Tugas Hands On 2

Nama : Marsella Yesi Natalia Sinaga

NIM : 121140174

Sistem/Teknologi Multimedia

SOAL 1

Dengan foto anda sendiri, lakukanlah eksperimen berikut ini. Editlah sebuah foto yang terdapat wajah anda, namun kali ini tidak dengan photoshop, melainkan dengan python. Lakukan penyesuaian berikut ini.

- Lakukan resize untuk foto anda menjadi 1080 pada dimension terpanjangnya. Gunakan cv2.resize untuk melakukan resize.
- Buatlah frame berwarna kuning (RGB value: 255, 255, 0) sebesar 25 pixel + dua digit terakhir NIM anda pada setiap sisi foto anda. Frame tersebut akan mengelilingi foto setebal 25 pixel + dua digit terakhir NIM anda.
- Aturlah intensitas warna pada channel merah di bagian tengah (50 - 150), dan naikkan sebesar 20 poin
- Turunkanlah intensitas warna pada channel biru di atas (200- 250), dan turunkan sebesar 20 poin
- Tampilkanlah histogram dari foto tersebut
- Jelaskan hasil eksperimen anda

Import Library

In [37]:
```
import numpy as np
import matplotlib.pyplot as plt
import cv2 # opencv
import os
```

Membaca Gambar

In [38]:
```
path_img = os.path.join(os.getcwd(), 'data', 'marsella.jpeg')

# mengecek apakah file gambar ada
# jika tidak ada, akan tercetak pesan "File tidak ditemukan: <path>"
if not os.path.exists(path_img):
    print('File tidak ditemukan:', path_img)
    exit()
```

Loading Gambar

In [39]:
```
img = cv2.imread(path_img)
```

Ubah Gambar BGR ke RGB

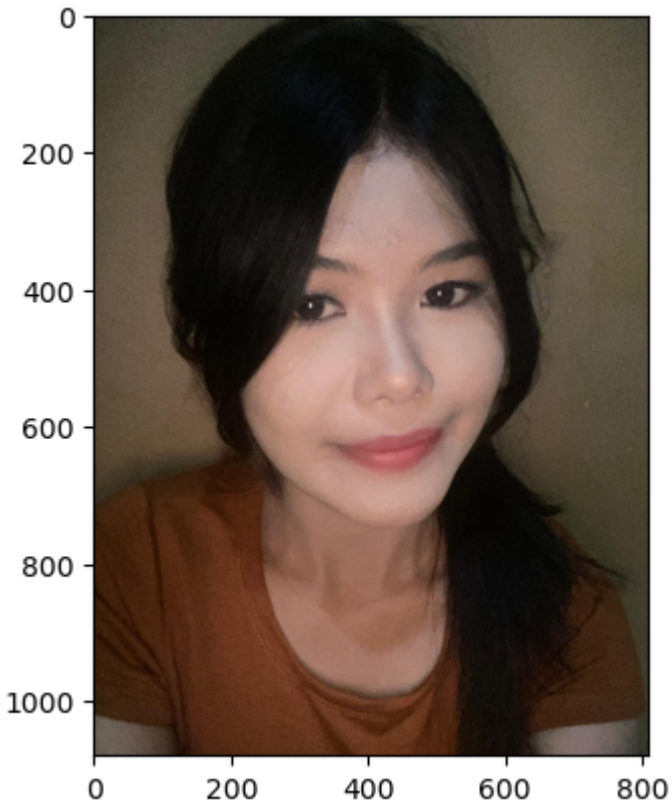
In [172…

```
# Mengubah gambar bgr ke rgb
img = cv2.imread(path_img) # masih dalam format BGR
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) # konversi ke format RGB

# Menampilkan gambar
plt.axis ('off')
plt.imshow(img)
plt.show()
```

Resize Gambar

In [173…

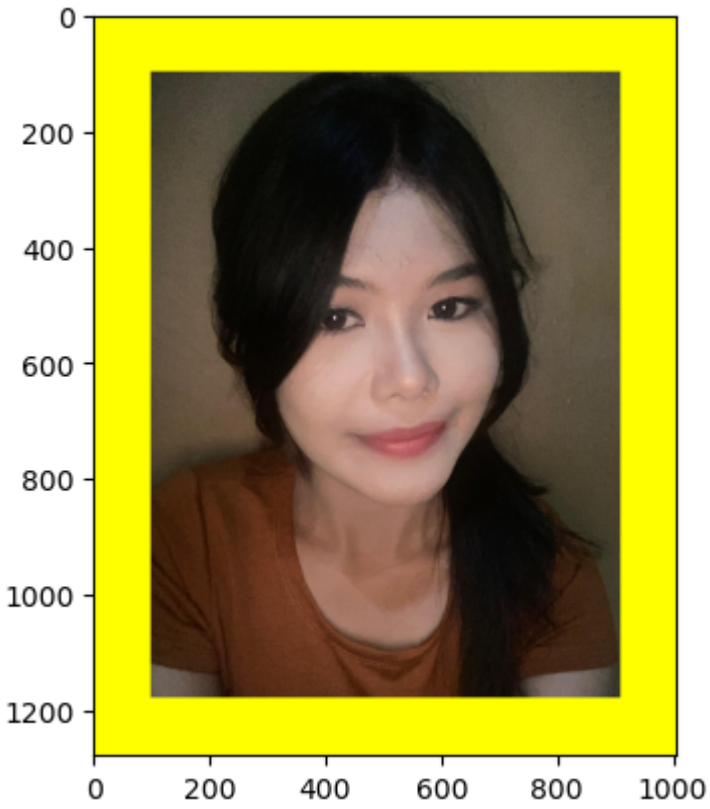
```
# Menghitung rasio untuk resize
height, width = img.shape[:2]
if height > width:
    new_height = 1080
    new_width = int((1080 / height) * width)
else:
    new_width = 1080
    new_height = int((1080 / width) * height)

# Resize foto
resized_img = cv2.resize(img, (new_width, new_height))
plt.imshow(resized_img)
plt.show()
```



Membuat Frame

In [217…

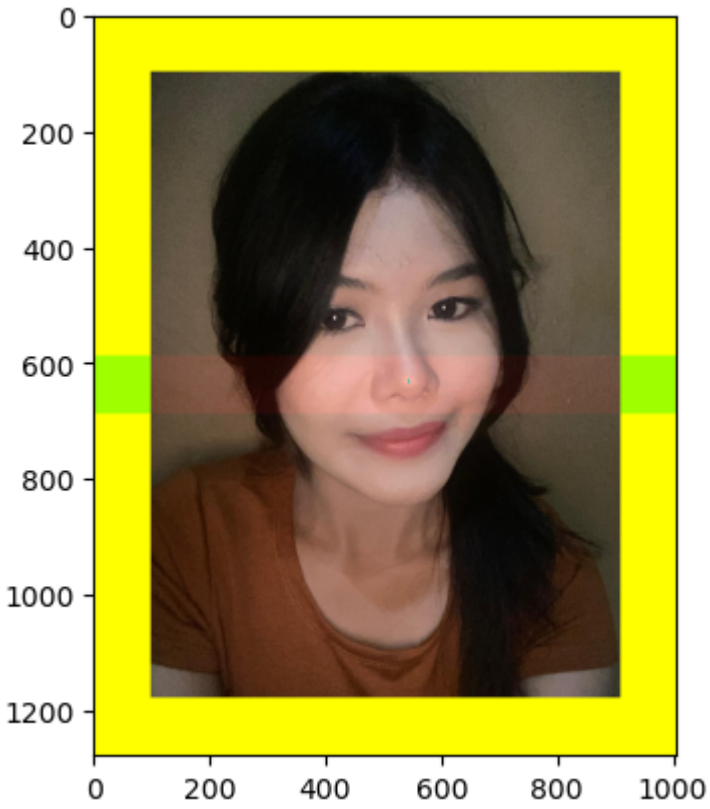
```
# Menghitung dimensi bingkai
frame_thickness = 25 + 74
yellow_frame = np.zeros((resized_img.shape[0] + frame_thickness * 2, resized_img
yellow_frame[:] = (255, 255, 0)

# Menempatkan gambar di tengah bingkai
yellow_frame[frame_thickness:frame_thickness + resized_img.shape[0], frame_thick
plt.imshow(yellow_frame)
plt.show()
```



Mengatur Intensitas Warna

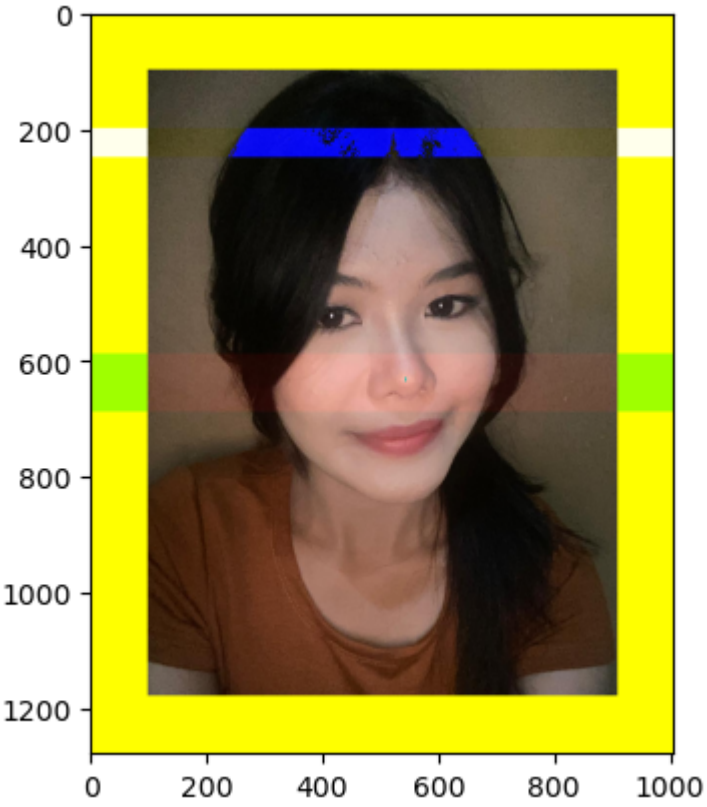
In [226…

```
# Naikkan channel merah di bagian tengah (50-150)
yellow_frame[frame_thickness:frame_thickness + resized_img.shape[0], frame_thick
middle_row_start = yellow_frame.shape[0] // 2 - 50
middle_row_end = yellow_frame.shape[0] // 2 + 50

# Menaikkan intensitas channel merah sebesar 20 poin
yellow_frame[middle_row_start:middle_row_end, :, 0] = np.clip(
    yellow_frame[middle_row_start:middle_row_end, :, 0] + 20, 0, 255
)
plt.imshow(yellow_frame)
plt.show()
```



In [227…

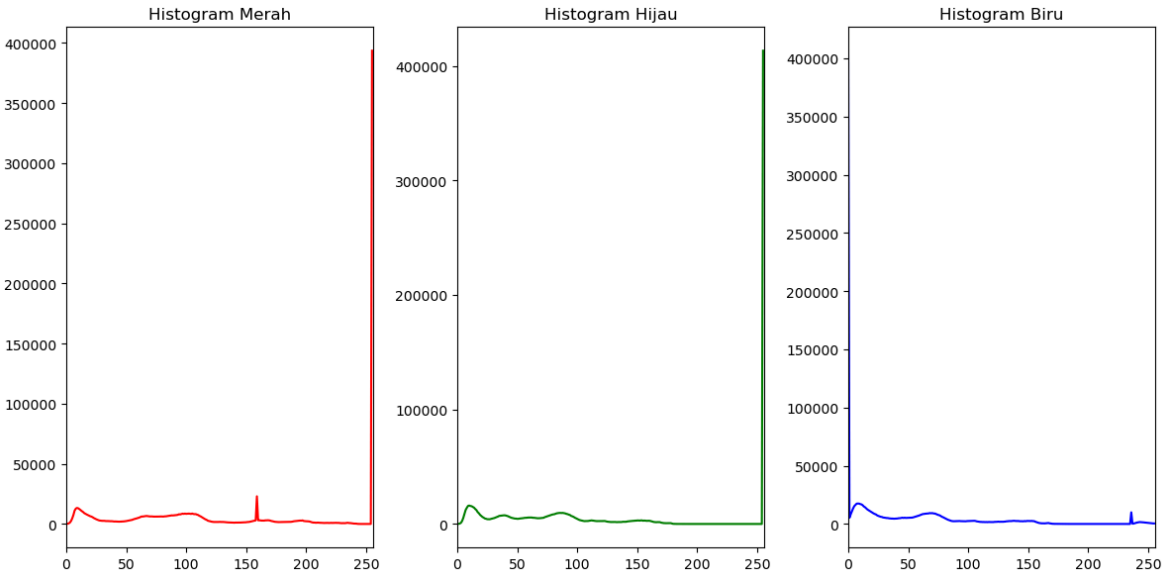
```
# Turunkan intensitas channel biru di bagian atas (200-250)
blue_row_start = 200
blue_row_end = 250

# Menurunkan intensitas channel biru sebesar 20 poin
yellow_frame[blue_row_start:blue_row_end, :, 2] = np.clip(
    yellow_frame[blue_row_start:blue_row_end, :, 2] - 20, 0, 255
)
plt.imshow(yellow_frame)
plt.show()
```



Tampilan Histogram

In [233…

```
# Menampilkan histogram dari gambar yang telah dimodifikasi
plt.figure(figsize=(12, 6))

# Menampilkan Histogram
red_hist, bins = np.histogram(yellow_frame[:, :, 0].ravel(), bins=256, range=[0,
plt.subplot(1, 3, 1)
plt.plot(bins[:-1], red_hist, color='red')
plt.xlim([0, 256])
plt.title('Histogram Merah')

# Histogram untuk channel hijau
green_hist, bins = np.histogram(yellow_frame[:, :, 1].ravel(), bins=256, range=[
plt.subplot(1, 3, 2)
plt.plot(bins[:-1], green_hist, color='green')
plt.xlim([0, 256])
plt.title('Histogram Hijau')

# Histogram untuk channel biru
blue_hist, bins = np.histogram(yellow_frame[:, :, 2].ravel(), bins=256, range=[0
plt.subplot(1, 3, 3)
plt.plot(bins[:-1], blue_hist, color='blue')
plt.xlim([0, 256])
plt.title('Histogram Biru')

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



Hasil Eksperimen

- Resizing: Foto diubah ukurannya sehingga dimensi terpanjangnya menjadi 1080 piksel, memastikan ukuran yang sesuai untuk analisis lebih lanjut.
- Frame Kuning: Bingkai berwarna kuning (RGB: 255, 255, 0) dengan ketebalan 25 piksel ditambah dua digit terakhir dari NIM ditambahkan di setiap sisi foto, memberikan efek visual yang menarik.
- Penyesuaian Intensitas Warna:

  Channel Merah: Intensitas pada channel merah di bagian tengah dinaikkan sebesar 20 poin, membuat area tersebut lebih cerah. Channel Biru: Intensitas pada channel biru untuk piksel dengan nilai 200-250 diturunkan sebesar 20 poin, mengurangi saturasi warna biru.
- Histogram: Setelah semua modifikasi dilakukan, histogram dari foto yang telah diedit ditampilkan. Histogram ini menunjukkan distribusi intensitas piksel untuk setiap channel warna (merah, hijau, dan biru), memberikan gambaran visual tentang bagaimana perubahan yang dilakukan memengaruhi komposisi warna keseluruhan gambar.

SOAL 2

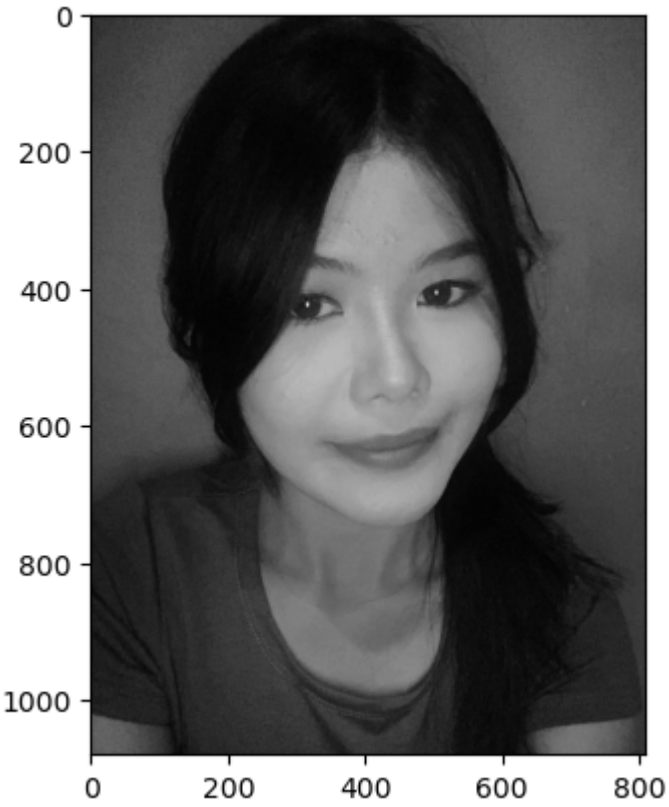
Dengan foto anda sendiri, lakukanlah eksperimen berikut ini

- Convert image dari RGB menjadi Grayscale
- Tampilkan histogram dari foto tersebut
- Lakukanlah normalisasi level intensitas warna pada foto tersebut. Aturlah agar intensitas warna terendah menjadi 0, dan intensitas warna tertinggi menjadi 255
- Tampilkan histogram dari hasil normalisasi
- Buatlah pixel-art dari foto tersebut. Caranya adalah dengan mengubah intensitas warna menjadi 0 atau 255. Jika intensitas warna < 128, maka ubah menjadi 0, dan

jika intensitas warna >= 128, maka ubah menjadi 255. Tunjukkan hasilnya

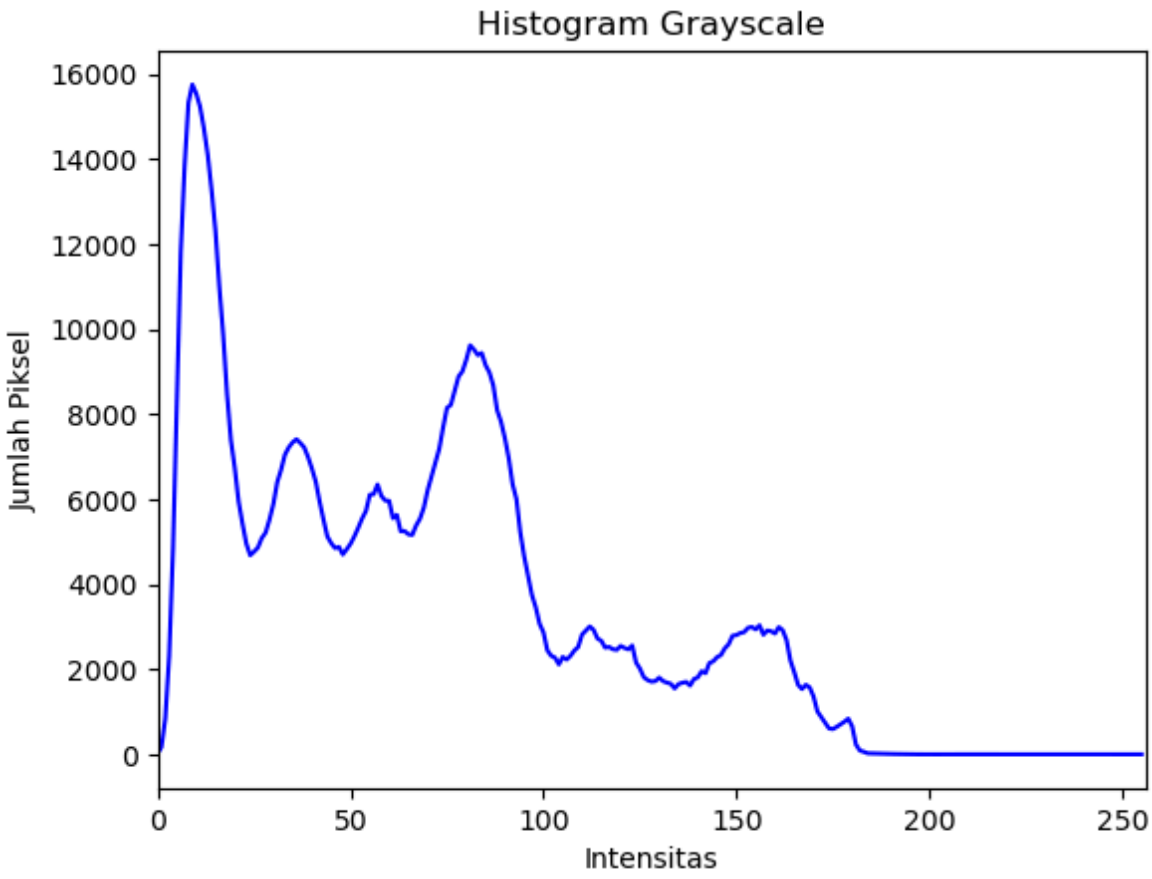
Convert Image dari RGB menjadi Grayscale

In [86]:
```
gray_img = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
plt.imshow(gray_img, cmap='gray')
plt.show()
```



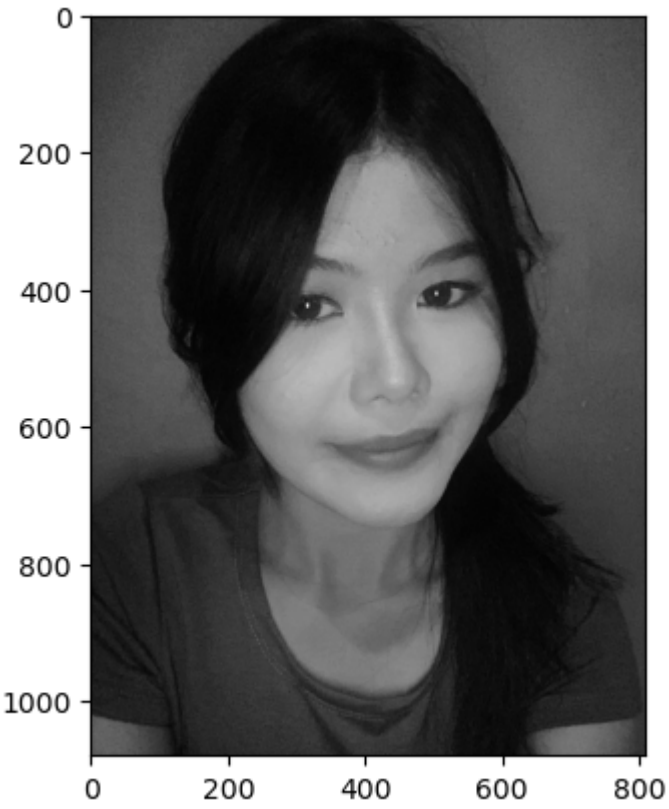
Tampilan Histogram

In [98]:
```
# Hitung histogram
hist_gray = cv2.calcHist([gray_img], [0], None, [256], [0, 256])

# Tampilkan histogram
plt.plot(hist_gray, color='blue')
plt.xlim([0, 256])
plt.title("Histogram Grayscale")
plt.xlabel("Intensitas")
plt.ylabel("Jumlah Piksel")
plt.show()
```



Normalisasi level intensitas warna

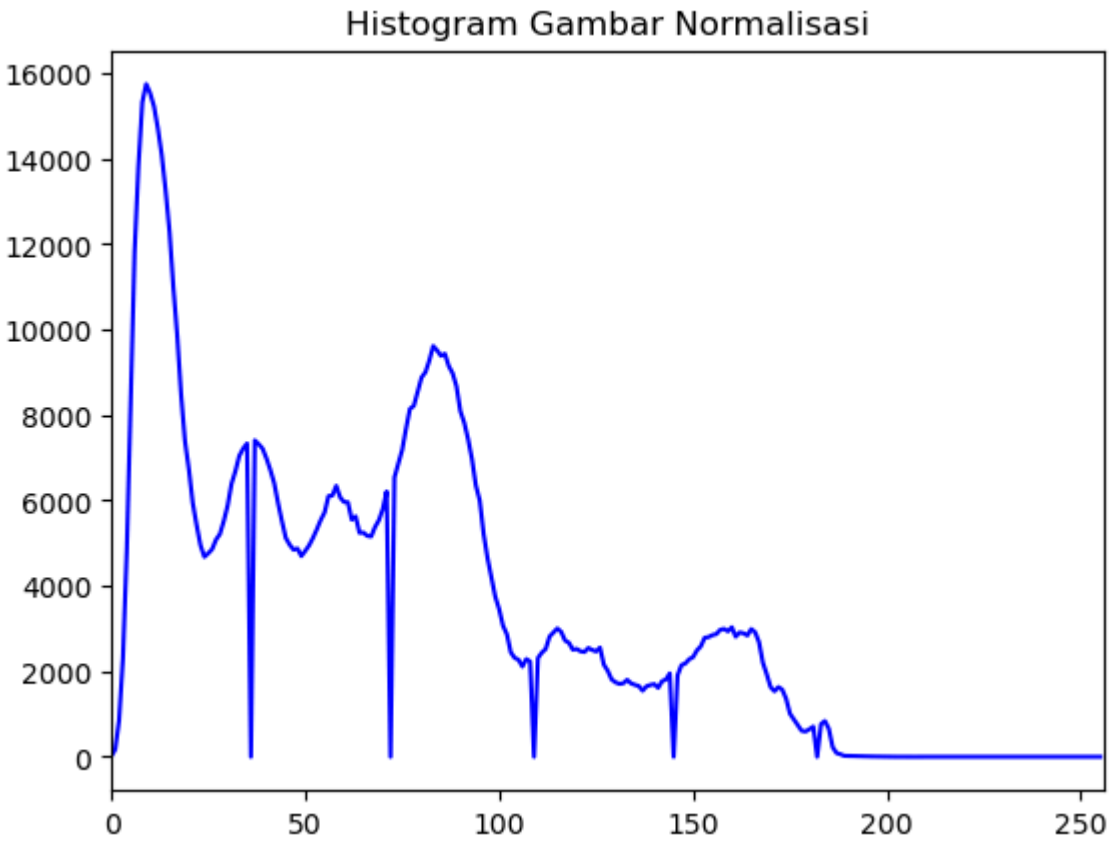
In [99]:
```
# Normalisasi level intensitas
min_val = np.min(gray_img)
max_val = np.max(gray_img)
normalized_img = (gray_img - min_val) * (255 / (max_val - min_val))
normalized_img = normalized_img.astype(np.uint8)
plt.imshow(normalized_img, cmap='gray')
plt.show()
```



Tampilan Histogram dari Hasil Normalisasi

In [132…

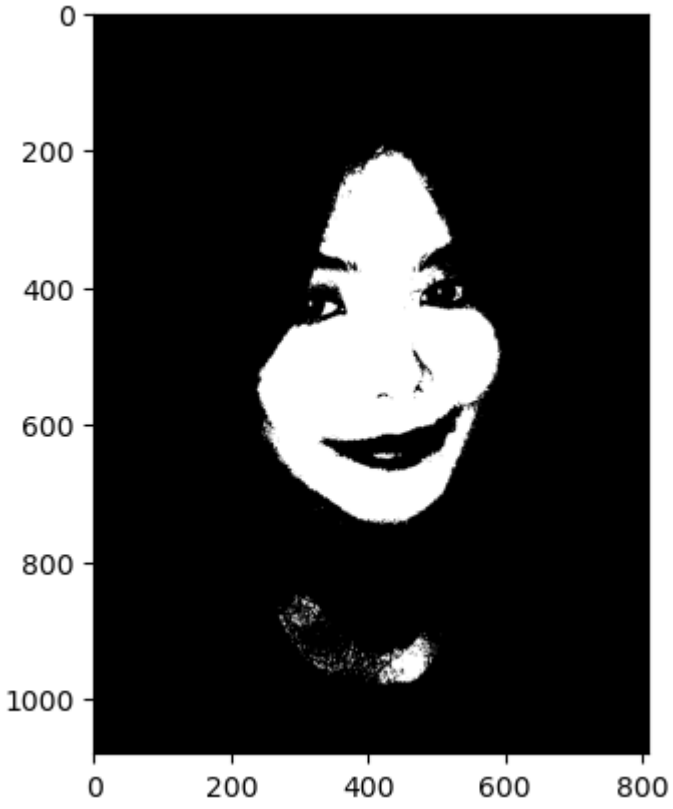
```
# Hitung histogram untuk gambar yang dinormalisasi
hist_normalized = cv2.calcHist([normalized_img], [0], None, [256], [0, 256])

# Tampilkan histogram
plt.plot(hist_normalized, color='blue')
plt.xlim([0, 256])
plt.title("Histogram Gambar Normalisasi")
plt.show()
```



Membuat Pixel-Art

In [133…

```
# Ubah intensitas warna menjadi 0 atau 255
pixel_art_img = np.where(normalized_img < 128, 0, 255).astype(np.uint8)
plt.imshow(pixel_art_img, cmap='gray')
plt.show()
```



Hasil Eksperimen

- Konversi Gambar dari RGB ke Grayscale: Gambar yang berisi wajah saya diubah menjadi format grayscale, di mana setiap piksel pada gambar baru merepresentasikan tingkat kecerahan, dengan nilai antara 0 (hitam) hingga 255 (putih). Proses ini menghilangkan informasi warna, sehingga fokus hanya pada kontras dan pencahayaan gambar.

- Histogram Gambar Grayscale: Histogram dari gambar grayscale ditampilkan untuk menunjukkan distribusi intensitas piksel. Histogram ini memberikan informasi tentang sebaran kecerahan dalam gambar, mengindikasikan seberapa banyak piksel berada pada masing-masing level intensitas.

- Normalisasi Level Intensitas Warna: Setelah konversi, dilakukan normalisasi pada intensitas warna agar nilai terendah menjadi 0 dan tertinggi menjadi 255. Proses ini meningkatkan kontras gambar dengan merentangkan nilai intensitas, sehingga detail gambar menjadi lebih jelas dan lebih terlihat.

- Histogram Setelah Normalisasi: Histogram dari gambar yang sudah dinormalisasi ditampilkan. Histogram ini menunjukkan perubahan distribusi intensitas setelah normalisasi, yang seharusnya menunjukkan rentang yang lebih luas dibandingkan histogram sebelumnya.

- Membuat Pixel-Art: Gambar grayscale diubah menjadi format pixel-art dengan mengganti intensitas warna. Jika nilai intensitas kurang dari 128, maka diubah menjadi 0 (hitam); jika lebih besar atau sama dengan 128, maka diubah menjadi 255 (putih). Hasilnya adalah gambar yang terlihat seperti pixel-art, di mana hanya dua warna (hitam dan putih) yang digunakan.

SOAL 3

Dengan foto anda sendiri, lakukanlah eksperimen berikut ini

- Naikkan kecerahan (brightnes) pada foto tersebut. Anda tidak boleh menggunakan library cv2 untuk menaikkan kecerahan. Anda hanya boleh menggunakan operasi matriks. Jelaskan langkah-langkah yang anda lakukan. Tampilkan histogram dari foto tersebut
- Turunkan saturasi warna pada foto tersebut. Anda tidak boleh menggunakan library cv2 untuk menurunkan saturasi warna. Anda hanya boleh menggunakan operasi matriks. Jelaskan langkah-langkah yang anda lakukan. Tampilkan histogram dari foto tersebut
- Turunkan kontras pada foto tersebut. Anda tidak boleh menggunakan library cv2 untuk menurunkan kontras. Anda hanya boleh menggunakan operasi matriks. Jelaskan langkah-langkah yang anda lakukan. Tampilkan histogram dari foto tersebut

Membaca Gambar

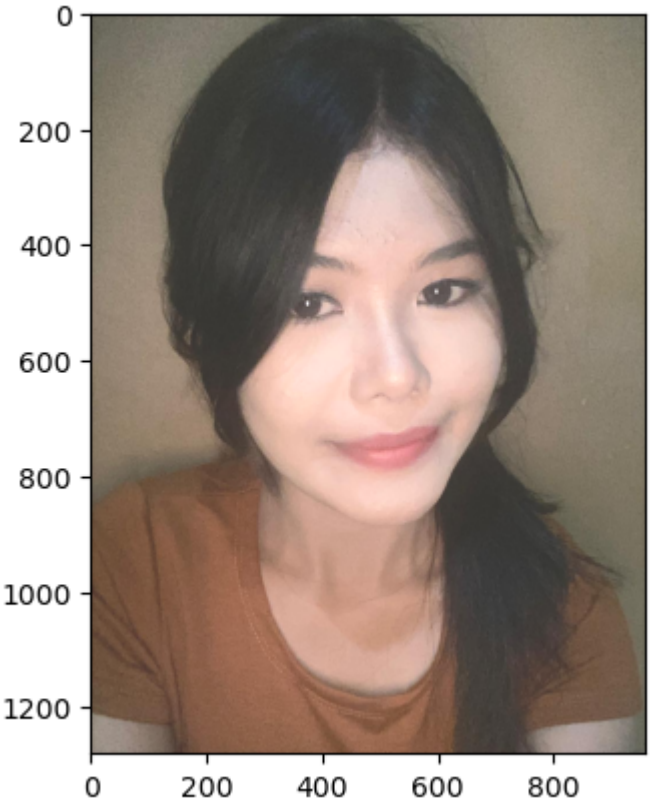
In [134…

```python
path_img = os.path.join(os.getcwd(), 'data', 'marsella.jpeg')

# mengecek apakah file gambar ada
# jika tidak ada, akan tercetak pesan "File tidak ditemukan: <path>"
if not os.path.exists(path_img):
    print('File tidak ditemukan:', path_img)
    exit()
img = plt.imread(path_img)

# Menampilkan gambar
plt.axis ('off')
plt.imshow(img)
plt.show()
```

Menaikkan Kecerahan (Brightnes)

In [135…

```python
# Manipulasi Gambar ke format Float
img = img.astype(np.float32)

# Menaikkan kecerahan
brightness_increase = 50
brightened_img = np.clip(img + brightness_increase, 0, 255)
plt.imshow(brightened_img.astype(np.uint8))
plt.show()
```



Penjelasan:

1. **Manipulasi Gambar ke format Float**

   Untuk mengonversi tipe data gambar dari format integer ke format float32. karna Operasi aritmatika dalam penambahan kecerahan, dapat menghasilkan nilai yang lebih besar dari 255 (nilai maksimum untuk uint8). Jika tidak mengonversi ke float, penambahan ini dapat menyebabkan overflow (nilai yang melebihi batas maksimum) dan menghasilkan artefak yang tidak diinginkan. Dengan menggunakan float32, maka dapat menangani nilai yang lebih besar tanpa kehilangan informasi. untuk hasil Gambar sekarang berada dalam format float, dengan nilai pixel yang dapat berkisar dari 0 hingga lebih dari 255.

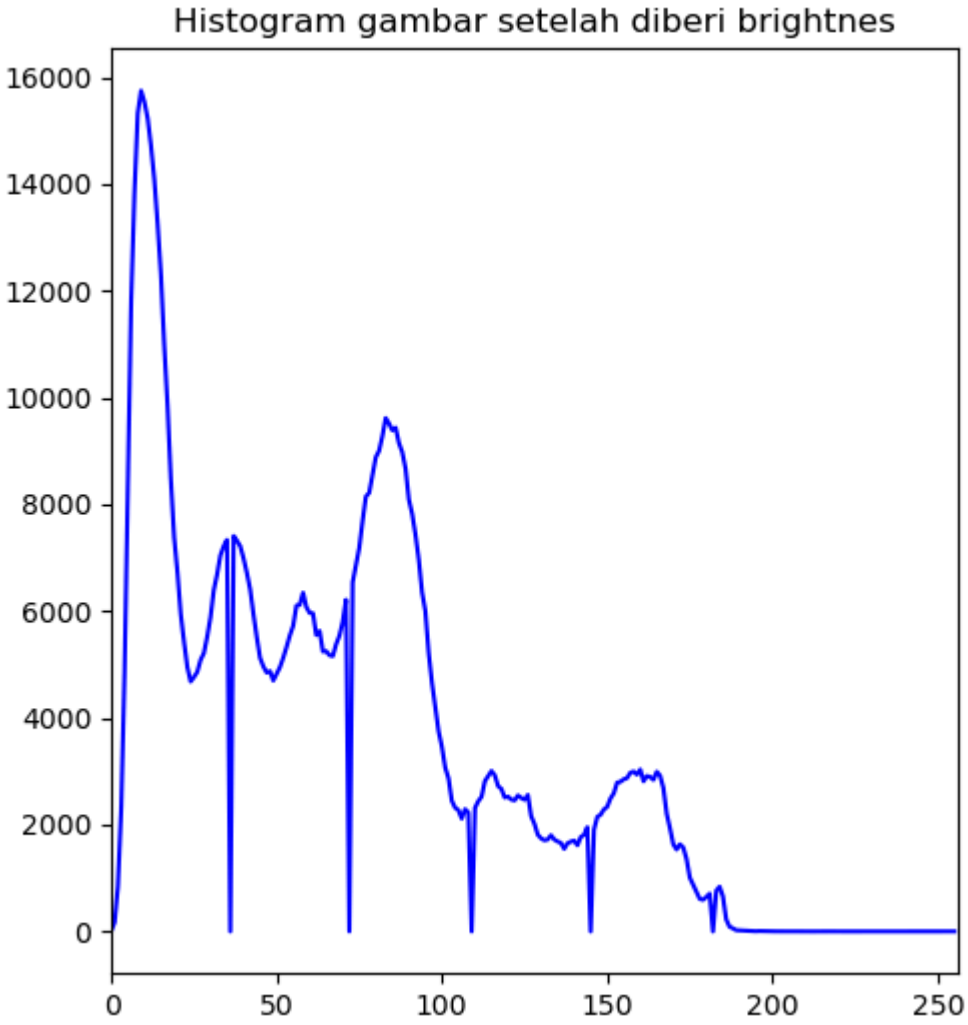
2. **Menaikkan kecerahan**

   - img + brightness_increase: untuk menambahkan nilai brightness_increase (50) ke setiap pixel dalam gambar. Ini berarti setiap pixel akan menjadi lebih terang dengan nilai 50.
   - np.clip(..., 0, 255): Fungsi np.clip digunakan untuk membatasi nilai pixel agar tetap dalam rentang 0 hingga 255. Jika hasil penambahan kecerahan melebihi 255, maka nilai tersebut akan dipotong menjadi 255. Jika ada nilai yang kurang dari 0 (meskipun dalam konteks ini tidak mungkin karena sudah menggunakan float), maka nilai tersebut akan dipotong menjadi 0.
   - gambar yang telah ditingkatkan kecerahannya, dengan semua nilai pixel yang berada dalam rentang yang valid (0-255).

Tampilan Histogram

In [156…

```
# Menampilkan histogram
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 2)
plt.plot(hist_normalized, color='blue')
plt.title('Histogram gambar setelah diberi brightnes')
plt.xlim([0, 256])
plt.show()
```



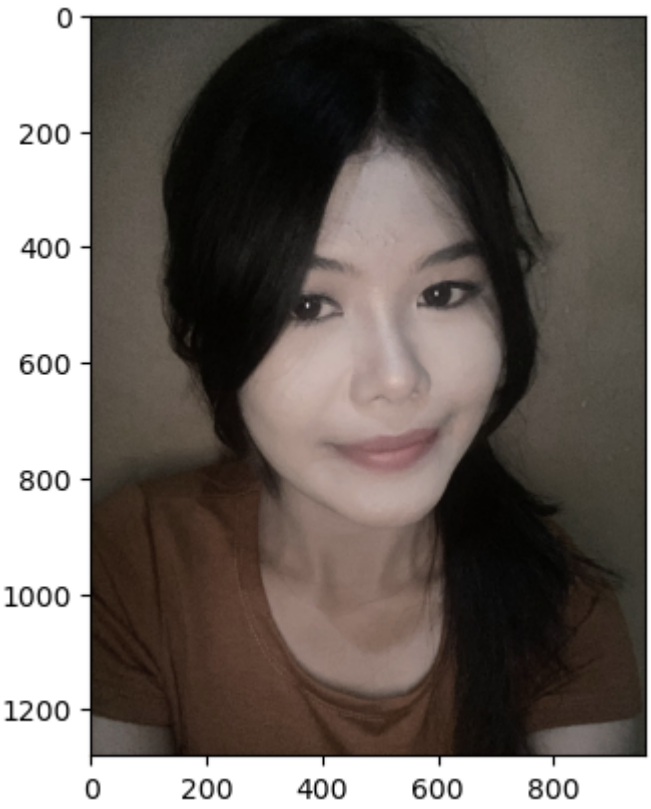
Menurunkan Saturasi Warna

In [165…

```
# Menghitung gambar grayscale dengan mengambil rata-rata dari R, G, B
gray_image = np.mean(img, axis=2, keepdims=True).astype(np.uint8)

# Menentukan faktor pengurangan saturasi
saturation_factor = 0.5

# Mengurangi saturasi dengan mencampur gambar grayscale dengan gambar asli
desaturated_image = (img * saturation_factor + gray_image * (1 - saturation_fact
plt.imshow(desaturated_image)
plt.show
```

Out[165…

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



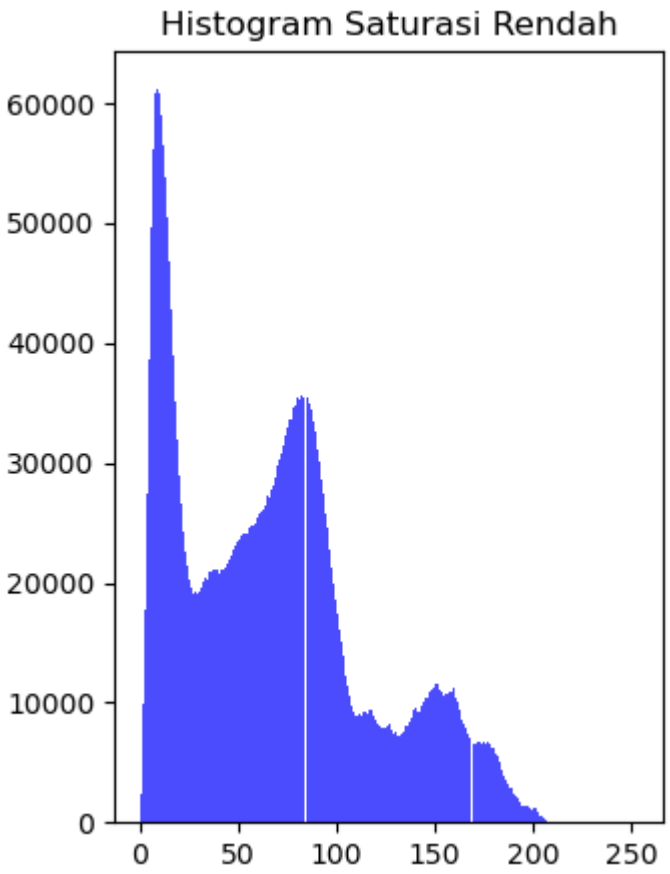
Penjelasan :

- pertama menghitung gambar grayscale dengan mengambil rata-rata nilai R, G, dan B setiap piksel menggunakan fungsi np.mean(img, axis=2, keepdims=True), yang menghasilkan matriks grayscale berukuran (tinggi, lebar, 1).
- Selanjutnya, menetapkan faktor pengurangan saturasi sebesar 0.5, yang berarti gambar akan memiliki saturasi setengah dari saturasi aslinya.
- Kemudian, mengurangi saturasi dengan mencampurkan gambar asli yang telah dikalikan dengan faktor pengurangan saturasi dan gambar grayscale yang telah dikalikan dengan (1 - saturation_factor), menghasilkan gambar akhir desaturated_image yang memiliki warna lebih lembut.
- Terakhir, gambar hasil ini ditampilkan menggunakan plt.imshow(desaturated_image) diikuti dengan plt.show() untuk memperlihatkan efek pengurangan saturasi yang telah diterapkan.

Tampilan Histogram

In [163…

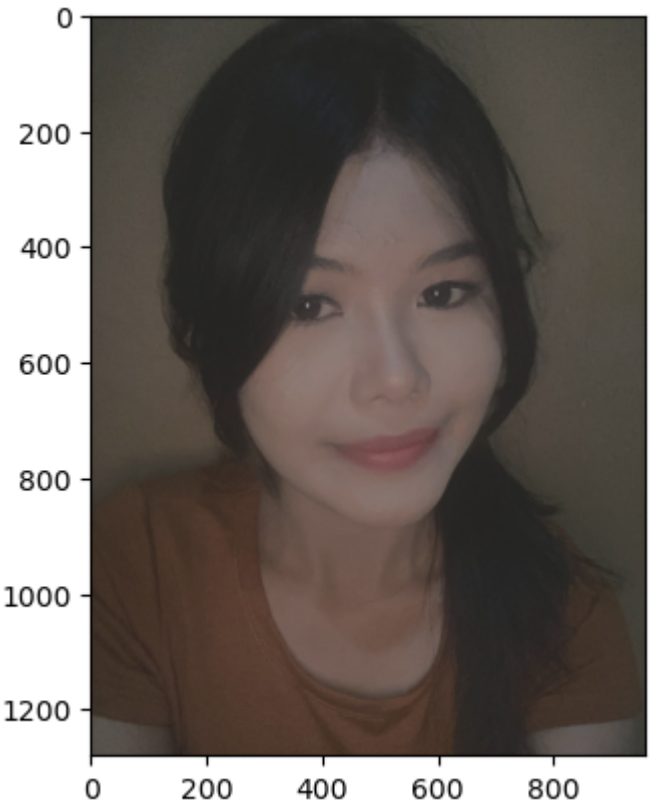
```
# Histogram gambar yang telah dikurangi saturasinya
plt.figure(figsize=(12, 5))
plt.subplot(1, 3, 3)
plt.hist(desaturated_image.ravel(), bins=256, color='blue', alpha=0.7)
plt.title('Histogram Saturasi Rendah')
plt.show()
```



Menurunkan Kontras

In [154…

```
# Menghitung rata-rata intensitas gambar
mean_intensity = np.mean(img)

# Menentukan faktor pengurangan kontras
contrast_factor = 0.5

# Mengurangi kontras menggunakan formula
low_contrast_image = ((img - mean_intensity) * contrast_factor + mean_intensity)

# Membatasi nilai piksel agar tetap dalam rentang 0-255
low_contrast_image = np.clip(low_contrast_image, 0, 255)
plt.imshow(low_contrast_image)
plt.show()
```



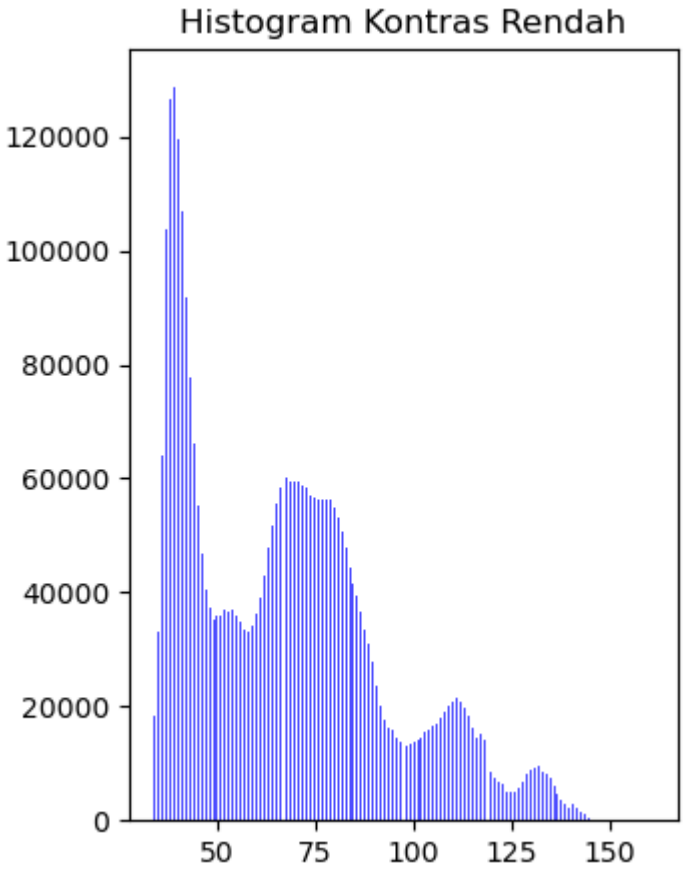
Penjelasan :

- mean_intensity = np.mean(img) digunakan sebagai acuan untuk menurunkan kontras.
- Nilai contrast_factor = 0.5 mengatur seberapa besar kontras dikurangi—dalam hal ini, setengahnya.
- Formula ((img - mean_intensity) * contrast_factor + mean_intensity) mendekatkan intensitas tiap piksel ke rata-rata, menurunkan perbedaan intensitas di gambar, sehingga kontras berkurang.
- Setelah itu, np.clip(low_contrast_image, 0, 255) membatasi nilai piksel agar tetap dalam rentang 0-255, sehingga tidak ada distorsi akibat intensitas yang terlalu tinggi atau rendah.
- Gambar hasil dengan kontras yang sudah berkurang kemudian ditampilkan menggunakan plt.imshow(low_contrast_image) dan plt.show().

Tampilan Histogram

In [164…

```
# Histogram gambar yang telah dikurangi kontrasnya
plt.figure(figsize=(12, 5))
plt.subplot(1, 3, 3)
plt.hist(low_contrast_image.ravel(), bins=256, color='blue', alpha=0.7)
plt.title('Histogram Kontras Rendah')
plt.show()
```



Hasil Eksperimen

- Peningkatan Kecerahan : Kecerahan foto ditingkatkan dengan menambahkan nilai konstan (misalnya, 50) ke setiap piksel dalam gambar. Untuk mencegah overflow, nilai yang melebihi 255 dibatasi menggunakan fungsi np.clip. Hasilnya, gambar menjadi lebih terang, dan histogram menunjukkan peningkatan jumlah piksel di area terang, menunjukkan pergeseran distribusi intensitas ke kanan.

- Penurunan Saturasi : Saturasi dikurangi dengan mencampurkan gambar asli dengan versi grayscale-nya. Grayscale dihitung dengan mengambil rata-rata nilai R, G, dan B untuk setiap piksel. Gambar hasil campuran ini memberikan efek warna yang lebih lembut. Histogram menunjukkan distribusi warna yang lebih merata, menandakan bahwa warna-warna telah berkurang intensitasnya.

- Penurunan Kontras : Kontras diturunkan dengan mendekatkan nilai piksel ke rata-rata intensitas gambar. Setiap nilai piksel dikurangi dari rata-rata berdasarkan faktor pengurangan kontras (misalnya, 0.5). Akibatnya, gambar terlihat lebih datar dan kurang tajam, sementara histogram menunjukkan penyebaran intensitas yang lebih sempit dengan lebih banyak piksel mendekati nilai rata-rata, menandakan penurunan kontras yang efektif.