# UTS

# PENGOLAHAN CITRA DIGITAL


NAMA : Nanda Revan Saputro

NIM : 202331016

KELAS : F

DOSEN : Dr. Dra. Dwina Kuswardani, M.Kom

NO.PC : 15

ASISTEN : 1. Sasikirana Ramadhanty Setiawan Putri

2. Rizqy Amanda

3. Ridho Chaerullah

4. Sakura Amastasya Salsabila Setiyanto

## INSTITUT TEKNOLOGI PLN

## TEKNIK INFORMATIKA

## 2024/2025

# DAFTAR ISI

DAFTAR ISI ..... 2
BAB I PENDAHULUAN ..... 3
**1.1 Rumusan Masalah** ..... 3
**1.2 Tujuan Masalah** ..... 3
**1.3 Manfaat Masalah** ..... 3
BAB II LANDASAN TEORI ..... 4
**2.1 Pengolahan Citra Digital** ..... 4
**2.2 Deteksi Warna pada Citra** ..... 4
**2.3 Histogram dan Thresholding** ..... 4
**2.4 Perbaikan Citra Backlight** ..... 5
**2.5 Pengolahan Citra Grayscale** ..... 5
**2.6 Segmentasi Warna Otomatis** ..... 6
**2.7 Integrasi Konsep dalam Praktik UTS** ..... 6
BAB III HASIL ..... 7
**Tulisan Nama (Nanda Revan Saputro)** ..... 7
**Gambar Selfie dengan Backlight** ..... 9
**Soal Nomor 1** ..... 11
**Soal Nomor 2** ..... 21
**Soal Nomor 3** ..... 32
BAB IV PENUTUP ..... 44
DAFTAR PUSTAKA ..... 45

# BAB I

# PENDAHULUAN

## 1.1 Rumusan Masalah

Dalam kegiatan praktikum UTS ini, terdapat beberapa permasalahan yang dirumuskan sebagai berikut:

1. Bagaimana cara mendeteksi warna secara efektif pada citra digital?
2. Bagaimana metode histogram dan thresholding dapat digunakan untuk segmentasi gambar?
3. Bagaimana cara memperbaiki citra backlight agar objek utama terlihat jelas?
4. Apa manfaat konversi citra RGB ke grayscale dalam pengolahan citra?
5. Bagaimana proses segmentasi warna otomatis dapat dilakukan secara akurat?.

## 1.2 Tujuan Masalah

Tujuan dari praktikum ini adalah untuk:

1. Mempelajari dan memahami proses deteksi warna pada citra digital.
2. Mengimplementasikan metode histogram dan thresholding untuk segmentasi objek.
3. Melakukan perbaikan citra backlight menggunakan teknik pemrosesan yang tepat.
4. Menganalisis fungsi dan manfaat konversi citra ke grayscale dalam pengolahan lanjutan.
5. Mengembangkan segmentasi warna otomatis untuk mendeteksi dan membedakan objek berdasarkan warna.

## 1.3 Manfaat Masalah

Adapun manfaat yang diperoleh dari praktikum ini antara lain:

**Secara Teoritis:**

- Menambah wawasan mahasiswa tentang konsep dasar pengolahan citra digital.
- Memberikan pemahaman mengenai berbagai teknik deteksi warna dan segmentasi.
- Menjadi dasar ilmiah untuk penelitian atau pengembangan sistem pengolahan citra di masa depan.

**Secara Praktis:**

- Mahasiswa mampu mengaplikasikan konsep-konsep pengolahan citra dalam bentuk kode program (misalnya menggunakan Python/OpenCV).
- Mahasiswa dapat menangani masalah pencahayaan seperti backlight secara mandiri.
- Mahasiswa memperoleh pengalaman langsung dalam melakukan analisis citra secara sistematis.

# BAB II

# LANDASAN TEORI

## 2.1 Pengolahan Citra Digital

Pengolahan citra digital (Digital Image Processing) adalah salah satu cabang ilmu komputer yang berfokus pada pengolahan dan analisis citra digital dengan bantuan perangkat lunak atau algoritma tertentu. Pengolahan ini bertujuan untuk meningkatkan kualitas citra, mengekstrak informasi, melakukan segmentasi, klasifikasi objek, hingga analisis lanjutan seperti pengenalan pola dan pengambilan keputusan secara otomatis. Citra digital sendiri merupakan representasi visual dari suatu objek dalam bentuk piksel-piksel digital yang memiliki nilai intensitas tertentu.

Dalam konteks tugas UTS ini, pengolahan citra melibatkan beberapa proses utama yang saling berkaitan, yaitu: deteksi warna, histogram dan thresholding, perbaikan gambar backlight, serta konversi citra ke grayscale dan segmentasi warna.

## 2.2 Deteksi Warna pada Citra

Deteksi warna merupakan teknik dasar dalam pengolahan citra yang digunakan untuk mengidentifikasi piksel tertentu berdasarkan nilai warnanya. Warna pada citra digital biasanya direpresentasikan dalam ruang warna RGB, HSV, atau ruang warna lain seperti YCbCr, CMYK, dan Lab. Teknik ini penting dalam proses segmentasi, pelacakan objek (object tracking), dan pengenalan objek (object recognition).

Secara umum, metode deteksi warna bekerja dengan cara membandingkan nilai piksel terhadap rentang warna yang telah ditentukan. Misalnya, untuk mendeteksi warna merah, maka sistem akan mencari piksel yang memiliki komponen R (Red) tinggi, sedangkan komponen G dan B-nya rendah. Teknik ini banyak digunakan dalam aplikasi robotik, augmented reality, hingga sistem pengawasan video.

Menurut Boldaji dan Semnani (2021), segmentasi warna dapat dilakukan dengan strategi thresholding berbasis histogram yang membantu memisahkan objek dari latar belakang berdasarkan intensitas warna. Teknik ini memungkinkan pemrosesan yang cepat dan efisien, terutama untuk gambar dengan latar belakang seragam atau warna dominan yang kontras.

Selain itu, ruang warna HSV juga sering digunakan karena lebih mendekati persepsi warna manusia, terutama dalam kondisi pencahayaan yang bervariasi. HSV memisahkan informasi warna (Hue) dari intensitas cahaya (Value), sehingga lebih tahan terhadap perubahan pencahayaan..

## 2.3 Histogram dan Thresholding

Histogram citra merupakan alat analisis yang menggambarkan frekuensi relatif dari masing-masing nilai intensitas dalam sebuah citra. Dengan kata lain, histogram menunjukkan seberapa sering nilai piksel tertentu muncul dalam gambar. Histogram sangat penting untuk mengetahui kualitas kontras suatu gambar, tingkat kecerahan, dan distribusi intensitas.

Sementara itu, thresholding adalah teknik segmentasi yang membagi citra menjadi dua bagian: objek dan latar belakang, dengan menggunakan nilai ambang batas tertentu. Nilai threshold dapat ditentukan secara manual maupun otomatis melalui metode-metode seperti Otsu, adaptive thresholding, atau metode berbasis histogram lainnya.

Barron (2020) dalam penelitiannya memperkenalkan metode Generalized Histogram Thresholding (GHT), yang memadukan prinsip dari metode Otsu dan Minimum Error Thresholding. Metode ini sangat berguna dalam aplikasi nyata karena mampu menyesuaikan nilai threshold dengan variasi pencahayaan dan noise, serta mampu mempertahankan akurasi segmentasi meskipun dalam kondisi gambar yang kompleks.

Histogram juga digunakan untuk keperluan peningkatan kontras, seperti teknik histogram equalization, yang berfungsi untuk mendistribusikan nilai intensitas piksel agar kontras gambar menjadi lebih baik dan detail objek lebih terlihat..

**2.4 Perbaikan Citra Backlight**

Citra backlight adalah citra yang mengalami pencahayaan dari belakang objek utama, sehingga objek utama menjadi gelap dan tidak terlihat jelas. Permasalahan ini sering terjadi ketika mengambil gambar di luar ruangan dengan sumber cahaya seperti matahari berada di belakang objek. Untuk mengatasi hal ini, digunakan teknik pemrosesan seperti kontras adaptif, pencahayaan lokal, atau algoritma Retinex.

Retinex adalah metode pemrosesan yang meniru cara kerja sistem visual manusia dalam mengoreksi pencahayaan dan mempertahankan persepsi warna. Metode ini bekerja dengan memisahkan pencahayaan (illumination) dan reflektansi (reflectance), sehingga dapat memperjelas objek tanpa mengubah warna aslinya secara drastis.

Nnolim (2017) dalam penelitiannya mengembangkan metode Retinex berbasis difusi anisotropik, yang tidak hanya memperbaiki pencahayaan, tetapi juga menjaga tepi dan struktur halus pada gambar. Ini penting karena perbaikan backlight yang agresif sering kali menyebabkan hilangnya detail objek atau munculnya artefak.

Dalam aplikasi praktis, metode ini sangat bermanfaat untuk citra wajah atau citra manusia secara umum, agar hasil akhir tetap natural dan informatif. Selain itu, teknik ini juga digunakan dalam dunia fotografi, CCTV, dan pengenalan wajah (face recognition) untuk meningkatkan kualitas deteksi.

**2.5 Pengolahan Citra Grayscale**

Konversi citra dari RGB ke grayscale merupakan tahap penting dalam pengolahan citra. Citra grayscale hanya mengandung informasi luminansi (intensitas cahaya) tanpa memperhitungkan warna. Dengan mengurangi dimensi data dari 3 channel (RGB) menjadi 1 channel (grayscale), maka proses komputasi menjadi lebih efisien dan analisis menjadi lebih sederhana.

Grayscale banyak digunakan sebagai pra-proses untuk operasi pengenalan tepi, segmentasi objek, filtering, dan transformasi spasial. Nilai intensitas dalam citra grayscale biasanya dinyatakan dalam skala 0–255, dengan 0 sebagai hitam pekat dan 255 sebagai putih terang.

Dalam studi yang dilakukan oleh Rani et al. (2023), kombinasi antara grayscale conversion dan peningkatan kontras adaptif seperti Adaptive Gamma Correction terbukti efektif dalam memperjelas detail pada citra yang memiliki kualitas rendah atau terkena noise pencahayaan. Hal ini menunjukkan bahwa grayscale bukan sekadar penyederhanaan visual, tetapi juga dapat meningkatkan hasil akhir dari analisis citra lanjutan.

**2.6 Segmentasi Warna Otomatis**

Segmentasi warna adalah proses memisahkan objek dalam citra berdasarkan perbedaan warnanya. Proses ini penting untuk mempermudah analisis lebih lanjut seperti deteksi objek, pelacakan, klasifikasi, atau ekstraksi fitur. Segmentasi dapat dilakukan dalam berbagai ruang warna, dan algoritma thresholding sering menjadi teknik utamanya.

Dalam penelitian terbaru oleh Cao et al. (2023), mereka mengembangkan metode segmentasi otomatis dua tahap menggunakan histogram. Pendekatan ini tidak memerlukan pelabelan manual atau data pelatihan, sehingga cocok untuk sistem yang ingin bekerja secara mandiri (unsupervised learning). Metode mereka terbukti mampu beradaptasi dalam berbagai kondisi latar belakang yang kompleks dan pencahayaan tidak merata.

Metode ini sangat bermanfaat dalam aplikasi dunia nyata seperti pemantauan lalu lintas, deteksi objek dalam drone, pengolahan citra satelit, hingga sistem pertanian berbasis kamera otomatis.

**2.7 Integrasi Konsep dalam Praktik UTS**

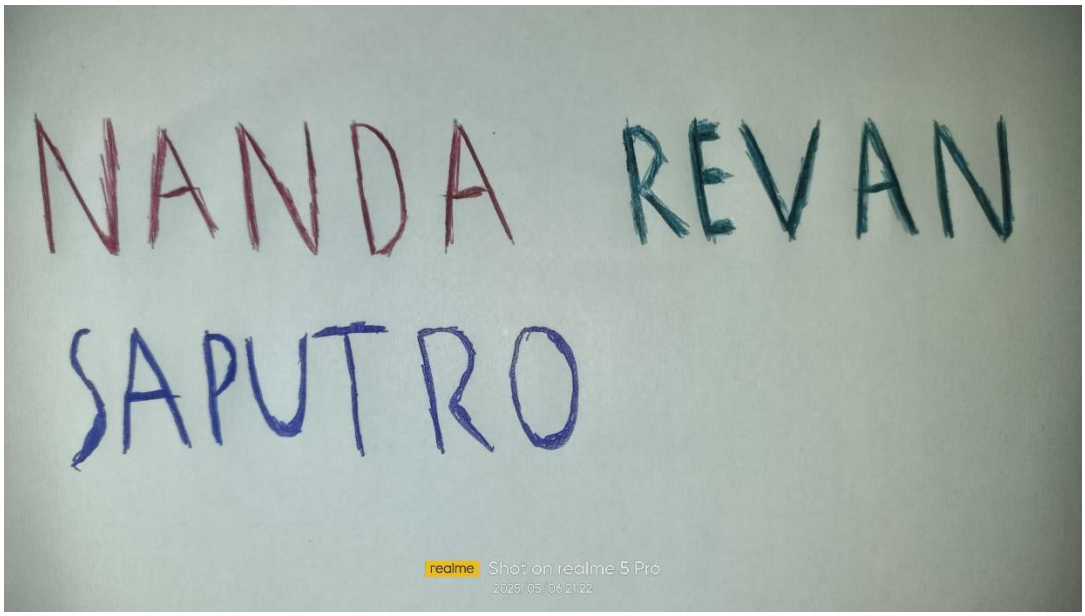
Dalam praktik UTS ini, seluruh konsep di atas diterapkan secara sinergis. Misalnya, proses deteksi warna dilakukan terlebih dahulu untuk menemukan bagian penting dari citra. Setelah itu, histogram dan thresholding digunakan untuk menyegmentasi citra menjadi objek dan latar belakang. Jika pencahayaan tidak memadai (seperti backlight), maka diterapkan metode perbaikan pencahayaan. Kemudian, dilakukan konversi ke grayscale untuk tahap analisis bentuk atau tekstur.

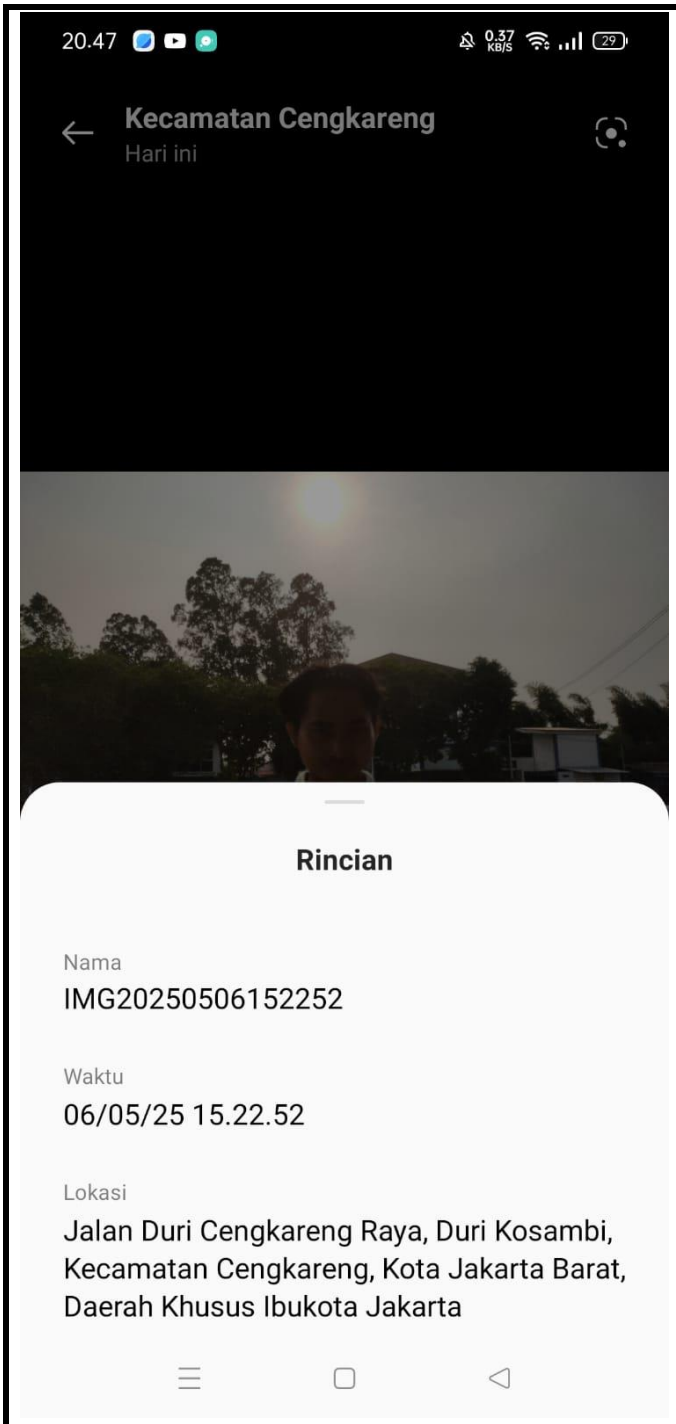
Dengan pemahaman teori yang kuat dan implementasi praktis yang tepat, maka hasil pengolahan citra dapat lebih optimal dan sesuai tujuan. Selain itu, penggabungan teori dan praktik ini juga menunjukkan pentingnya dasar ilmiah dalam pengembangan sistem berbasis citra, yang tidak hanya bekerja secara visual, tetapi juga akurat dan efisien.

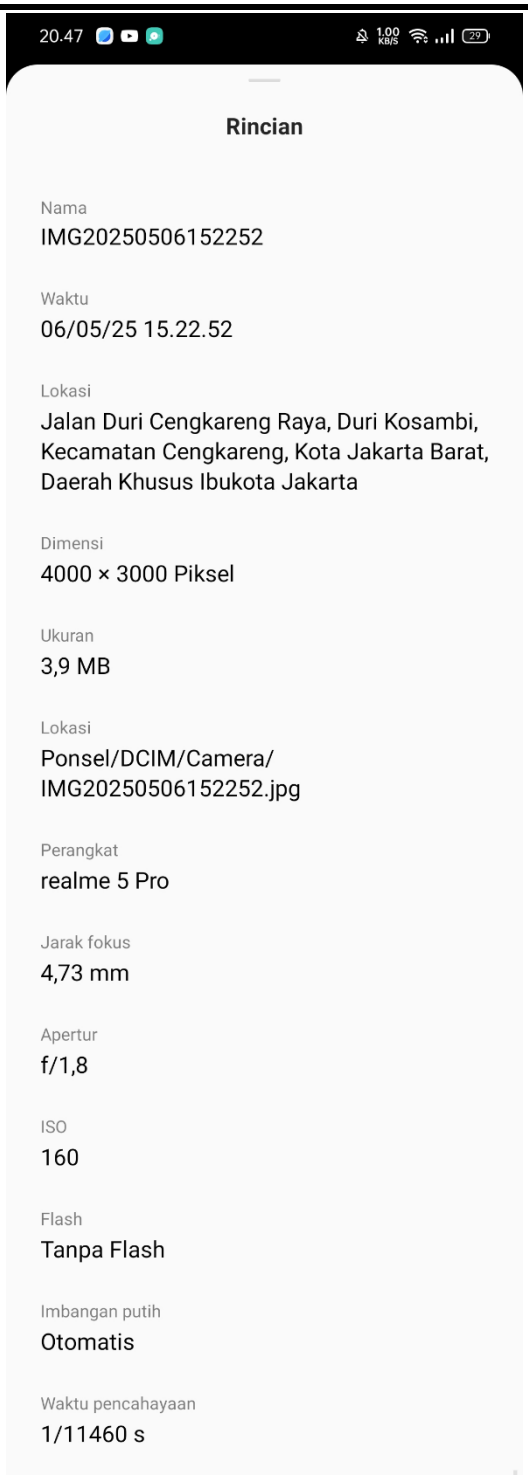
# BAB III

# HASIL

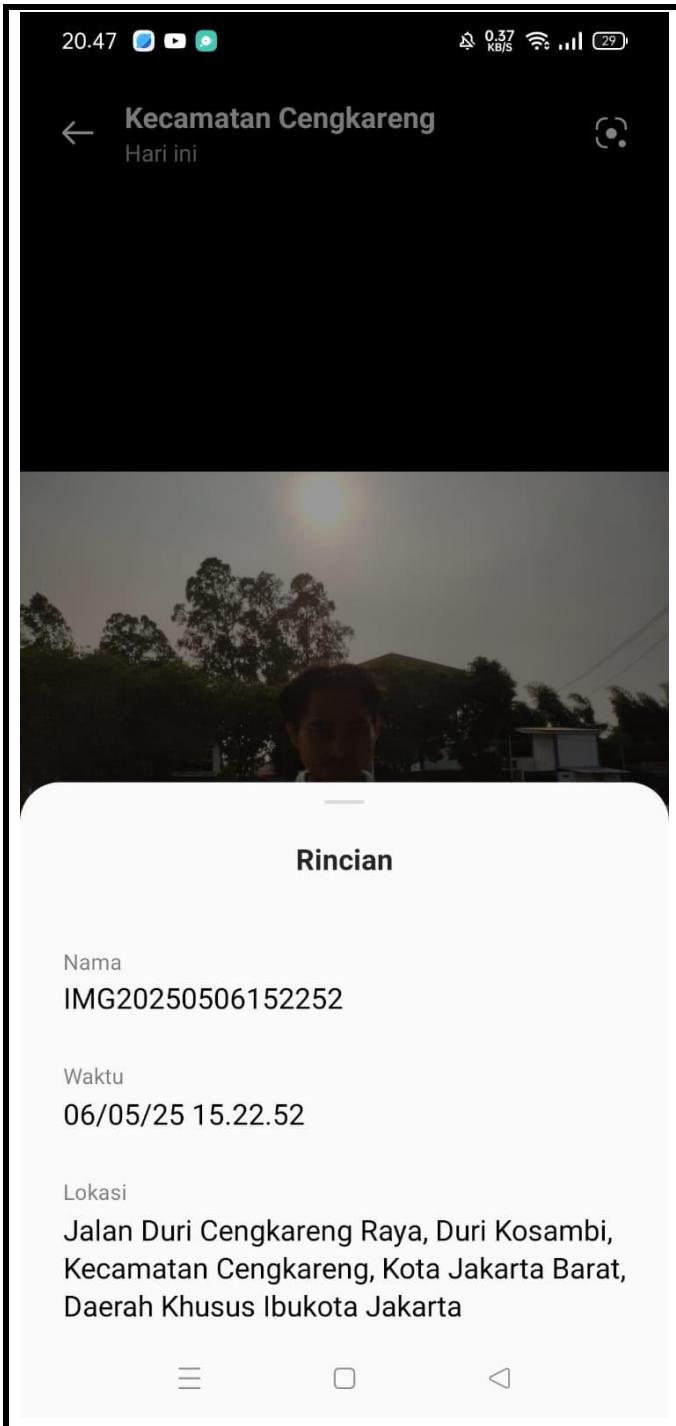
**Tulisan Nama (Nanda Revan Saputro)**

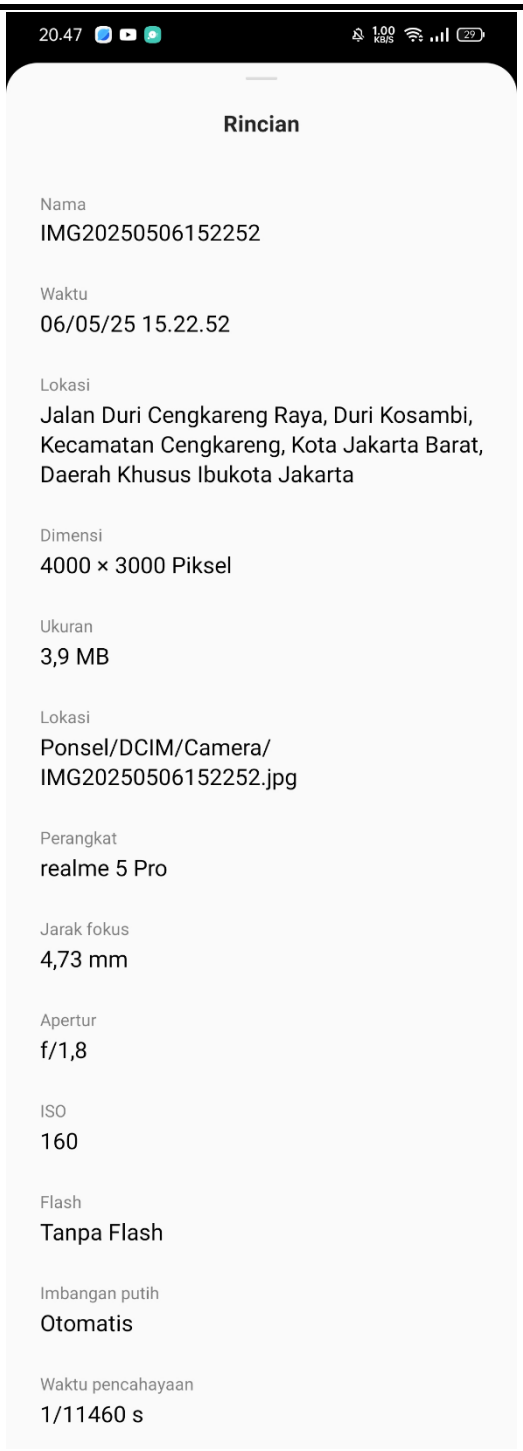



20.47
Kecamatan Cengkareng
Hari ini
Rincian
Nama
IMG20250506152252
Waktu
06/05/25 15.22.52
Lokasi
Jalan Duri Cengkareng Raya, Duri Kosambi, Kecamatan Cengkareng, Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta


20.47
Rincian
Nama
IMG20250506152252
Waktu
06/05/25 15.22.52
Lokasi
Jalan Duri Cengkareng Raya, Duri Kosambi, Kecamatan Cengkareng, Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta
Dimensi
4000 × 3000 Piksel
Ukuran
3,9 MB
Lokasi
Ponsel/DCIM/Camera/ IMG20250506152252.jpg
Perangkat
realme 5 Pro
Jarak fokus
4,73 mm
Apertur
f/1,8
ISO
160
Flash
Tanpa Flash
Imbangan putih
Otomatis
Waktu pencahayaan
1/11460 s

**Gambar Selfie dengan Backlight**


20.47
Kecamatan Cengkareng
Hari ini
Rincian
Nama
IMG20250506152252
Waktu
06/05/25 15.22.52
Lokasi
Jalan Duri Cengkareng Raya, Duri Kosambi, Kecamatan Cengkareng, Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta


20.47
Rincian
Nama
IMG20250506152252
Waktu
06/05/25 15.22.52
Lokasi
Jalan Duri Cengkareng Raya, Duri Kosambi, Kecamatan Cengkareng, Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta
Dimensi
4000 × 3000 Piksel
Ukuran
3,9 MB
Lokasi
Ponsel/DCIM/Camera/ IMG20250506152252.jpg
Perangkat
realme 5 Pro
Jarak fokus
4,73 mm
Apertur
f/1,8
ISO
160
Flash
Tanpa Flash
Imbangan putih
Otomatis
Waktu pencahayaan
1/11460 s

**Penjelasan Kodingan dan Langkah-langkah:**

**Soal Nomor 1**

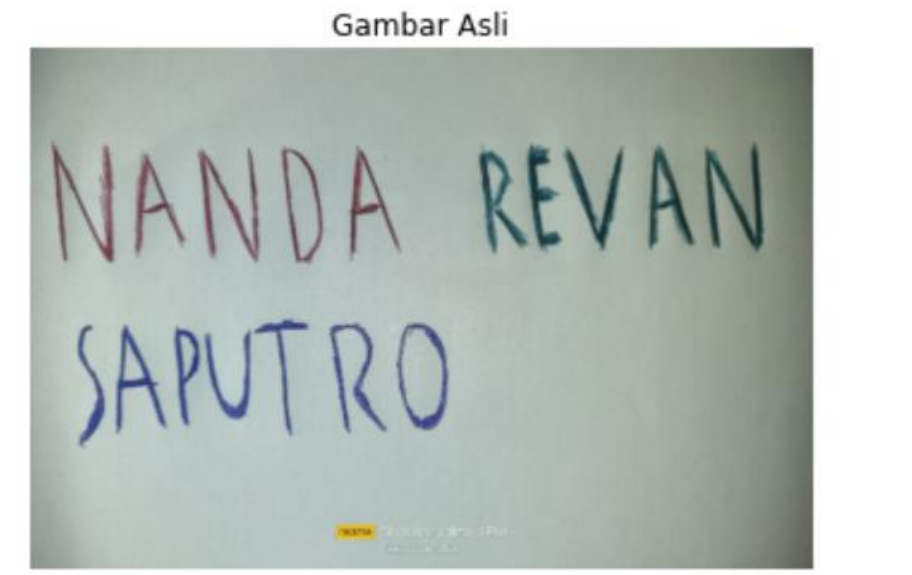
In [2]:
```
import cv2
import matplotlib.pyplot as plt
import numpy as np

#202331016_Nanda Revan Saputro
```

**Baris ini mengimpor pustaka yang dibutuhkan**

- cv2 untuk pengolahan citra menggunakan OpenCV.
- matplotlib.pyplot sebagai alat untuk menampilkan gambar dan grafik.
- numpy digunakan untuk membuat array numerik seperti ambang batas warna.

In [3]:
```
# Load gambar dan resize agar lebih kecil
image = cv2.imread('tulisan7.jpg')
resized_image = cv2.resize(image, (600, 400))

# Konversi ke RGB untuk visualisasi
rgb_image = cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB)

plt.imshow(rgb_image)
plt.title("Gambar Asli")
plt.axis('off')
plt.show()


#202331016_Nanda Revan Saputro
```

**Output:**



**Penjelasan:**

- image = cv2.imread('tulisan7.jpg')

Digunakan untuk membaca gambar dari file bernama tulisan7.jpg. Gambar dibaca dalam format warna BGR (Blue-Green-Red), yaitu format warna standar pada OpenCV.

- resized_image = cv2.resize(image, (600, 400))

Gambar hasil pembacaan diubah ukurannya menjadi 600 piksel lebar dan 400 piksel tinggi. Tujuannya agar lebih ringan diproses dan ditampilkan (apalagi kalau gambar aslinya besar).

- rgb_image = cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB)

Karena OpenCV membaca gambar dalam format BGR, sedangkan matplotlib.pyplot.imshow() menggunakan format RGB, maka gambar perlu dikonversi ke format warna RGB terlebih dahulu agar warna tampil dengan benar.

- Visualisasi dengan Matplotlib:
  - plt.imshow(rgb_image): Menampilkan gambar berwarna RGB.
  - plt.title("Gambar Asli"): Memberi judul tampilan gambar.
  - plt.axis('off'): Menyembunyikan garis sumbu (x dan y) supaya tampilan bersih.
  - plt.show(): Memunculkan tampilan gambar ke layar.

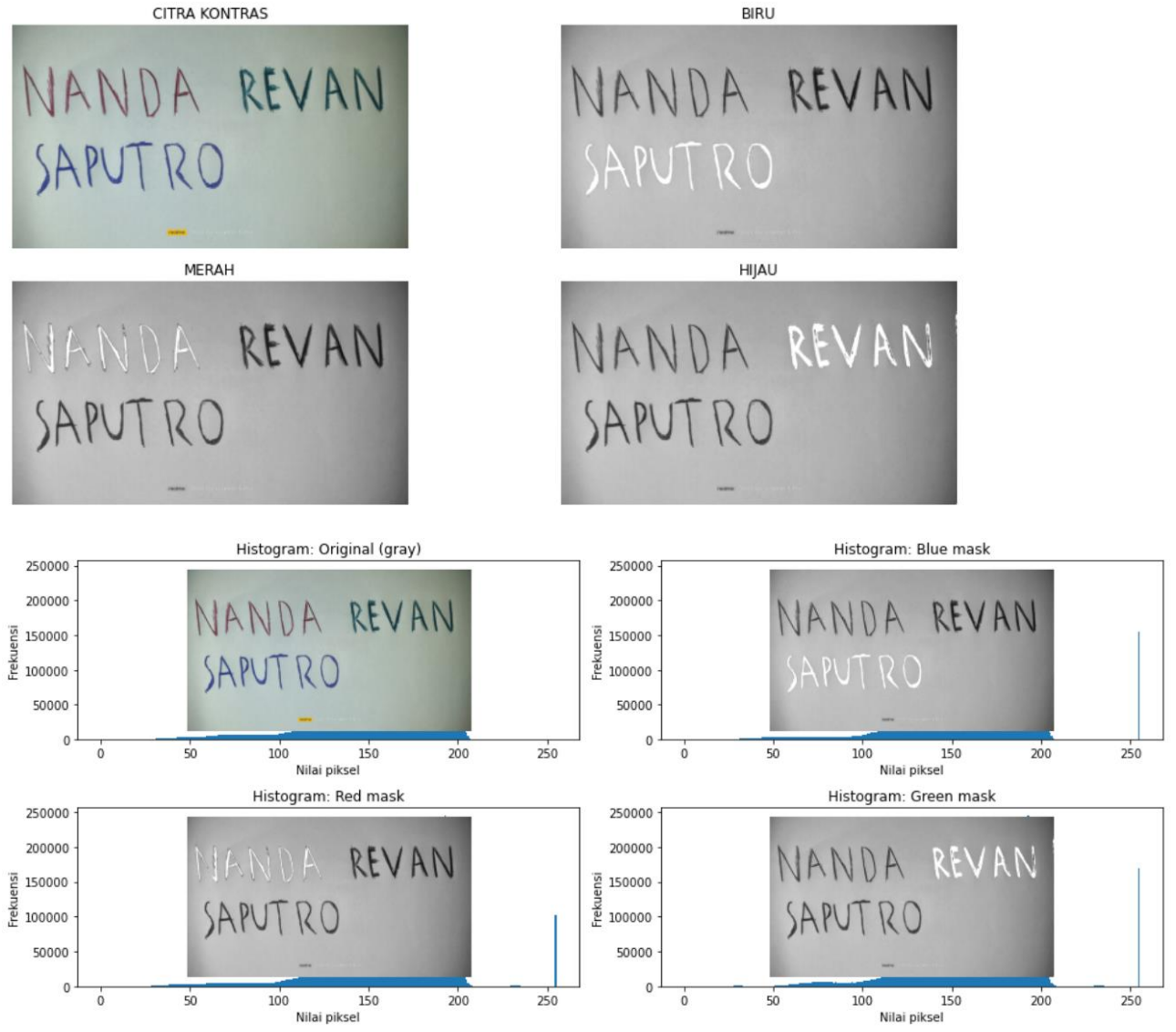
## Menampilkan Warna 202331016_Nanda Revan Saputro

```
# 1. Baca & konversi
img_bgr = cv2.imread('tulisan7.jpg')
if img_bgr is None:
    raise FileNotFoundError("tulisan7.jpg tidak ditemukan di direktori kerja!")
img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)

# 2. Grayscale & HSV
gray = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2GRAY)
hsv  = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2HSV)

# 3. Rentang HSV untuk mask
ranges = {
    'blue'  : (np.array([100,  50,  50]), np.array([140, 255, 255])),
    'green' : (np.array([ 30,  40,  40]), np.array([100, 255, 255])),
    'red1'  : (np.array([  0,  50,  50]), np.array([ 10, 255, 255])),
    'red2'  : (np.array([160,  50,  50]), np.array([180, 255, 255])),
}
mask_blue  = cv2.inRange(hsv, *ranges['blue'])
mask_green = cv2.inRange(hsv, *ranges['green'])
mask_red   = cv2.inRange(hsv, *ranges['red1']) + cv2.inRange(hsv, *ranges['red2'])

# 4. Fungsi highlight teks
def highlight(gray_img, mask):
    out = gray_img.copy()
    out[mask > 0] = 255
    return out
```

```
h_blue  = highlight(gray, mask_blue)
h_green = highlight(gray, mask_green)
h_red   = highlight(gray, mask_red)

# 5. Tampilkan empat citra
fig, axs = plt.subplots(2, 2, figsize=(14, 6))
axs[0,0].imshow(img_rgb);         axs[0,0].set_title('CITRA KONTRAS'); axs[0,0].axis('off')
axs[0,1].imshow(h_blue,  cmap='gray'); axs[0,1].set_title('BIRU');       axs[0,1].axis('off')
axs[1,0].imshow(h_red,   cmap='gray'); axs[1,0].set_title('MERAH');      axs[1,0].axis('off')
axs[1,1].imshow(h_green, cmap='gray'); axs[1,1].set_title('HIJAU');      axs[1,1].axis('off')
plt.tight_layout()
plt.show()

# 6. Plot histogram dengan inset yang semakin besar
mapping = [
    ('Histogram: Original (gray)', gray,      img_rgb),
    ('Histogram: Blue mask',       h_blue,   h_blue),
    ('Histogram: Red mask',        h_red,    h_red),
    ('Histogram: Green mask',      h_green,  h_green),
]
positions = [(0,0), (0,1), (1,0), (1,1)]
```

```
fig, axs = plt.subplots(2, 2, figsize=(14, 6))
for (title, data, inset_img), (r, c) in zip(mapping, positions):
    ax = axs[r][c]
    ax.hist(data.ravel(), bins=256, range=(0,255))
    ax.set_title(title)
    ax.set_xlabel('Nilai piksel')
    ax.set_ylabel('Frekuensi')
    # Inset sangat besar: almost full subplot but keep margin
    axins = ax.inset_axes([0.05, 0.05, 0.9, 0.9])
    if inset_img.ndim == 2:
        axins.imshow(inset_img, cmap='gray')
    else:
        axins.imshow(inset_img)
    axins.axis('off')

plt.tight_layout()
plt.show()

#202331016_Nanda Revan Saputro
```

**Output:**



**Penjelasan:**

Langkah 1: Membaca dan Mengonversi Gambar

- Pertama, gambar tulisan7.jpg dibaca menggunakan OpenCV dengan fungsi cv2.imread().
- Jika gambar tidak ditemukan (misalnya nama file salah atau file tidak ada), akan muncul error dengan pesan "tulisan7.jpg tidak ditemukan".
- Gambar yang dibaca secara default berada dalam format BGR (Blue-Green-Red), tetapi untuk keperluan visualisasi dengan matplotlib, gambar harus dikonversi ke format RGB.
- Selain itu, gambar juga dikonversi ke format Grayscale agar mudah dianalisis dengan satu channel, dan ke format HSV (Hue, Saturation, Value) yang lebih efektif dalam mendeteksi warna dibandingkan RGB.

Langkah 2: Menentukan Rentang Warna (HSV)

- Rentang warna untuk masing-masing kategori (biru, hijau, merah) ditentukan berdasarkan ruang warna HSV. Di HSV, nilai Hue mewakili warna, Saturation mewakili kejenuhan warna, dan Value mewakili kecerahan.
- Rentang untuk warna biru berada pada nilai Hue 100 hingga 140, dengan Saturation dan Value yang cukup tinggi untuk menangkap warna biru pada gambar.
- Warna hijau ditangkap antara Hue 30 hingga 100, dengan Saturation dan Value yang sesuai.
- Untuk warna merah, ada dua rentang Hue yang digunakan: satu pada rentang 0 hingga 10, dan satu lagi di rentang 160 hingga 180, karena skala Hue di HSV hanya mencapai 180, jadi merah yang terletak di sekitar 0 derajat juga perlu ditangkap dalam rentang ini.

Langkah 3: Membuat Mask Berdasarkan Warna

- Fungsi cv2.inRange() digunakan untuk membuat mask berdasarkan rentang warna yang telah ditentukan.
- Setiap mask akan menghasilkan gambar biner, di mana area yang cocok dengan warna yang ditentukan akan berwarna putih (255) dan area yang tidak cocok akan berwarna hitam (0).
- Mask biru, hijau, dan merah dihasilkan berdasarkan rentang warna yang sudah ditentukan sebelumnya, dengan hasil akhir berupa area-area yang relevan dengan warna yang diinginkan.

Langkah 4: Fungsi Highlight

- Fungsi highlight() digunakan untuk menyoroti area dengan warna tertentu di gambar grayscale.
- Fungsi ini membuat salinan dari gambar grayscale dan kemudian mengganti piksel yang sesuai dengan warna yang terdeteksi menjadi putih (255), sementara sisanya tetap gelap.

- Tujuan dari fungsi ini adalah untuk menyoroti teks yang berwarna biru, hijau, atau merah dalam gambar, sehingga teks tersebut lebih terlihat jelas di citra grayscale.

Langkah 5: Menampilkan Citra dalam Grid

- Gambar-gambar yang telah diproses ditampilkan dalam grid 2x2 menggunakan plt.subplots().
- Di dalam grid ini, ada empat gambar:
  1. Gambar asli (dalam format RGB).
  2. Hasil highlight untuk warna biru, di mana semua area biru ditampilkan putih di atas latar belakang gelap.
  3. Hasil highlight untuk warna merah.
  4. Hasil highlight untuk warna hijau.
- Semua gambar ini ditampilkan tanpa sumbu menggunakan axis('off') untuk memudahkan visualisasi.

Langkah 6: Histogram dan Gambar Inset

- Untuk menganalisis distribusi piksel, empat histogram ditampilkan, masing-masing untuk:
  1. Gambar grayscale asli.
  2. Gambar hasil highlight untuk biru.
  3. Gambar hasil highlight untuk merah.
  4. Gambar hasil highlight untuk hijau.
- Histogram menunjukkan distribusi nilai intensitas piksel dalam rentang 0-255. Untuk gambar asli grayscale, distribusinya lebih tersebar, sementara gambar highlight cenderung memiliki nilai piksel yang sangat tinggi (putih) atau sangat rendah (hitam), tergantung pada area yang disorot.
- Di setiap subplot histogram, terdapat gambar inset yang lebih besar, hampir memenuhi keseluruhan subplot. Gambar inset ini menampilkan gambar yang relevan dengan histogram yang sedang ditampilkan, sehingga bisa langsung melihat area mana yang berwarna biru, merah, atau hijau.

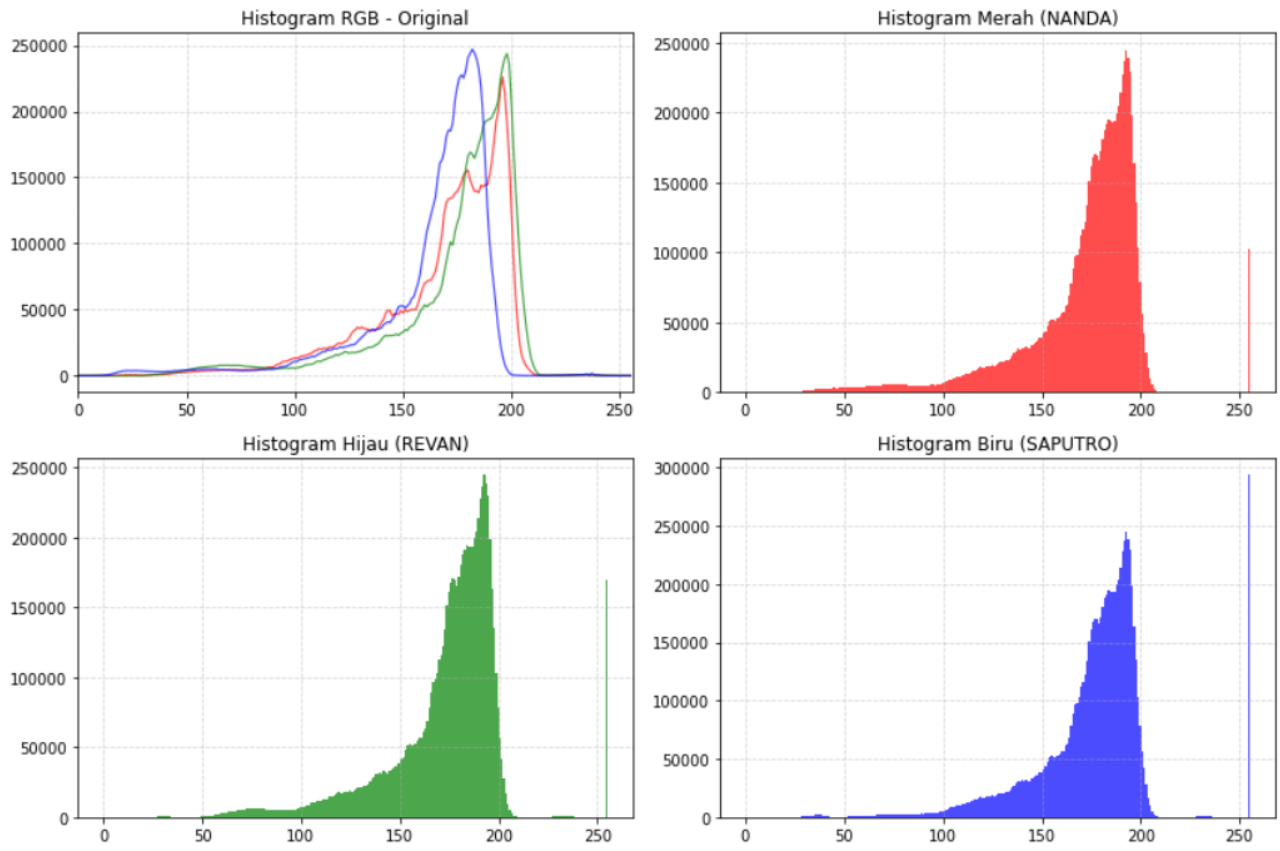
## Menampilkan Histogram

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

# 1. Baca citra
img_bgr = cv2.imread('tulisan7.jpg')
if img_bgr is None:
    raise FileNotFoundError("File tulisan7.jpg tidak ditemukan!")

img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
gray = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2GRAY)
hsv = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2HSV)

# 2. Rentang HSV untuk mask
mask_red   = cv2.inRange(hsv, (0, 50, 50), (10, 255, 255)) + cv2.inRange(hsv, (160, 50, 50), (180, 255, 255))
mask_green = cv2.inRange(hsv, (30, 40, 40), (100, 255, 255))
mask_blue  = cv2.inRange(hsv, (80, 50, 50), (140, 255, 255))

# 3. Highlight teks dengan warna murni
def highlight(gray_img, mask, color='red'):
    out = cv2.cvtColor(gray_img, cv2.COLOR_GRAY2BGR)
    if color == 'red':
        out[mask > 0] = (0, 0, 255)
    elif color == 'green':
        out[mask > 0] = (0, 255, 0)
    elif color == 'blue':
        out[mask > 0] = (255, 0, 0)
    return out
```

```
h_red   = highlight(gray, mask_red, 'red')
h_green = highlight(gray, mask_green, 'green')
h_blue  = highlight(gray, mask_blue, 'blue')

# 4. Ambil channel warna yang relevan saja
red_channel   = h_red[:, :, 2]     # Red channel
green_channel = h_green[:, :, 1]   # Green channel
blue_channel  = h_blue[:, :, 0]    # Blue channel

# 5. Fungsi plot histogram RGB untuk citra asli
def plot_rgb_histogram(image, title, ax):
    colors = ('r', 'g', 'b')
    for i, color in enumerate(colors):
        hist = cv2.calcHist([image], [i], None, [256], [0, 256])
        ax.plot(hist, color=color, alpha=0.7, linewidth=1.2)
    ax.set_title(title)
    ax.set_xlim([0, 256])
    ax.grid(True, linestyle='--', alpha=0.5)

# 6. Plot 4 histogram saja
fig, axs = plt.subplots(2, 2, figsize=(12, 8))

# Histogram RGB - Original
plot_rgb_histogram(cv2.cvtColor(img_rgb, cv2.COLOR_RGB2BGR), "Histogram RGB - Original", axs[0, 0])
```

```
# Histogram Merah (NANDA)
axs[0, 1].hist(red_channel.ravel(), bins=256, range=(0,255), color='red', alpha=0.7)
axs[0, 1].set_title("Histogram Merah (NANDA)")
axs[0, 1].grid(True, linestyle='--', alpha=0.5)

# Histogram Hijau (REVAN)
axs[1, 0].hist(green_channel.ravel(), bins=256, range=(0,255), color='green', alpha=0.7)
axs[1, 0].set_title("Histogram Hijau (REVAN)")
axs[1, 0].grid(True, linestyle='--', alpha=0.5)

# Histogram Biru (SAPUTRO)
axs[1, 1].hist(blue_channel.ravel(), bins=256, range=(0,255), color='blue', alpha=0.7)
axs[1, 1].set_title("Histogram Biru (SAPUTRO)")
axs[1, 1].grid(True, linestyle='--', alpha=0.5)

plt.tight_layout()
plt.show()

#202331016_Nanda Revan Saputro
```

**Output:**




**Penjelasan:**

1. Membaca Citra

- cv2.imread('tulisan7.jpg'): Membaca file gambar tulisan7.jpg dalam format BGR.
- cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB): Mengubah citra dari format BGR ke RGB, karena OpenCV secara default membaca citra dalam format BGR.

- cv2.cvtColor(img_rgb, cv2.COLOR_RGB2GRAY): Mengonversi citra RGB menjadi citra grayscale (grayscale hanya menggunakan intensitas terang-gelap tanpa informasi warna).
- cv2.cvtColor(img_rgb, cv2.COLOR_RGB2HSV): Mengubah citra RGB menjadi format HSV (Hue, Saturation, Value) yang lebih cocok untuk pemrosesan warna.

2. Rentang HSV untuk Mask

- Rentang warna dalam ruang HSV digunakan untuk membuat mask berdasarkan warna tertentu (merah, hijau, biru):
- cv2.inRange(hsv, (0, 50, 50), (10, 255, 255)): Membuat mask untuk warna merah pada rentang Hue 0-10.
- cv2.inRange(hsv, (160, 50, 50), (180, 255, 255)): Mask untuk warna merah pada rentang Hue 160-180.
- cv2.inRange(hsv, (30, 40, 40), (100, 255, 255)): Mask untuk warna hijau.
- cv2.inRange(hsv, (80, 50, 50), (140, 255, 255)): Mask untuk warna biru.
- Mask yang dihasilkan akan memiliki nilai 255 (putih) pada area yang sesuai dengan rentang warna dan 0 (hitam) untuk area lainnya.

3. Highlight Teks dengan Warna Murni

- Fungsi highlight() digunakan untuk menyoroti area teks yang terdeteksi dengan warna tertentu (merah, hijau, biru):
- cv2.cvtColor(gray_img, cv2.COLOR_GRAY2BGR): Mengonversi citra grayscale kembali ke format BGR.
- out[mask > 0] = (0, 0, 255): Jika mask merah (warna merah) terdeteksi (nilai > 0), piksel diubah menjadi merah (0, 0, 255).
- Fungsi ini mengulangi proses yang sama untuk warna hijau dan biru dengan menggunakan warna yang sesuai (hijau: (0, 255, 0), biru: (255, 0, 0)).

4. Ambil Channel Warna yang Relevan

- h_red[:, :, 2]: Mengambil channel merah dari citra yang sudah disorot merah.
- h_green[:, :, 1]: Mengambil channel hijau dari citra yang sudah disorot hijau.
- h_blue[:, :, 0]: Mengambil channel biru dari citra yang sudah disorot biru.
- Setiap channel ini digunakan untuk menghitung histogram warna yang relevan.

5. Fungsi Plot Histogram RGB untuk Citra Asli

- Fungsi plot_rgb_histogram() menghitung dan menampilkan histogram untuk setiap channel warna (merah, hijau, biru) pada citra asli:

- cv2.calcHist([image], [i], None, [256], [0, 256]): Menghitung histogram untuk setiap channel warna.
- i menentukan channel yang dihitung (0 untuk merah, 1 untuk hijau, 2 untuk biru).
- Histogram ini menggambarkan distribusi nilai intensitas piksel untuk setiap warna.
- ax.plot(): Menampilkan histogram pada subplot yang sesuai.

6. Plot 4 Histogram

- Histogram yang ditampilkan pada 4 subplot dalam grid 2x2:
- Histogram RGB - Original: Menampilkan histogram untuk citra asli (RGB).
- Histogram Merah (NANDA): Menampilkan histogram untuk channel merah setelah teks yang berwarna merah disorot.
- Histogram Hijau (REVAN): Menampilkan histogram untuk channel hijau setelah teks yang berwarna hijau disorot.
- Histogram Biru (SAPUTRO): Menampilkan histogram untuk channel biru setelah teks yang berwarna biru disorot.
- Pengaturan grid menggunakan plt.tight_layout() memastikan bahwa plot tidak saling tumpang tindih dan tampil dengan rapi.

Hasil Visualisasi

- Histogram RGB - Original: Menampilkan distribusi intensitas untuk semua warna di citra asli.
- Histogram Merah (NANDA): Menampilkan distribusi intensitas untuk warna merah yang terdeteksi.
- Histogram Hijau (REVAN): Menampilkan distribusi intensitas untuk warna hijau yang terdeteksi.
- Histogram Biru (SAPUTRO): Menampilkan distribusi intensitas untuk warna biru yang terdeteksi.

**Soal Nomor 2**

In [2]:
```
import cv2
import matplotlib.pyplot as plt
import numpy as np

#202331016_Nanda Revan Saputro
```

**Baris ini mengimpor pustaka yang dibutuhkan**

- cv2 untuk pengolahan citra menggunakan OpenCV.
- matplotlib.pyplot sebagai alat untuk menampilkan gambar dan grafik.
- numpy digunakan untuk membuat array numerik seperti ambang batas warna.

In [3]:
```
# Load gambar dan resize agar lebih kecil
image = cv2.imread('tulisan7.jpg')
resized_image = cv2.resize(image, (600, 400))

# Konversi ke RGB untuk visualisasi
rgb_image = cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB)

plt.imshow(rgb_image)
plt.title("Gambar Asli")
plt.axis('off')
plt.show()


#202331016_Nanda Revan Saputro
```

**Output:**

**Penjelasan:**

- image = cv2.imread('tulisan7.jpg')

Digunakan untuk membaca gambar dari file bernama tulisan7.jpg. Gambar dibaca dalam format warna BGR (Blue-Green-Red), yaitu format warna standar pada OpenCV.

- resized_image = cv2.resize(image, (600, 400))

Gambar hasil pembacaan diubah ukurannya menjadi 600 piksel lebar dan 400 piksel tinggi. Tujuannya agar lebih ringan diproses dan ditampilkan (apalagi kalau gambar aslinya besar).

- rgb_image = cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB)

Karena OpenCV membaca gambar dalam format BGR, sedangkan matplotlib.pyplot.imshow() menggunakan format RGB, maka gambar perlu dikonversi ke format warna RGB terlebih dahulu agar warna tampil dengan benar.

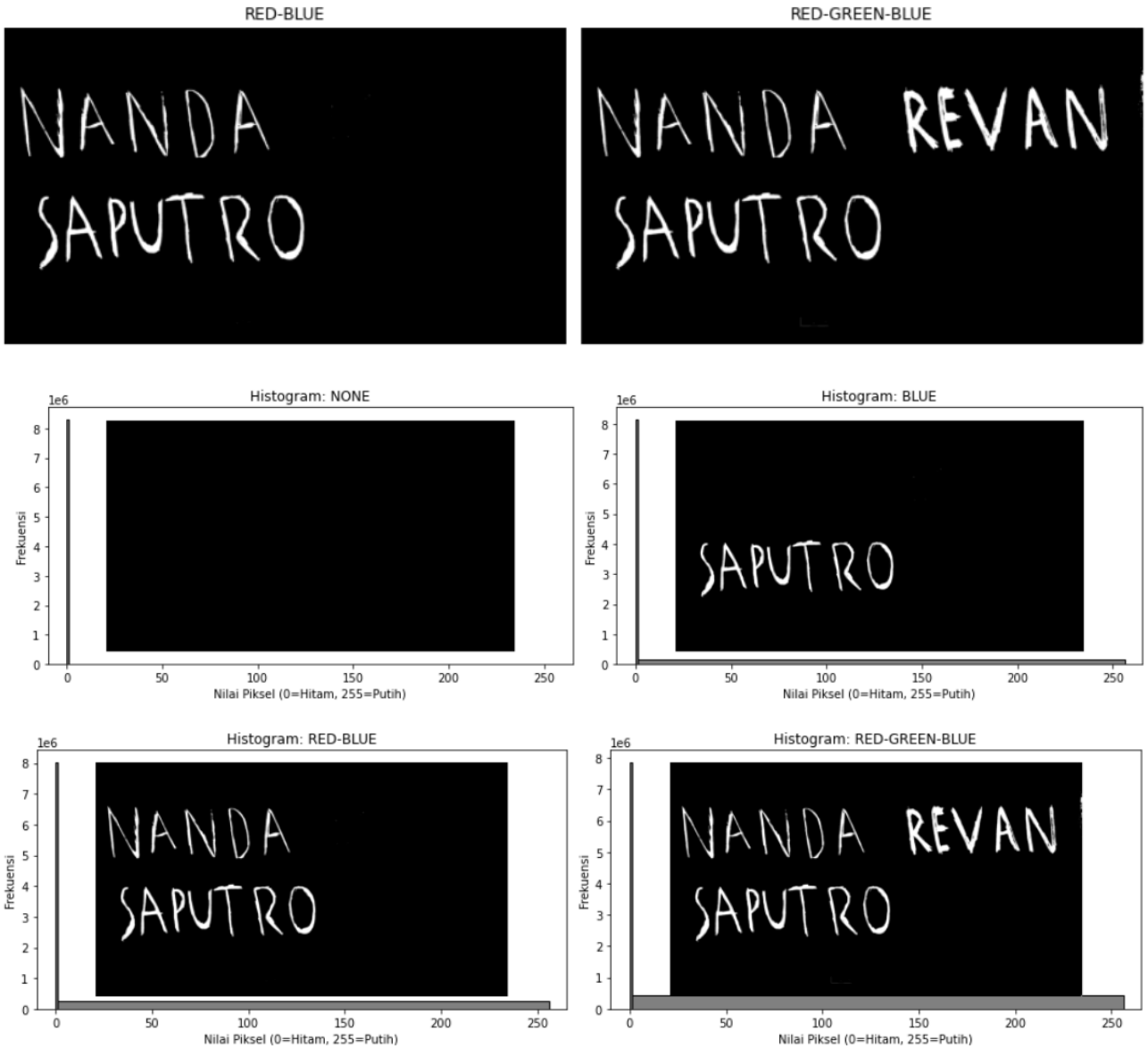
- Visualisasi dengan Matplotlib:
  - plt.imshow(rgb_image): Menampilkan gambar berwarna RGB.
  - plt.title("Gambar Asli"): Memberi judul tampilan gambar.
  - plt.axis('off'): Menyembunyikan garis sumbu (x dan y) supaya tampilan bersih.
  - plt.show(): Memunculkan tampilan gambar ke layar.

## Kategori Ambang Batas 202331016_Nanda Revan Saputro

```
# 1. Baca & konversi ke RGB + HSV
img_bgr = cv2.imread('tulisan7.jpg')
if img_bgr is None:
    raise FileNotFoundError("tulisan7.jpg tidak ditemukan!")
img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
hsv     = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2HSV)

# 2. Definisi rentang HSV untuk tiap warna
ranges = {
    'blue':  (np.array([100, 50, 50]),  np.array([140, 255, 255])),
    'green': (np.array([ 30, 40, 40]),  np.array([100, 255, 255])),
    'red1':  (np.array([  0, 50, 50]),  np.array([ 10, 255, 255])),
    'red2':  (np.array([160, 50, 50]),  np.array([180, 255, 255])),
}

# 3. Buat mask untuk masing-masing warna
mask_blue  = cv2.inRange(hsv, *ranges['blue'])
mask_green = cv2.inRange(hsv, *ranges['green'])
mask_red   = cv2.inRange(hsv, *ranges['red1']) | cv2.inRange(hsv, *ranges['red2'])

# 4. Siapkan kombinasi mask sesuai kategori
mask_none       = np.zeros_like(mask_blue)                          # NONE
mask_only_blue  = mask_blue                                         # BLUE
mask_red_blue   = (mask_red > 0) | (mask_blue > 0)                  # RED-BLUE
mask_all_colors = (mask_red > 0) | (mask_green > 0) | (mask_blue > 0)  # RED-GREEN-BLUE

masks = [
    ('NONE',        mask_none),
    ('BLUE',        mask_only_blue),
    ('RED-BLUE',    mask_red_blue.astype(np.uint8)*255),
    ('RED-GREEN-BLUE', mask_all_colors.astype(np.uint8)*255),
]
```

```python
# 5. Fungsi untuk membuat output binary: putih untuk mask, hitam untuk lainnya
def apply_binary(mask):
    # mask sudah 0/255; langsung tampilkan sebagai grayscale
    return mask

# 6. Plot 2×2
fig, axes = plt.subplots(2, 2, figsize=(12, 8))
axes = axes.ravel()

for ax, (title, mask) in zip(axes, masks):
    out = apply_binary(mask)
    ax.imshow(out, cmap='gray', vmin=0, vmax=255)
    ax.set_title(title)
    ax.axis('off')

plt.tight_layout()
plt.show()

# 7. Plot histogram dari mask dengan inset besar
fig, axs = plt.subplots(2, 2, figsize=(14, 8))
positions = [(0, 0), (0, 1), (1, 0), (1, 1)]

for (title, mask), (r, c) in zip(masks, positions):
    ax = axs[r][c]

    # Histogram dari nilai 0 dan 255
    ax.hist(mask.ravel(), bins=[0, 1, 256], color='gray', edgecolor='black')
    ax.set_title(f'Histogram: {title}')
    ax.set_xlabel('Nilai Piksel (0=Hitam, 255=Putih)')
    ax.set_ylabel('Frekuensi')
    ax.set_xlim([-10, 265])

    # Inset besar untuk tampilan mask-nya
    axins = ax.inset_axes([0.05, 0.05, 0.9, 0.9])
    axins.imshow(mask, cmap='gray', vmin=0, vmax=255)
    axins.axis('off')
plt.tight_layout()
plt.show()
#202331016 Nanda Revan Saputro
```

**Output:**




**Penjelasan:**

1. Baca & Konversi Gambar

- Gambar dibaca menggunakan cv2.imread() dan diasumsikan bernama tulisan7.jpg.
- Jika file tidak ditemukan, program langsung dihentikan dengan pesan error (FileNotFoundError).
- Gambar dikonversi dari format BGR (standar OpenCV) ke RGB, karena RGB lebih sesuai untuk visualisasi.
- Selanjutnya, gambar RGB dikonversi ke HSV karena HSV lebih cocok untuk segmentasi warna. Hue mewakili warna dasar, Saturation menunjukkan kejenuhan warna, dan Value adalah kecerahan.

2. Definisi Rentang HSV

- Rentang warna didefinisikan dalam bentuk dictionary ranges untuk:

1. Biru (blue): dengan Hue antara 100–140 derajat.
2. Hijau (green): dengan Hue antara 30–100 derajat.
3. Merah (red): dibagi dua, karena warna merah berada di dua sisi Hue, yaitu:
4. Red1: Hue 0–10
5. Red2: Hue 160–180

- Semua rentang disimpan sebagai numpy array untuk kemudahan pemrosesan dalam OpenCV.

3. Pembuatan Masker Warna

- Masker dibuat untuk tiap warna dengan cv2.inRange(), yang mengembalikan citra biner:
- Nilai 255 (putih) untuk piksel yang berada dalam rentang warna.
- Nilai 0 (hitam) untuk piksel lainnya.
- Masker merah dibuat dengan menggabungkan dua rentang (red1 dan red2) menggunakan operasi OR (|).
- Hasilnya adalah tiga masker biner: merah, hijau, dan biru.

4. Kombinasi Masker

- Dibuat empat kombinasi kategori:
  1. NONE: Tidak ada warna terdeteksi. Diwakili oleh array kosong (semua nilai 0).
  2. BLUE: Hanya piksel biru yang muncul.
  3. RED-BLUE: Kombinasi dari piksel merah dan biru.
  4. RED-GREEN-BLUE: Gabungan semua warna yang terdeteksi (merah, hijau, biru).
- Kombinasi dibuat dengan operasi logika OR antar-masker.
- Masker dikonversi ke uint8 dan dikali 255 agar tetap dalam format citra biner (0 dan 255).

5. Fungsi Output Biner

- Fungsi apply_binary() hanya mengembalikan masker karena nilainya sudah dalam bentuk biner.
- Tidak dilakukan perubahan apapun, cukup untuk visualisasi grayscale hitam-putih.

6. Plot Masker (2×2)

- Dibuat subplot 2 baris 2 kolom menggunakan matplotlib.
- Setiap kombinasi masker divisualisasikan sebagai citra grayscale:
- Putih (255) menunjukkan area yang sesuai dengan warna.
- Hitam (0) menunjukkan area lain.

- Judul setiap subplot mencerminkan kategori maskernya (misal: RED-BLUE).
- Sumbu tidak ditampilkan agar fokus ke gambar.

7. Histogram Piksel dengan Inset

- Dibuat histogram untuk masing-masing masker.
- Histogram memperlihatkan jumlah piksel yang bernilai:
- 0 = hitam (tidak termasuk dalam mask)
- 255 = putih (termasuk dalam mask)
- Digunakan ax.hist() untuk menghitung frekuensi masing-masing nilai piksel.
- Di dalam setiap subplot histogram, ditambahkan inset gambar untuk memperlihatkan masker biner secara visual.
- inset_axes menampilkan kembali maskernya dalam ukuran kecil di dalam histogram.
- Histogram ini memberikan informasi tentang berapa banyak area dalam gambar yang sesuai dengan kategori warna tertentu.

Hasil Visualisasi

1. Gambar Masker (2×2 Grid)

Pada grid ini, ada empat gambar yang menunjukkan bagaimana warna tertentu diekstraksi dari gambar asli menggunakan masker biner:

- NONE:
  Gambar ini menunjukkan area yang tidak ada warna yang terdeteksi berdasarkan rentang HSV yang ditentukan. Semua area adalah hitam, yang berarti tidak ada warna yang cocok dengan kriteria yang ditentukan.

- BLUE:
  Pada gambar ini, hanya area yang sesuai dengan warna biru dalam rentang HSV yang akan berwarna putih (255). Semua area lain akan berwarna hitam (0).
  Hanya bagian dari gambar yang memiliki nilai HSV dalam rentang biru yang akan tampak putih, sedangkan bagian lain tetap hitam.

- RED-BLUE:
  Gambar ini merupakan kombinasi masker merah dan biru. Area yang memiliki nilai dalam rentang merah atau biru akan berwarna putih.
  Ini menunjukkan gabungan piksel yang terdeteksi sebagai merah dan biru dalam gambar asli.

- RED-GREEN-BLUE:
  Gambar ini menunjukkan semua piksel yang sesuai dengan warna merah, hijau, atau biru berdasarkan rentang HSV yang telah didefinisikan sebelumnya.

Area yang sesuai dengan salah satu dari tiga warna tersebut akan berwarna putih, sementara yang lain tetap hitam.

2. Histogram dari Masker

Setiap masker juga memiliki histogram yang menunjukkan distribusi nilai piksel (0 untuk hitam dan 255 untuk putih):

Histogram menunjukkan frekuensi kemunculan nilai piksel tertentu dalam gambar setelah penerapan masker.

- Piksel dengan nilai 0:
  Mewakili area yang tidak termasuk dalam mask (hitam), artinya area tersebut tidak memenuhi kriteria warna yang ditentukan dalam rentang HSV.

- Piksel dengan nilai 255:
  Mewakili area yang termasuk dalam mask (putih), artinya area tersebut memenuhi kriteria warna yang ditentukan dalam rentang HSV.

- Histograms untuk masing-masing warna (Red, Green, Blue):
  Untuk setiap warna, histogram menunjukkan seberapa banyak piksel dalam gambar yang terdeteksi sesuai dengan warna tersebut (merah, hijau, biru). Misalnya:

- Histogram Biru (Blue): Menampilkan jumlah piksel yang berada dalam rentang warna biru. Histogram akan menunjukkan puncak besar di nilai 255 jika gambar banyak mengandung warna biru.

- Histogram Merah (Red): Menampilkan jumlah piksel yang terdeteksi sebagai warna merah. Ada dua puncak karena warna merah dibagi menjadi dua rentang (Red1 dan Red2).

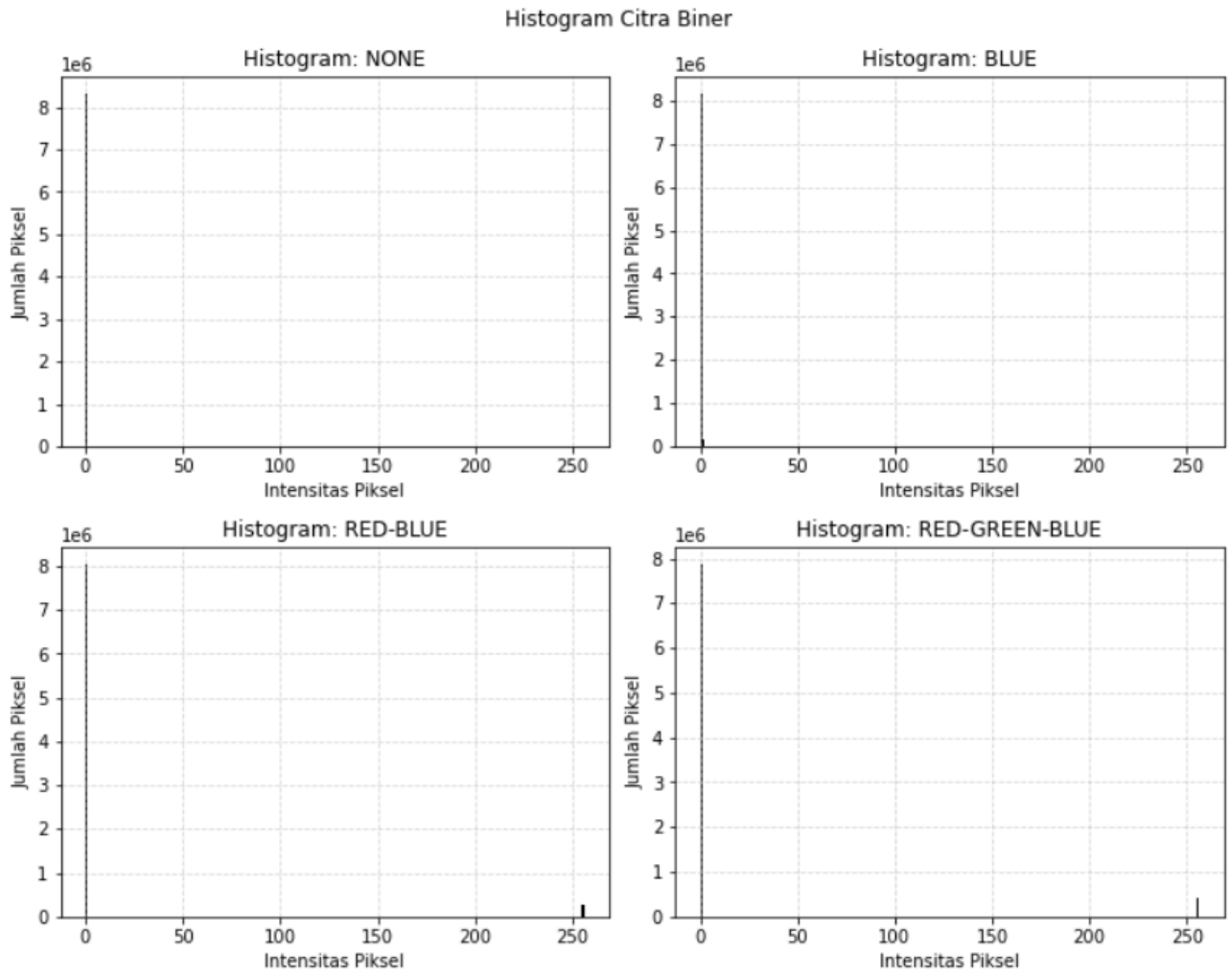
- Histogram Hijau (Green): Menampilkan jumlah piksel hijau yang terdeteksi dalam gambar.

- Inset Gambar dalam Histogram:
  Setiap histogram memiliki inset yang menunjukkan gambar mask yang sesuai dengan warna tertentu. Ini memberikan gambaran visual dari area yang terdeteksi berdasarkan masker, memperlihatkan bentuk dan lokasi area yang relevan dengan histogram tersebut.

# Histogram

```
# Plot 2×2 histogram
fig2, axes2 = plt.subplots(2, 2, figsize=(10, 8))
axes2 = axes2.ravel()
for ax, (title, mask) in zip(axes2, masks):
    ax.hist(mask.ravel(), bins=256, range=(0,256), color='black')
    ax.set_title(f"Histogram: {title}")
    ax.set_xlabel('Intensitas Piksel')
    ax.set_ylabel('Jumlah Piksel')
    ax.grid(True, linestyle='--', alpha=0.5)

plt.suptitle("Histogram Citra Biner")
plt.tight_layout()
plt.show()

#202331016_Nanda Revan Saputro
```

**Output:**



**Penjelasan:**

1. Membuat Subplot 2×2

- fig2, axes2 = plt.subplots(2, 2, figsize=(10, 8)): Membuat grid subplot dengan ukuran 2x2. Dengan figsize=(10, 8), ukuran gambar keseluruhan akan disesuaikan dengan lebih lebar dan tinggi.
- axes2 = axes2.ravel(): Mengubah array 2D dari axes menjadi array 1D agar mudah digunakan dalam loop.

2. Plot Histogram untuk Masing-masing Masker

- Loop for ax, (title, mask) in zip(axes2, masks)::
- Loop ini digunakan untuk mengiterasi setiap masker dalam masks, yang berisi kombinasi masker (misalnya, BLUE, RED-BLUE, dll.).
- title berisi nama kategori masker, sementara mask adalah citra biner yang berisi nilai 0 dan 255.

3. Plot Histogram

- ax.hist(mask.ravel(), bins=256, range=(0, 256), color='black'):
- mask.ravel(): Mengubah masker 2D menjadi 1D agar dapat dihitung histogramnya.
- bins=256: Menggunakan 256 bin, yang artinya intensitas piksel dihitung dalam rentang 0 hingga 255 (nilai standar untuk intensitas piksel dalam citra grayscale).
- range=(0, 256): Menetapkan rentang histogram dari 0 hingga 255.
- color='black': Menampilkan histogram dengan warna hitam.

4. Pengaturan Label dan Grid

- ax.set_title(f"Histogram: {title}"): Memberikan judul subplot dengan nama kategori masker.
- ax.set_xlabel('Intensitas Piksel') dan ax.set_ylabel('Jumlah Piksel'): Menambahkan label pada sumbu x dan y untuk menjelaskan histogram:
- Sumbu x menunjukkan intensitas piksel (0 hingga 255).
- Sumbu y menunjukkan jumlah piksel dengan intensitas tersebut.
- ax.grid(True, linestyle='--', alpha=0.5): Menambahkan grid dengan garis putus-putus dan transparansi 50% untuk membuat plot lebih mudah dibaca.

5. Pengaturan Judul dan Layout

- plt.suptitle("Histogram Citra Biner"): Menambahkan judul utama pada seluruh gambar subplot.

- plt.tight_layout(): Mengatur layout agar elemen-elemen plot tidak tumpang tindih dan tata letaknya rapi.
- plt.show(): Menampilkan plot pada layar.

Hasil Output:

Haislnya yaitu empat subplot yang menunjukkan histogram dari setiap masker yang didefinisikan (misalnya, NONE, BLUE, RED-BLUE, RED-GREEN-BLUE), masing-masing dengan intensitas piksel dan jumlah piksel yang sesuai.

Setiap histogram akan menunjukkan distribusi nilai piksel 0 dan 255, dengan sumbu x mewakili intensitas piksel dan sumbu y menunjukkan frekuensi.

**Soal Nomor 3**

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

#202331016_Nanda Revan Saputro
```

**Baris ini mengimpor pustaka yang dibutuhkan**

- cv2 untuk pengolahan citra menggunakan OpenCV.
- matplotlib.pyplot sebagai alat untuk menampilkan gambar dan grafik.
- numpy digunakan untuk membuat array numerik seperti ambang batas warna.

```
# Load Gambar
# ================================
img = cv2.imread('revan2.jpg')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Cerahkan
bright = cv2.convertScaleAbs(gray, alpha=1, beta=50)

# Kontras (langsung dari gray)
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
contrast_only = clahe.apply(gray)

# Cerah + Kontras (Final)
contrast_final = clahe.apply(bright)


#202331016_Nanda Revan Saputro
```

**Penjelasan**

1. Membaca dan Mengkonversi Gambar

img = cv2.imread('revan2.jpg'): Membaca file gambar 'revan2.jpg' menggunakan OpenCV dan menyimpannya dalam variabel img.

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB): Mengkonversi gambar dari format BGR (format standar OpenCV) ke RGB, sehingga gambar bisa ditampilkan dengan benar pada library yang menggunakan format RGB (misalnya Matplotlib).

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY): Mengkonversi gambar asli (img) menjadi citra grayscale (hitam-putih) yang digunakan untuk analisis lebih lanjut.

2. Pencahayaan (Brightening)

bright = cv2.convertScaleAbs(gray, alpha=1, beta=50):

Fungsi cv2.convertScaleAbs() digunakan untuk mengubah intensitas piksel. Dalam hal ini:

alpha=1: Tidak ada perubahan pada faktor skala, sehingga intensitas asli tetap sama.

beta=50: Nilai ini digunakan untuk menambahkan pencahayaan pada citra dengan menambahkan 50 ke setiap nilai piksel, menghasilkan gambar yang lebih terang.

3. Meningkatkan Kontras (CLAHE)

clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8)):

CLAHE (Contrast Limited Adaptive Histogram Equalization) adalah metode untuk meningkatkan kontras lokal gambar dengan memperbaiki distribusi intensitas piksel di daerah lokal citra.

clipLimit=2.0: Menentukan batas maksimum untuk kontras lokal. Nilai ini mengontrol seberapa banyak kontras dapat ditingkatkan.

tileGridSize=(8,8): Mengatur ukuran grid lokal untuk pembagian citra menjadi blok 8x8 piksel. Setiap blok akan diproses untuk meningkatkan kontrasnya.

contrast_only = clahe.apply(gray): Menerapkan teknik CLAHE pada citra grayscale untuk meningkatkan kontrasnya.

4. Menggabungkan Pencahayaan dan Kontras

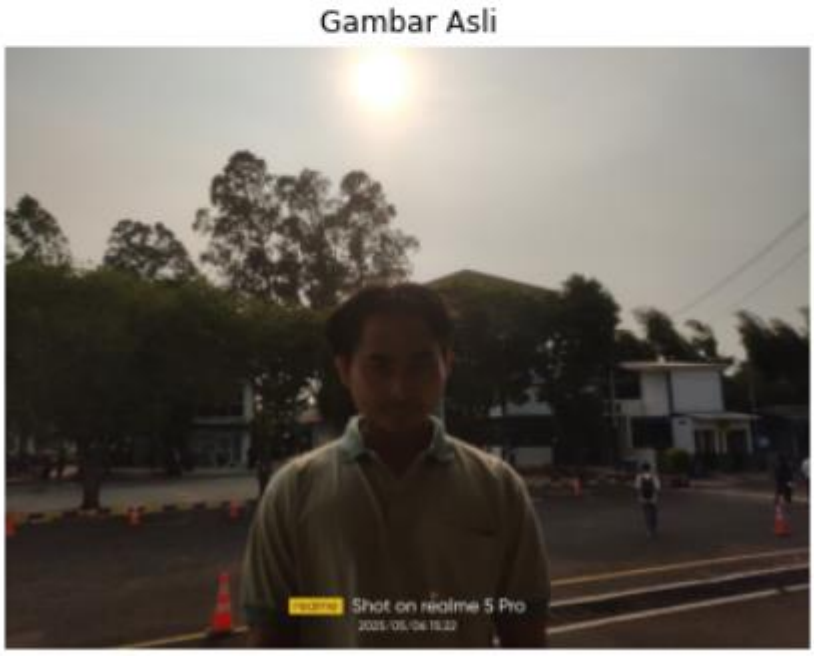
contrast_final = clahe.apply(bright):

Menggunakan hasil gambar yang telah diterangi sebelumnya (bright), kemudian menerapkan teknik CLAHE untuk meningkatkan kontras pada gambar yang sudah cerah.

Hasilnya adalah gambar dengan tingkat pencahayaan dan kontras yang lebih baik, memberikan detail yang lebih jelas pada gambar.

# 1. Gambar Asli

```
# 1. Gambar Asli
# ================================
plt.figure(figsize=(6,6))
plt.imshow(img_rgb)
plt.title('Gambar Asli')
plt.axis('off')
plt.show()


#202331016_Nanda Revan Saputro
```

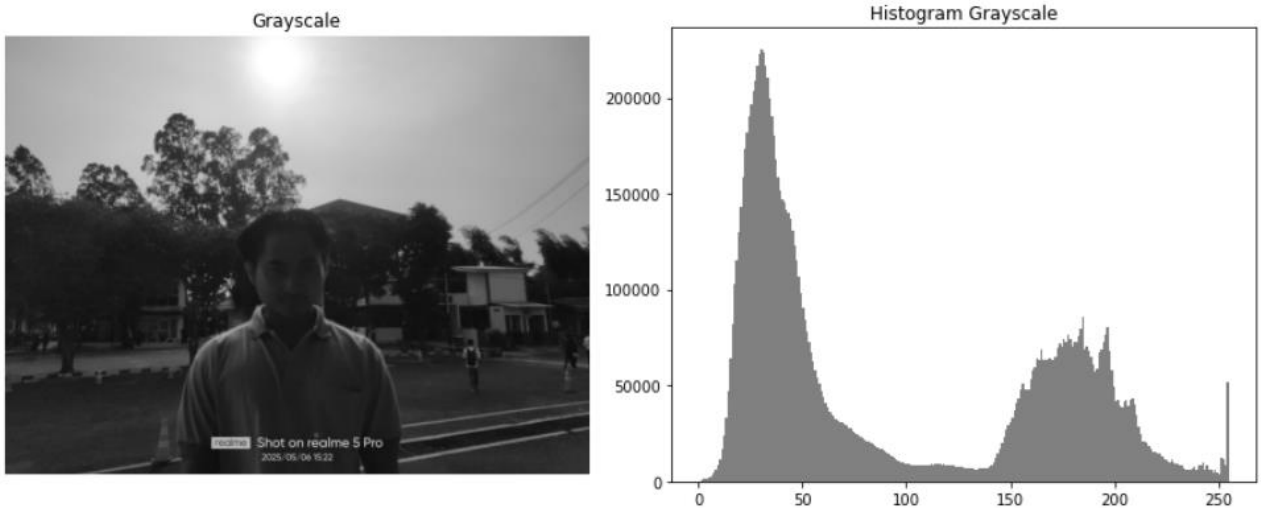



**Menampilkan Gambar Asli**

- plt.figure(figsize=(6,6)):
- Fungsi ini digunakan untuk membuat sebuah figure baru untuk plot, dengan ukuran 6x6 inci. Hal ini menentukan ukuran tampilan gambar yang akan ditampilkan pada plot.
- plt.imshow(img_rgb):
- Fungsi imshow() digunakan untuk menampilkan gambar pada plot. Di sini, gambar yang telah dikonversi ke format RGB (img_rgb) ditampilkan. Ini memastikan bahwa gambar tampil dengan warna yang benar, karena OpenCV menggunakan format BGR secara default.
- plt.title('Gambar Asli'):
- Memberikan judul pada plot dengan teks "Gambar Asli", yang membantu dalam memberikan konteks untuk gambar yang sedang ditampilkan.

## 2. Grayscale

```
# 2. Grayscale + Histogram
# ================================
fig, ax = plt.subplots(1, 2, figsize=(12, 5))
ax[0].imshow(gray, cmap='gray')
ax[0].set_title('Grayscale')
ax[0].axis('off')
ax[1].hist(gray.ravel(), bins=256, color='gray')
ax[1].set_title('Histogram Grayscale')
plt.tight_layout()
plt.show()


#202331016_Nanda Revan Saputro
```

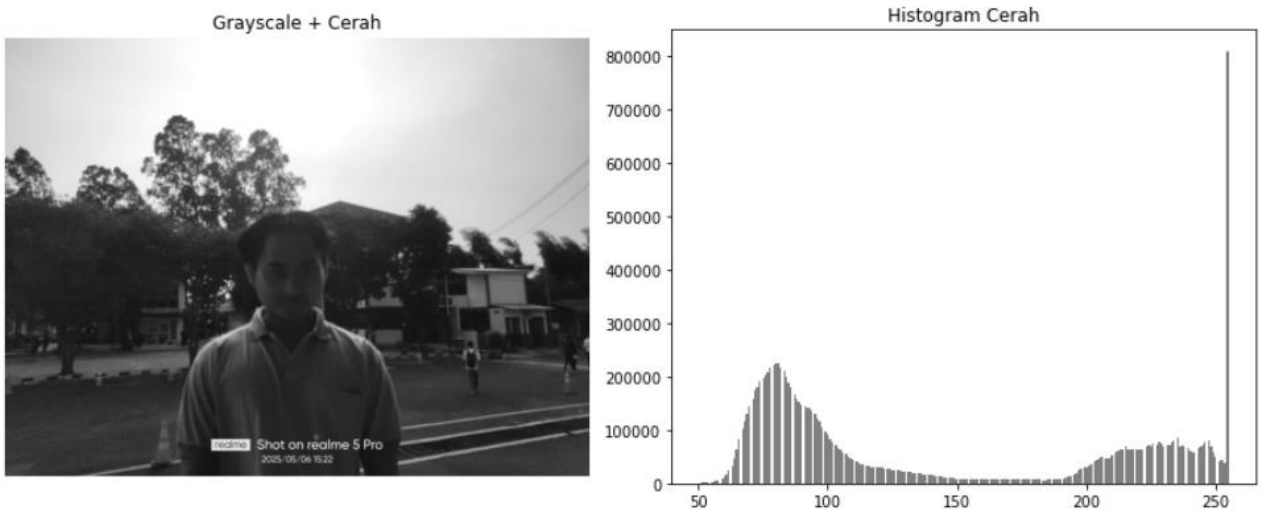



**Menampilkan Grayscale**

- plt.subplot(1, 2, 1) dan plt.subplot(1, 2, 2): Digunakan untuk membagi area plot menjadi 1 baris dan 2 kolom, lalu menampilkan gambar grayscale di kolom pertama dan histogram di kolom kedua.
- plt.imshow(gray, cmap='gray'): Menampilkan gambar grayscale.
- plt.hist(gray.ravel(), bins=256, color='gray'): Membuat histogram dari gambar grayscale.
- plt.axis('off'): Menghilangkan sumbu pada gambar grayscale agar tampil lebih bersih.

## 3. Gray yang dipercerah

```
# 3. Grayscale + Cerah + Histogram
# ================================
fig, ax = plt.subplots(1, 2, figsize=(12, 5))
ax[0].imshow(bright, cmap='gray')
ax[0].set_title('Grayscale + Cerah')
ax[0].axis('off')
ax[1].hist(bright.ravel(), bins=256, color='gray')
ax[1].set_title('Histogram Cerah')
plt.tight_layout()
plt.show()


#202331016_Nanda Revan Saputro
```




**Penjelasan:**

Membuat Plot 2x1:

- fig, ax = plt.subplots(1, 2, figsize=(12, 5)): Membuat figure dengan dua kolom (1 baris, 2 kolom) untuk menampilkan dua gambar (grayscale dan histogram).

Menampilkan Gambar Grayscale:

- ax[0].imshow(gray, cmap='gray'): Menampilkan gambar grayscale di subplot pertama (kolom pertama).
- ax[0].set_title('Grayscale'): Menambahkan judul pada subplot pertama.
- ax[0].axis('off'): Menghilangkan axis atau sumbu agar gambar tampil lebih bersih.

Membuat Histogram dari Gambar Grayscale:

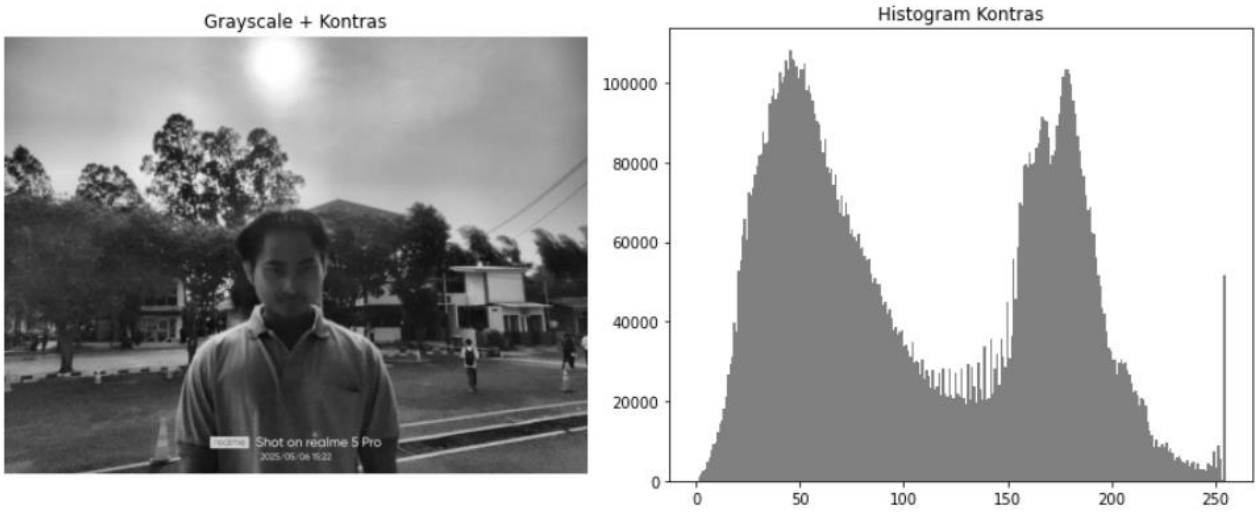
- ax[1].hist(gray.ravel(), bins=256, color='gray'): Membuat histogram berdasarkan piksel dari gambar grayscale.
- ax[1].set_title('Histogram Grayscale'): Menambahkan judul pada subplot kedua.

Tata Letak Plot:

- plt.tight_layout(): Mengatur layout agar gambar dan histogram tidak tumpang tindih.
- plt.show(): Menampilkan hasil visualisasi.

## 4. Gray yang diperkontras

```
# 4. Grayscale + Kontras + Histogram
# ================================
fig, ax = plt.subplots(1, 2, figsize=(12, 5))
ax[0].imshow(contrast_only, cmap='gray')
ax[0].set_title('Grayscale + Kontras')
ax[0].axis('off')
ax[1].hist(contrast_only.ravel(), bins=256, color='gray')
ax[1].set_title('Histogram Kontras')
plt.tight_layout()
plt.show()


#202331016_Nanda Revan Saputro
```




**Penjelasan:**

Membuat Plot 2x1:

- fig, ax = plt.subplots(1, 2, figsize=(12, 5)): Membuat figure dengan dua kolom untuk menampilkan gambar kontras dan histogram secara berdampingan.

Menampilkan Gambar dengan Kontras:

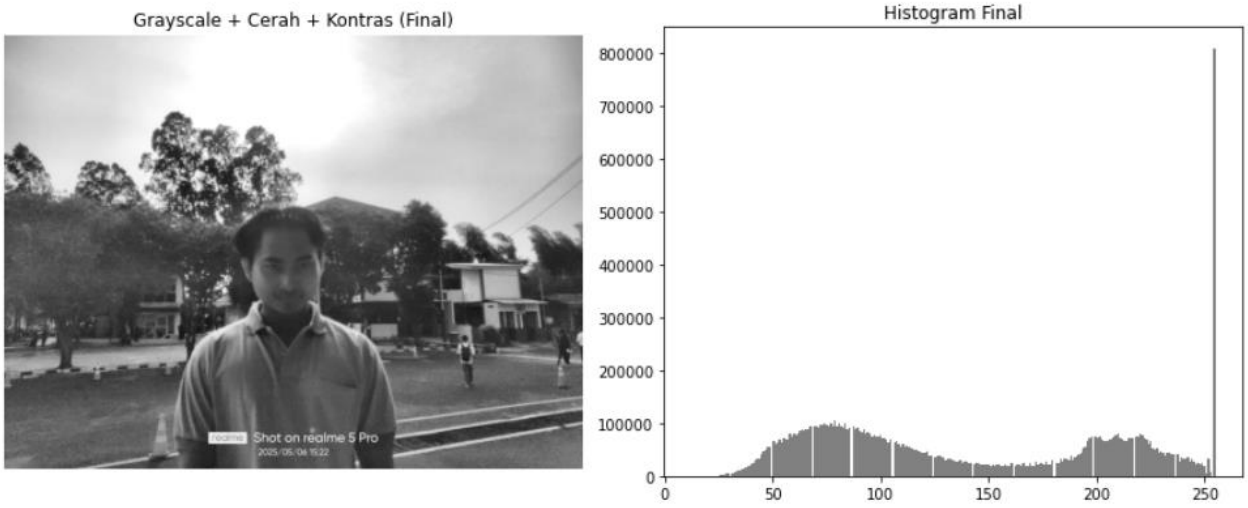
- ax[0].imshow(contrast_only, cmap='gray'): Menampilkan gambar yang sudah diterapkan peningkatan kontras menggunakan CLAHE (Contrast Limited Adaptive Histogram Equalization) di subplot pertama.
- ax[0].set_title('Grayscale + Kontras'): Menambahkan judul pada subplot pertama.
- ax[0].axis('off'): Menghilangkan axis atau sumbu agar gambar terlihat lebih bersih tanpa gangguan.

Membuat Histogram dari Gambar Kontras:

- ax[1].hist(contrast_only.ravel(), bins=256, color='gray'): Membuat histogram berdasarkan piksel dari gambar yang sudah diterapkan kontras.
- ax[1].set_title('Histogram Kontras'): Menambahkan judul pada subplot kedua.

## 5. Gray yang dipercerah dan diperkontras

```
# 5. Grayscale + Cerah + Kontras + Histogram
# ================================
fig, ax = plt.subplots(1, 2, figsize=(12, 5))
ax[0].imshow(contrast_final, cmap='gray')
ax[0].set_title('Grayscale + Cerah + Kontras (Final)')
ax[0].axis('off')
ax[1].hist(contrast_final.ravel(), bins=256, color='gray')
ax[1].set_title('Histogram Final')
plt.tight_layout()
plt.show()


#202331016_Nanda Revan Saputro
```




**Penjelasan:**

Membuat Plot 2x1:

- fig, ax = plt.subplots(1, 2, figsize=(12, 5)): Membuat figure dengan dua kolom untuk menampilkan gambar dengan kecerahan dan kontras yang telah diproses serta histogramnya.

Menampilkan Gambar dengan Cerah dan Kontras:

- ax[0].imshow(contrast_final, cmap='gray'): Menampilkan gambar yang telah diterapkan baik kecerahan (brightening) maupun kontras (CLAHE) di subplot pertama.
- ax[0].set_title('Grayscale + Cerah + Kontras (Final)'): Menambahkan judul pada subplot pertama untuk memberi tahu bahwa ini adalah hasil akhir dari proses pengolahan gambar.
- ax[0].axis('off'): Menghilangkan axis atau sumbu agar gambar tampil lebih bersih tanpa gangguan dari axis.

Membuat Histogram dari Gambar Final:

- ax[1].hist(contrast_final.ravel(), bins=256, color='gray'): Membuat histogram berdasarkan distribusi piksel dari gambar yang sudah diproses.

- ax[1].set_title('Histogram Final'): Menambahkan judul pada subplot kedua yang menunjukkan histogram gambar yang telah diproses.

Tata Letak Plot:

- plt.tight_layout(): Mengatur layout agar gambar dan histogram tidak saling tumpang tindih, sehingga lebih rapi.
- plt.show(): Menampilkan hasil visualisasi gambar dan histogram.

## Perbandingan Gambar Awal vs Final

```
# 6. Perbandingan Awal vs Final
# ================================
fig, ax = plt.subplots(1, 2, figsize=(12, 6))
ax[0].imshow(img_rgb)
ax[0].set_title('Gambar Asli')
ax[0].axis('off')
ax[1].imshow(contrast_final, cmap='gray')
ax[1].set_title('Gambar Akhir (Fokus Profil)')
ax[1].axis('off')
plt.tight_layout()
plt.show()


#202331016_Nanda Revan Saputro
```


**Penjelasan:**

Membuat Dua Area Tampilan (Subplot)

- Membuat figure dengan 2 kolom dalam 1 baris, ukuran plot total 12x6 inci.
- ax berisi dua area untuk menampilkan gambar asli dan hasil akhir.

Menampilkan Gambar Asli

- Menampilkan gambar asli berwarna (img_rgb) pada subplot pertama.
- Diberi judul "Gambar Asli".
- Sumbu koordinat dimatikan agar tampilan lebih bersih.

Menampilkan Gambar yang Sudah Diperbaiki

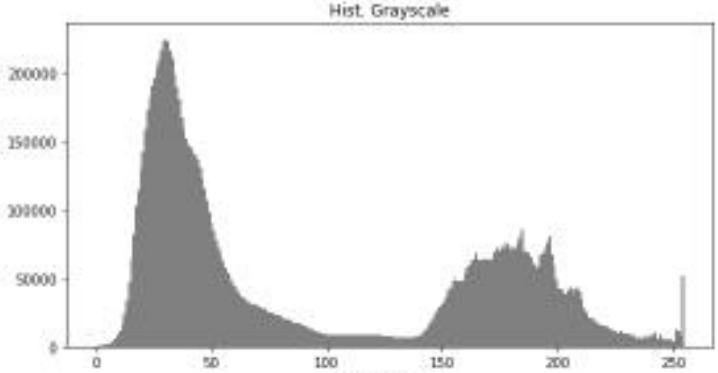
- Menampilkan gambar hasil akhir (contrast_final) dalam skala abu-abu.
- Judul "Gambar Akhir (Fokus Profil)" menjelaskan bahwa gambar telah difokuskan pada peningkatan kualitas wajah atau objek utama.
- Sumbu dimatikan untuk tampilan bersih.

## Tampilkan Semua

```
# 7. Semua Tahap + Histogram (Grid)
# ================================
fig, axs = plt.subplots(5, 2, figsize=(14, 20))

# Tahap 1
axs[0,0].imshow(gray, cmap='gray')
axs[0,0].set_title('Grayscale')
axs[0,0].axis('off')
axs[0,1].hist(gray.ravel(), bins=256, color='gray')
axs[0,1].set_title('Hist. Grayscale')

# Tahap 2
axs[1,0].imshow(bright, cmap='gray')
axs[1,0].set_title('Grayscale + Cerah')
axs[1,0].axis('off')
axs[1,1].hist(bright.ravel(), bins=256, color='gray')
axs[1,1].set_title('Hist. Cerah')

# Tahap 3
axs[2,0].imshow(contrast_only, cmap='gray')
axs[2,0].set_title('Grayscale + Kontras')
axs[2,0].axis('off')
axs[2,1].hist(contrast_only.ravel(), bins=256, color='gray')
axs[2,1].set_title('Hist. Kontras')

# Tahap 4
axs[3,0].imshow(contrast_final, cmap='gray')
axs[3,0].set_title('Cerah + Kontras (Final)')
axs[3,0].axis('off')
axs[3,1].hist(contrast_final.ravel(), bins=256, color='gray')
axs[3,1].set_title('Hist. Final')

# Tahap 5
axs[4,0].imshow(img_rgb)
axs[4,0].set_title('Gambar Asli (RGB)')
axs[4,0].axis('off')
axs[4,1].imshow(contrast_final, cmap='gray')
axs[4,1].set_title('Gambar Akhir (Final)')
axs[4,1].axis('off')

plt.tight_layout()
plt.show()


#202331016_Nanda Revan Saputro
```

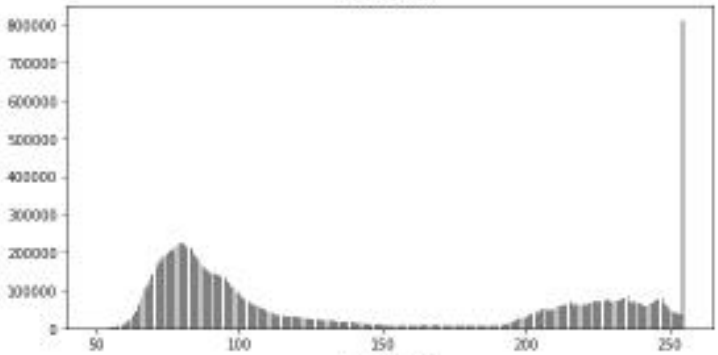
Grayscale

Hist. Grayscale



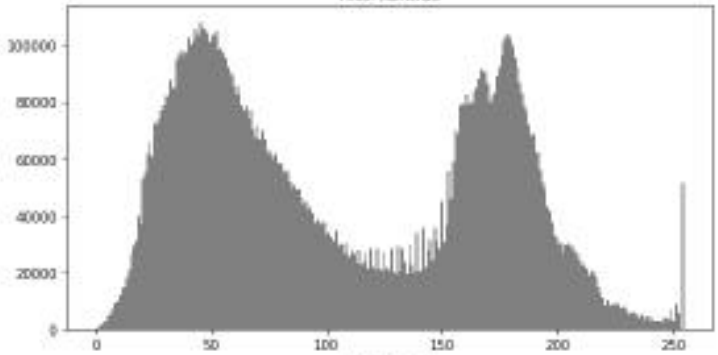

Grayscale + Cerah

Hist. Cerah



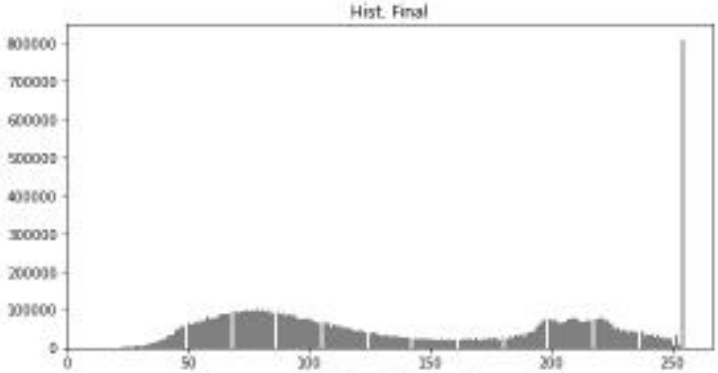

Grayscale + Kontras

Hist. Kontras




Cerah + Kontras (Final)

Hist. Final




Gambar Asli (RGB)

Gambar Akhir (Final)

**Selesai**

**Sekarang ke tiga soal sudah selesai.**

# BAB IV

# PENUTUP

## KESIMPULAN

**Berdasarkan hasil praktikum UTS Pengolahan Citra Digital, dapat disimpulkan bahwa:**

1. Deteksi warna pada citra merupakan langkah awal yang penting dalam proses segmentasi dan identifikasi objek visual. Dengan menggunakan metode RGB atau HSV, objek berwarna tertentu dapat dikenali secara akurat.
2. Penggunaan histogram dan thresholding sangat efektif untuk memisahkan objek dari latar belakang. Thresholding membantu menentukan batas intensitas yang memisahkan piksel penting dan tidak penting.
3. Citra backlight dapat diperbaiki melalui teknik pemrosesan seperti algoritma Retinex dan peningkatan kontras, sehingga objek utama menjadi lebih jelas tanpa merusak struktur gambar.
4. Konversi ke grayscale menyederhanakan proses komputasi dan digunakan sebagai tahap awal dalam berbagai analisis citra seperti deteksi tepi dan segmentasi bentuk.
5. Segmentasi warna otomatis dengan dua tahap thresholding berbasis histogram mampu meningkatkan akurasi segmentasi terutama pada citra dengan warna dominan yang jelas.

Secara keseluruhan, praktikum ini memberikan pemahaman menyeluruh tentang bagaimana citra digital dapat diolah dan dianalisis menggunakan berbagai metode teknis. Penguasaan materi ini sangat penting dalam pengembangan aplikasi di bidang pengenalan wajah, robotika, sistem pemantauan, dan berbagai sistem berbasis visual lainnya.

## DAFTAR PUSTAKA

Boldaji, M. N., & Semnani, S. H. (2021). Color image segmentation using multi-objective particle swarm optimization and multi-level histogram thresholding. arXiv preprint arXiv:2110.09217. https://arxiv.org/abs/2110.09217

Barron, J. T. (2020). Generalized Histogram Thresholding. arXiv preprint arXiv:2007.07350. https://arxiv.org/abs/2007.07350

Rani, S., Bhatia, S., Kumar, R., & Saini, R. (2023). A hybrid image enhancement approach using histogram equalization and adaptive gamma correction. Heliyon, 9(5), e13644. https://doi.org/10.1016/j.heliyon.2023.e13644

Cao, J., Zhang, Y., & Zhu, Q. (2023). An Automatic Color Image Segmentation Method Based on Two-Stage Histogram Thresholding. Sensors, 23(6), 3361. https://doi.org/10.3390/s23063361