terang gelap saja sehingga proses selanjutnya lebih mudah dan lebih stabil.
- Perubahan ini menghilangkan gangguan warna dan membuat tepi objek lebih jelas.

## Koreksi Perspektif

- Tahap ini memakai empat titik sudut kertas sebagai acuan homografi.
- Transformasi ini mengubah tampilan miring menjadi tampak datar dan sejajar seperti hasil scan.
- Hasilnya citra lebih rapi dan mudah dibaca.

## Thresholding Otsu

- Tahap ini memilih nilai ambang otomatis dari histogram.
- Teknik ini menyingkirkan latar belakang dan menegaskan garis objek.
- Objek utama jadi lebih kontras dan bersih tanpa menebak nilai threshold secara manual.

## Grid Perbandingan

- Tahap ini menyusun semua hasil dalam satu tampilan.
- Grid memudahkan melihat peningkatan kualitas pada tiap langkah.
- Perbandingan dari asli sampai hasil akhir menunjukkan proses meningkatkan keterbacaan citra.

chatgpt ref :

```
In [14]: import cv2

img = cv2.imread("assets_ws4/gpt.png")

import matplotlib.pyplot as plt

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

plt.imshow(img_rgb)
```

```
plt.axis("off")
plt.show()
```



## Aturan Umum Pengerjaan

- 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 diekspor 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.