UTS

PENGOLAHAN CITRA

NAMA : Hania Mutia Khalisa

NIM : 202331279

KELAS : C

DOSEN : Ir. Darma Rusjdi, M.Kom

NO.PC : 26

ASISTEN : 1. Abdur Rasyid Ridho

2. Rizqy Amanda

3. Kashrina Masyid Azka

4. Izzat Islami Kagapi

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2024/2025

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

1. Bagaimana cara mendeteksi warna merah, hijau, dan biru secara akurat pada sebuah citra digital yang diambil secara mandiri?
2. Bagaimana menentukan dan mengurutkan nilai ambang batas terkecil hingga terbesar dalam citra untuk membedakan komponen warna secara optimal?
3. Bagaimana cara membangun program Python yang dapat menampilkan deteksi warna serta histogram dari citra yang dianalisis?
4. Apa efek dari kondisi pencahayaan belakang (backlight) terhadap kualitas citra wajah, dan bagaimana cara memperbaikinya dengan teknik pengolahan citra?
5. Seberapa efektif peningkatan kecerahan dan kontras pada gambar grayscale dalam menonjolkan profil wajah pada citra backlight?

1.2 Tujuan Masalah

1. Mengembangkan program Python yang dapat mendeteksi warna primer (merah, hijau, dan biru) dari citra buatan sendiri.
2. Menentukan nilai ambang batas warna dengan urutan dari terkecil hingga terbesar untuk klasifikasi warna.
3. Menganalisis karakteristik warna dalam citra menggunakan histogram dan melakukan interpretasi data visual tersebut.
4. Menerapkan teknik pengolahan citra untuk memperbaiki foto backlight, terutama meningkatkan kejelasan objek utama (profil wajah).
5. Mendemonstrasikan pemahaman terhadap konversi citra ke grayscale, serta peningkatan kecerahan dan kontras sebagai bagian dari teknik penyempurnaan citra.

1.3 Manfaat Masalah

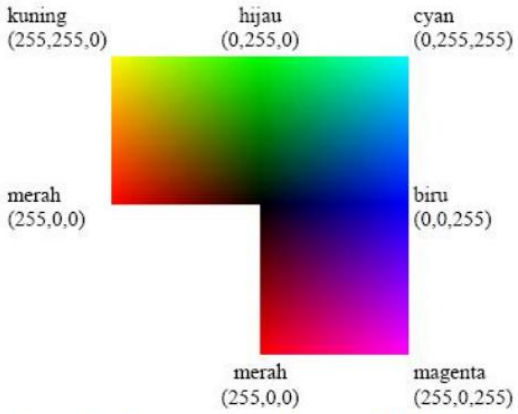
1. Meningkatkan pemahaman praktis mengenai konsep dasar pengolahan citra digital.
2. Mampu mengembangkan solusi berbasis Python untuk analisis warna pada citra.
3. Mengetahui cara menetapkan ambang batas warna untuk klasifikasi visual yang lebih akurat.
4. Mampu memperbaiki kualitas citra dengan kondisi pencahayaan tidak ideal (backlight).
5. Menumbuhkan keterampilan analisis visual melalui interpretasi histogram dan hasil deteksi warna.

BAB II

LANDASAN TEORI

Citra digital dapat didefinisikan sebagai fungsi dua variabel, f(x,y), dimana x dan y adalah koordinat spasial dan nilai f(x,y) yang merupakan intensitas citra pada koordinat tersebut. Teknologi dasar untuk menciptakan dan menampilkan warna pada citra digital berdasarkan pada penelitian bahwa sebuah warna merupakan kombinasi dari tiga warna dasar, yaitu merah, hijau, dan biru (Red, Green, Blue - RGB). RGB adalah suatu model warna yang terdiri dari merah, hijau, dan biru, digabungkan dalam membentuk suatu susunan warna yang luas. Setiap warna dasar, misalnya merah, dapat diberi rentang nilai. Untuk monitor komputer, nilai rentangnya paling kecil = 0 dan paling besar = 255. Pilihan skala 256 ini didasarkan pada cara mengungkap 8 digit bilangan biner yang digunakan oleh mesin komputer. Dengan cara ini, akan diperoleh warna campuran sebanyak 256 x 256 x 256 = 1677726 jenis warna. Sebuah jenis warna, dapat dibayangkan sebagai sebuah vektor di ruang dimensi 3 yang biasanya dipakai dalam matematika, koordinatnya dinyatakan dalam bentuk tiga bilangan, yaitu komponen-x, komponen-y dan komponen-z. Misalkan sebuah vektor dituliskan sebagai r = (x,y,z). Untuk warna, komponen-komponen tersebut digantikan oleh komponen R(ed), G(reen), B(lue). Jadi, sebuah jenis warna dapat dituliskan sebagai berikut: warna = RGB(30, 75, 255). Putih = RGB (255,255,255), sedangkan untuk hitam= RGB(0,0,0).[6]

Bentuk Representasi warna dari sebuah citra digitial dapat dilihat pada Gambar 2.2.



Gambar 2.2 Representasi Warna RGB Pada Citra Digital

Misalnya terdapat gambar berukuran 100 pixel x 100 pixel dengan color encoding 24 bit dengan R = 8 bit, G = 8 bit, B = 8 bit, maka color encoding akan mampu mewakili 0 ... 16.777.215 (mewakili 16 juta warna), dan ruang disk yang dibutuhkan = 100 x 100 x 3 bit (karena RGB) = 30.000 bit = 30 KB atau 100 x 100 x 24 bit = 240.000 bit.[6]

Citra Grayscale

Citra yang ditampilkan dari citra jenis ini terdiri atas warna abu-abu, bervariasi pada warna hitam pada bagian yang intensitas terlemah dan warna putih pada intensitas terkuat. Citra grayscale berbeda dengan citra ”hitam-putih”, dimana pada konteks komputer, citra hitam putih hanya terdiri atas 2 warna saja yaitu ”hitam” dan ”putih” saja. Pada citra grayscale warna bervariasi antara hitam dan

putih, tetapi variasi warna diantaranya sangat banyak. Citra grayscale seringkali merupakan perhitungan dari intensitas cahaya pada setiap pixel pada spektrum elektromagnetik single band.[4]

Citra grayscale disimpan dalam format 8 bit untuk setiap sample pixel, yang memungkinkan sebanyak 256 intensitas. Format ini sangat membantu dalam pemrograman karena manipulasi bit yang tidak terlalu banyak. Untuk mengubah citra berwarna yang mempunyai nilai matrik masing-masing R, G dan B menjadi citra grayscale dengan nilai X, maka konversi dapat dilakukan dengan mengambil rata-rata dari nilai R, G dan B.[5] 2.4 Citra Threshold Citra threshold dilakukan dengan mempertegas citra dengan cara mengubah citra hasil yang memiliki derajat keabuan 255 (8 bit), menjadi hanya dua buah yaitu hitam dan putih. Hal yang perlu diperhatikan pada proses threshold adalah memilih sebuah nilai threshold (T) dimana piksel yang bernilai dibawah nilai

Histogram Citra

• Histogram citra (image histogram) merupakan informasi yang penting mengenai isi citra digital. • Histogram citra adalah grafik yang menggambarkan penyebaran nilai-nilai intensitas pixel dari suatu citra atau bagian tertentu di dalam citra.

1. Dari sebuah histogram dapat diketahui frekuensi kemunculan nisbi (relative) dari intensitas pada citra tersebut.
2. Histogram juga dapat menunjukkan banyak hal tentang kecerahan (brightness) dan kontas (contrast) dari sebuah gambar.
3. Karena itu, histogram adalah alat bantu yang berharga dalam pekerjaan pengolahan citra baik secara kualitatif maupun kuantitatif. Menghitung histogram
4. Misalkan sebuah citra mempunyai L level nilai keabuan, [0, L-1].
5. Hitung frekuensi kemunculan setiap nilai keabuan j dengan cara menghitung jumlah pixel yang mempunyai nilai keabuan tersebut.
6. Perhitungan ini dilakukan untuk j = 0, 1, 2, ..., L – 1. Untuk citra berwarna dengan komponen R, G, dan B, histogram dibuat untuk setiap kanal warna

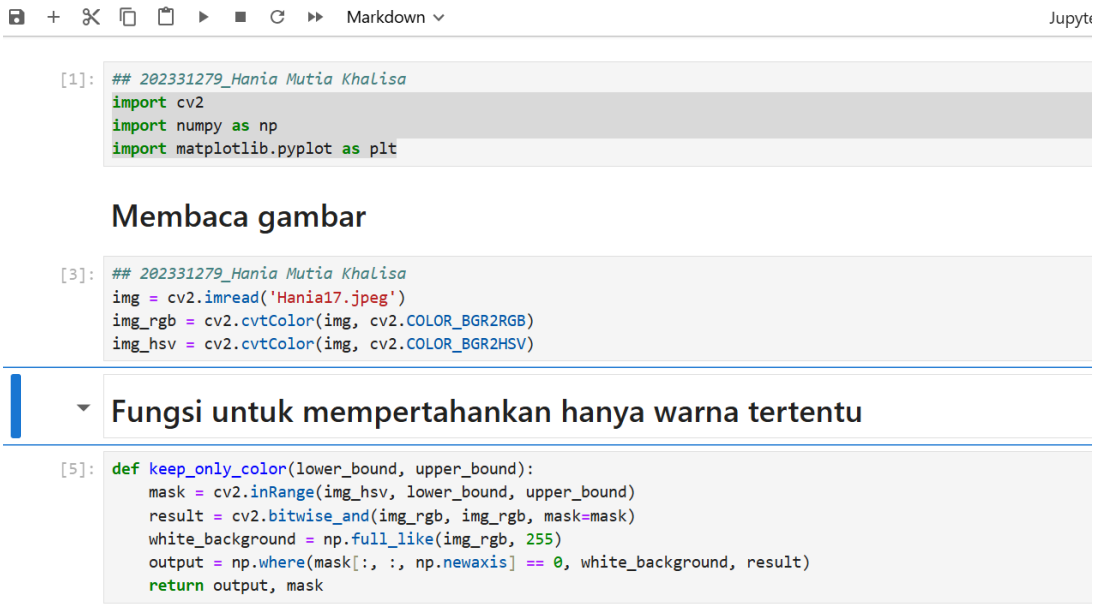
Empat tipe citra berdasarkan kekontrasannya: 1. Citra gelap (under exposed) 2. Citra terang (over exposed) 3. Citra kontras rendah (low contrast) 4. Citra kontras tinggi (high contrast) Kontras menyatakan sebaran terang (lightness) dan gelap (darkness) di dalam sebuah citra.

Segmentasi Segmentasi adalah proses untuk membagi citra menjadi region yang terhubung. Segmentasi citra akan menghasilkan objek-objek dalam citra, atau menghasilkan objek yang sudah dipisah dengan latar belakang. Segmentasi citra dilakukan dengan mempertimbangkan properti yang telah dipilih seperti nilai intensitas dan warna dari pixel. Sebagai contoh, gambar 3 disegmentasi menjadi langit, awan, danau, tanah, dan padang rumput. Gambar 2 disegmentasi berdasarkan warna dan tekstur dari pixel yang terhubung [3]

BAB III

HASIL

1)



```
[1]: ## 202331279_Hania Mutia Khalisa
     import cv2
     import numpy as np
     import matplotlib.pyplot as plt
```

Membaca gambar

```
[3]: ## 202331279_Hania Mutia Khalisa
     img = cv2.imread('Hania17.jpeg')
     img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
     img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

Fungsi untuk mempertahankan hanya warna tertentu

```
[5]: def keep_only_color(lower_bound, upper_bound):
         mask = cv2.inRange(img_hsv, lower_bound, upper_bound)
         result = cv2.bitwise_and(img_rgb, img_rgb, mask=mask)
         white_background = np.full_like(img_rgb, 255)
         output = np.where(mask[:, :, np.newaxis] == 0, white_background, result)
         return output, mask
```

Import library:

- cv2 → OpenCV untuk manipulasi gambar.
- numpy → Operasi array.
- matplotlib.pyplot → Menampilkan citra dan histogram.

Load dan konversi gambar:

- imread membaca gambar.
- cvtColor(...BGR2RGB) → agar gambar tampil dengan warna yang benar di matplotlib.
- cvtColor(...BGR2HSV) → untuk proses deteksi warna berdasarkan *Hue, Saturation, Value*.

Penjelasan fungsi:

- cv2.inRange membuat mask untuk warna tertentu.
- bitwise_and menyaring hanya warna target.
- Bagian lain (tidak termasuk warna) diubah menjadi **putih**.
- Output: hasil gambar dan mask.

Fungsi histogram RGB

```
[7]: def plot_rgb_histogram(image, mask=None, title="Histogram"):
         color = ('r', 'g', 'b')
         plt.figure(figsize=(6, 4))
         for i, col in enumerate(color):
             hist = cv2.calcHist([image], [i], mask, [256], [0, 256])
             plt.plot(hist, color=col)
         plt.title(title)
         plt.xlim([0, 256])
         plt.xlabel("Intensitas")
         plt.ylabel("Frekuensi")
         plt.grid()
         plt.show()
```

Rentang HSV untuk warna

```
[9]: lower_blue = np.array([100, 150, 0])
     upper_blue = np.array([140, 255, 255])

     lower_green = np.array([40, 40, 40])
     upper_green = np.array([80, 255, 255])

     lower_red1 = np.array([0, 50, 50])
     upper_red1 = np.array([10, 255, 255])
     lower_red2 = np.array([160, 50, 50])
     upper_red2 = np.array([180, 255, 255])
```

Proses deteksi warna

```
[11]: blue_output, blue_mask = keep_only_color(lower_blue, upper_blue)
      green_output, green_mask = keep_only_color(lower_green, upper_green)

      mask_red1 = cv2.inRange(img_hsv, lower_red1, upper_red1)
      mask_red2 = cv2.inRange(img_hsv, lower_red2, upper_red2)
      mask_red = cv2.bitwise_or(mask_red1, mask_red2)
      red_result = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_red)
```

1. Mengatur batas bawah dan atas warna dalam format HSV.
2. Menjalankan fungsi deteksi untuk warna biru dan hijau
3. Gabungan 2 mask untuk warna merah karena merah berada di dua rentang hue.

jupyter no 1 Last Checkpoint: 2 days ago

File Edit View Run Kernel Settings Help

```
red_result = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_red)
white_background = np.full_like(img_rgb, 255)
red_output = np.where(mask_red[:, :, np.newaxis] == 0, white_background, red_result)
```

Tampilkan hasil citra

```
[13]: fig, axs = plt.subplots(2, 2, figsize=(12, 10))

      axs[0, 0].imshow(img_rgb)
      axs[0, 0].set_title("Citra Asli")
      axs[0, 0].axis('off')

      axs[0, 1].imshow(blue_output)
      axs[0, 1].set_title("Deteksi Biru")
      axs[0, 1].axis('off')

      axs[1, 0].imshow(red_output)
      axs[1, 0].set_title("Deteksi Merah")
      axs[1, 0].axis('off')

      axs[1, 1].imshow(green_output)
      axs[1, 1].set_title("Deteksi Hijau")
      axs[1, 1].axis('off')
```

Setiap subplot menampilkan hasil:

- Gambar asli
- Segmentasi biru
- Segmentasi merah
- Segmentasi hijau

Citra Asli

- **Deskripsi:** Citra asli merupakan hasil pembacaan file Hania mutia k.jpeg, dikonversi ke format RGB dari BGR (standar OpenCV).
- **Tujuan:** Sebagai acuan dasar sebelum dilakukan segmentasi berdasarkan warna.

Histogram Citra Asli:

- Ketiga kurva warna (merah, hijau, biru) biasanya tampak menyebar sesuai dengan keberadaan warna dominan dalam gambar.
- Jika gambar cenderung cerah, histogram akan banyak di sisi kanan (intensitas tinggi).
- Jika gambar gelap, histogram banyak di sisi kiri.

2. Deteksi Warna Biru

- **Metode:** Segmentasi dilakukan menggunakan HSV, dengan rentang [100, 150, 0] hingga [140, 255, 255].
- **Output Citra:** Bagian berwarna biru tetap terlihat, sedangkan warna lain dijadikan putih.
- **Tujuan:** Memisahkan objek/bagian yang memiliki warna biru dari citra asli.

Histogram Deteksi Biru:

- Kurva biru mendominasi, karena hanya warna biru yang dipertahankan.
- Kurva merah dan hijau sangat kecil atau mendekati nol.
- Distribusi histogram membantu memastikan apakah segmentasi biru berhasil atau tidak.

3. Deteksi Warna Merah

- **Metode:** Karena merah berada di dua rentang di HSV, digunakan 2 mask: [0, 50, 50] ke [10, 255, 255] dan [160, 50, 50] ke [180, 255, 255].
- **Output Citra:** Hanya bagian merah pada gambar yang dipertahankan.
- **Tujuan:** Menyaring bagian yang mengandung warna merah (misal baju, latar, dll).

Histogram Deteksi Merah:

- Kurva merah terlihat jelas dan dominan.
- Kurva hijau dan biru kecil atau hampir nol.
- Jika histogram merah menyebar ke intensitas tinggi, berarti objek merah memiliki pencahayaan baik.

4. Deteksi Warna Hijau

- **Metode:** Segmentasi HSV pada rentang [40, 40, 40] sampai [80, 255, 255].
- **Output Citra:** Hanya warna hijau yang dipertahankan, sisanya putih.

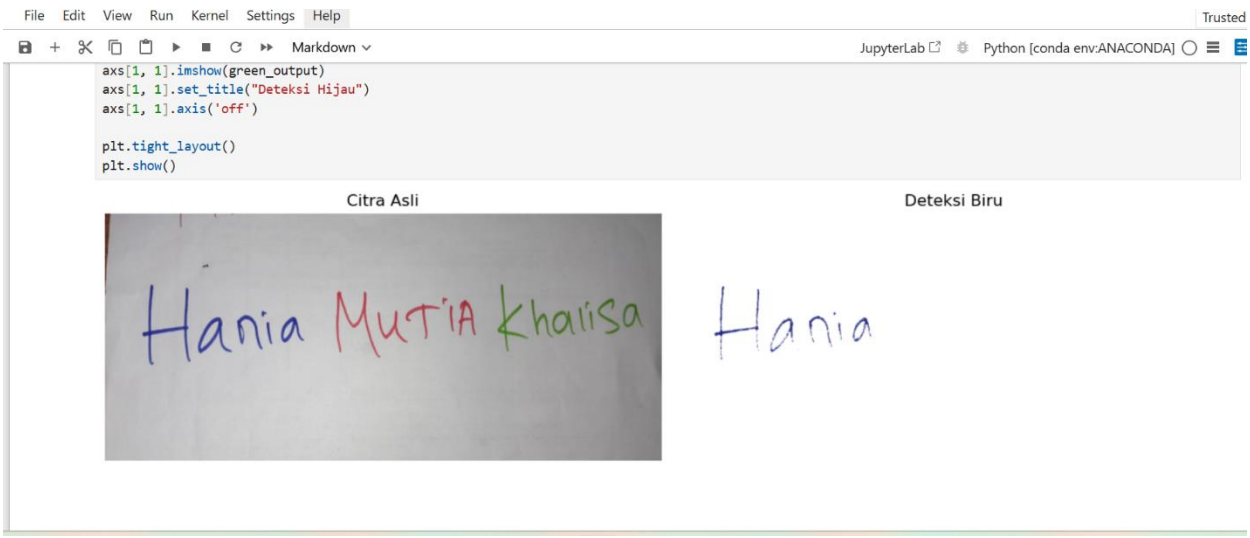
- **Tujuan:** Menyaring komponen yang mengandung warna hijau, seperti tanaman atau pakaian hijau.

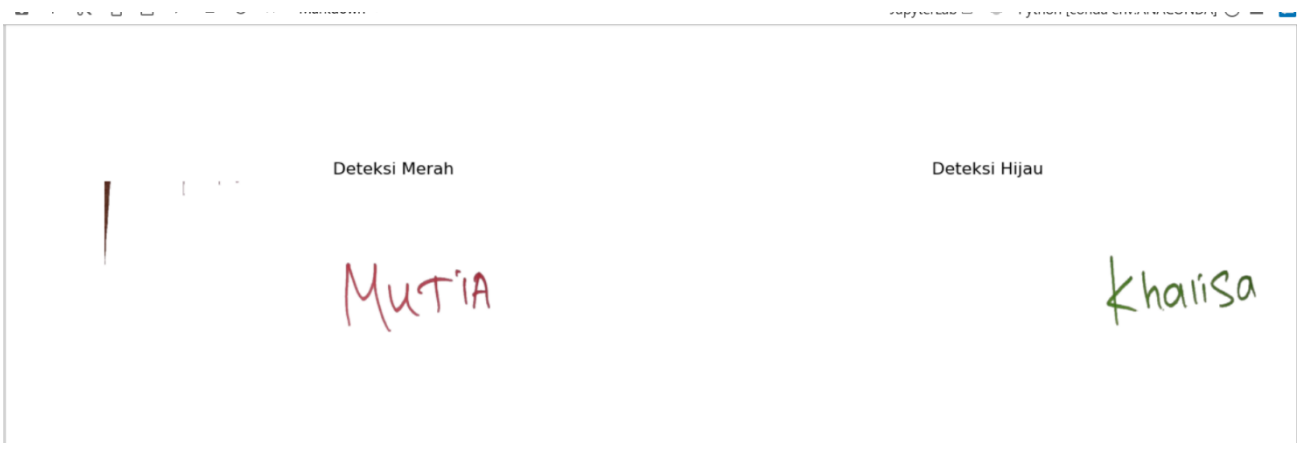
Histogram Deteksi Hijau:

- Kurva hijau paling tinggi, menandakan dominasi hijau di area tersegmentasi.
- Warna merah dan biru hampir tidak tampak.
- Histogram bisa digunakan untuk mengetahui seberapa banyak intensitas hijau di gambar.

Kesimpulan:

- Segmentasi warna menggunakan ruang warna **HSV** lebih efektif dibandingkan langsung RGB karena HSV memisahkan komponen warna dan pencahayaan.
- **Histogram RGB** berguna untuk:
  - Mengevaluasi hasil segmentasi
  - Melihat dominasi dan distribusi warna
  - Menganalisis pencahayaan dan kontras warna

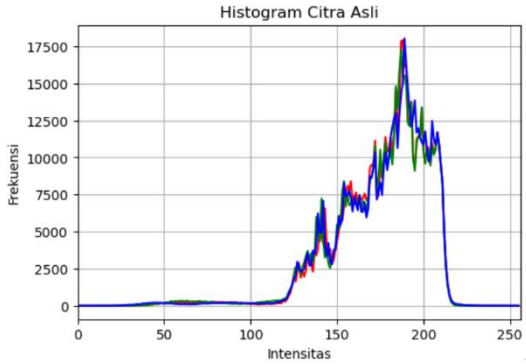


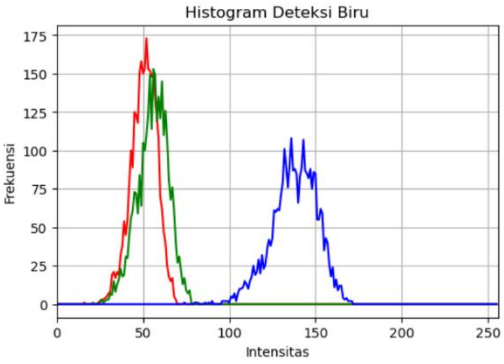


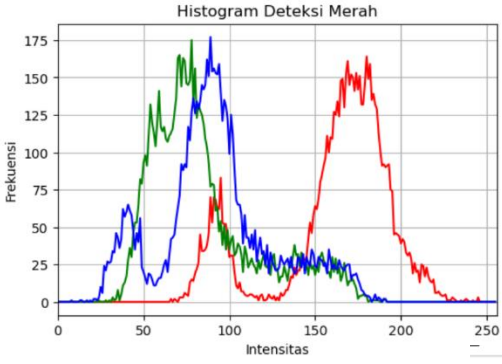
Markdown

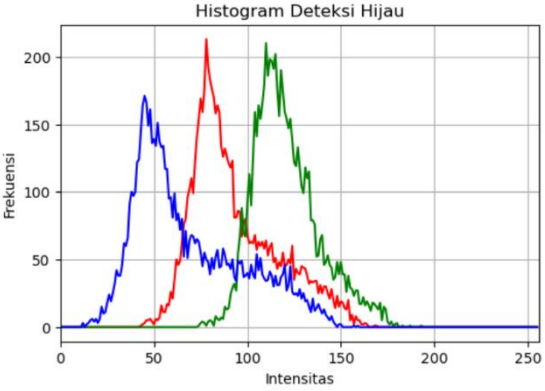
Tampilkan histogram untuk masing-masing hasil

[15]:
```
plot_rgb_histogram(img_rgb, None, "Histogram Citra Asli")
plot_rgb_histogram(blue_output, blue_mask, "Histogram Deteksi Biru")
plot_rgb_histogram(red_output, mask_red, "Histogram Deteksi Merah")
plot_rgb_histogram(green_output, green_mask, "Histogram Deteksi Hijau")
```

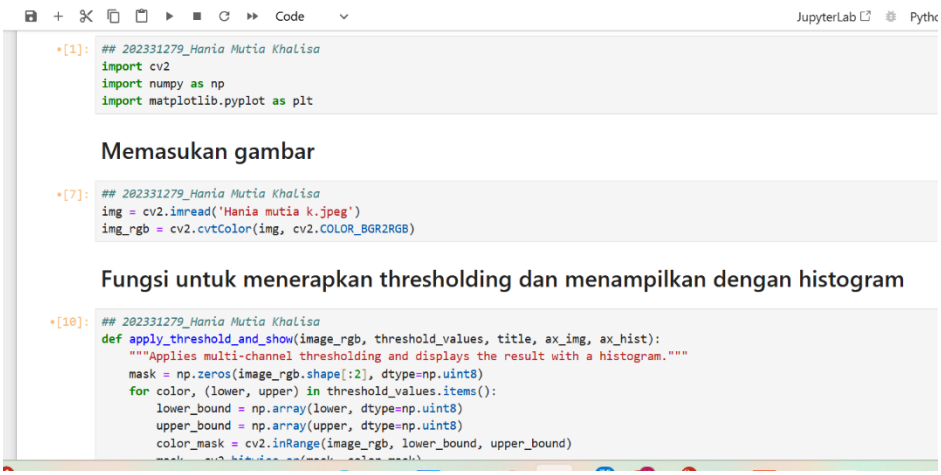








2)



```
[1]: ## 202331279_Hania Mutia Khalisa
     import cv2
     import numpy as np
     import matplotlib.pyplot as plt

Memasukan gambar

[7]: ## 202331279_Hania Mutia Khalisa
     img = cv2.imread('Hania mutia k.jpeg')
     img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

Fungsi untuk menerapkan thresholding dan menampilkan dengan histogram

[10]: ## 202331279_Hania Mutia Khalisa
      def apply_threshold_and_show(image_rgb, threshold_values, title, ax_img, ax_hist):
          """Applies multi-channel thresholding and displays the result with a histogram."""
          mask = np.zeros(image_rgb.shape[:2], dtype=np.uint8)
          for color, (lower, upper) in threshold_values.items():
              lower_bound = np.array(lower, dtype=np.uint8)
              upper_bound = np.array(upper, dtype=np.uint8)
              color_mask = cv2.inRange(image_rgb, lower_bound, upper_bound)
```

1. Memasukkan Gambar

img = cv2.imread('Hania mutia k.jpeg')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

Output:

- Gambar 'Hania mutia k.jpeg' dibaca ke dalam variabel img menggunakan OpenCV.
- Warna gambar dikonversi dari format BGR ke RGB untuk ditampilkan dengan benar menggunakan matplotlib.

2. Fungsi apply_threshold_and_show()

Fungsi ini memiliki dua bagian output:

a. Hasil Thresholding

thresholded_img = cv2.bitwise_and(image_rgb, image_rgb, mask=mask)
ax_img.imshow(thresholded_img)

```
Code
        color_mask = cv2.inRange(image_rgb, lower_bound, upper_bound)
        mask = cv2.bitwise_or(mask, color_mask)

    thresholded_img = cv2.bitwise_and(image_rgb, image_rgb, mask=mask)
    ax_img.imshow(thresholded_img)
    ax_img.set_title(title)
    ax_img.axis('off')

    color = ('r', 'g', 'b')
    for i, col in enumerate(color):
        histr = cv2.calcHist([thresholded_img], [i], mask, [256], [0, 256])
        ax_hist.plot(histr, color=col)
        ax_hist.set_xlim([0, 256])
    ax_hist.set_title(f'HISTOGRAM {title}')
```

Output:

- Menampilkan gambar hasil thresholding dengan kombinasi warna tertentu, misalnya hanya bagian-bagian yang berwarna biru, merah, atau kombinasi lainnya sesuai parameter threshold_values.

- Gambar yang ditampilkan akan menunjukkan bagian yang lolos dari filter warna yang ditentukan.

b. Histogram

histr = cv2.calcHist([thresholded_img], [i], mask, [256], [0, 256])
ax_hist.plot(histr, color=col)

Output:

- Menampilkan histogram distribusi intensitas dari masing-masing kanal warna (R, G, B) pada hasil gambar thresholded.
- Membantu melihat sebaran warna dalam gambar yang tersaring.

3. Menentukan Nilai Threshold

```
thresholds = {
   "NONE": {},
   "BLUE": {"BLUE": ([0, 0, 100], [100, 100, 255])},
   ...
}
```

Output:

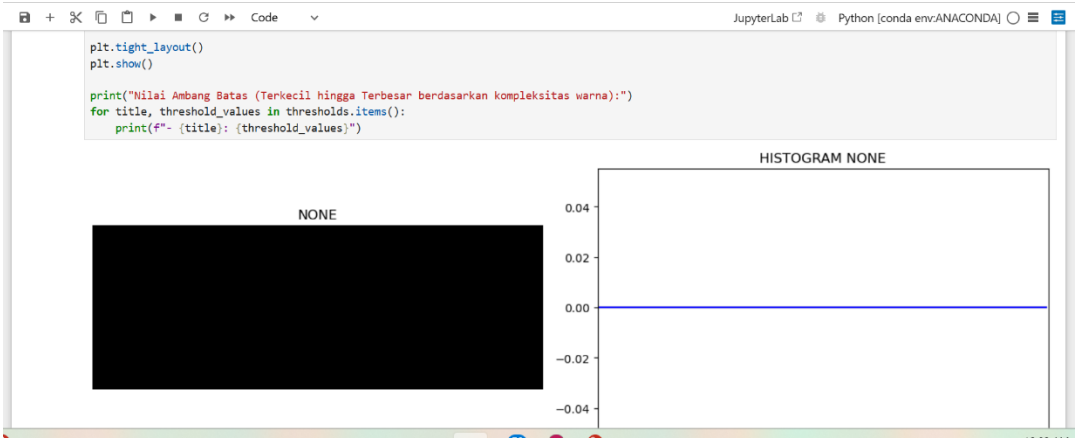
- Dictionary ini menyimpan batas nilai threshold untuk filter warna. Misalnya:
  - "BLUE": menampilkan hanya bagian yang warnanya dalam rentang biru tertentu.
  - "RED-GREEN-BLUE": gabungan 3 filter warna sekaligus.

4. Menampilkan Output dalam Subplots

fig, axes = plt.subplots(4, 2, figsize=(12, 16))
...
apply_threshold_and_show(...)

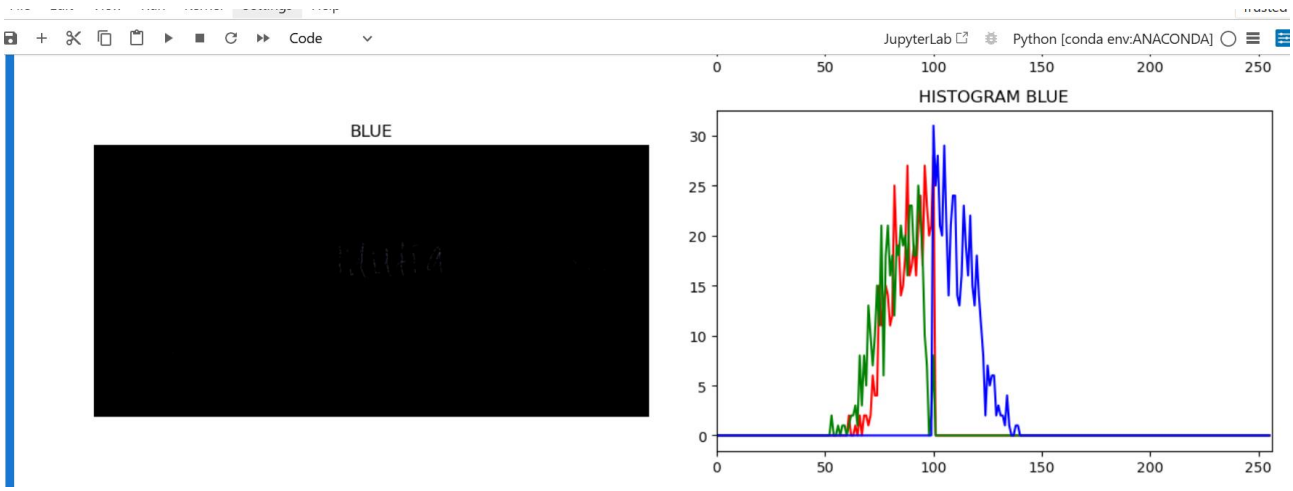
Output:

- Menampilkan 4 hasil gambar threshold dan 4 histogramnya dalam satu figure menggunakan matplotlib.
- Total subplot: 8 (4 gambar + 4 histogram).
- Memudahkan analisis visualisasi warna dari hasil thresholding.



1. Output: NONE

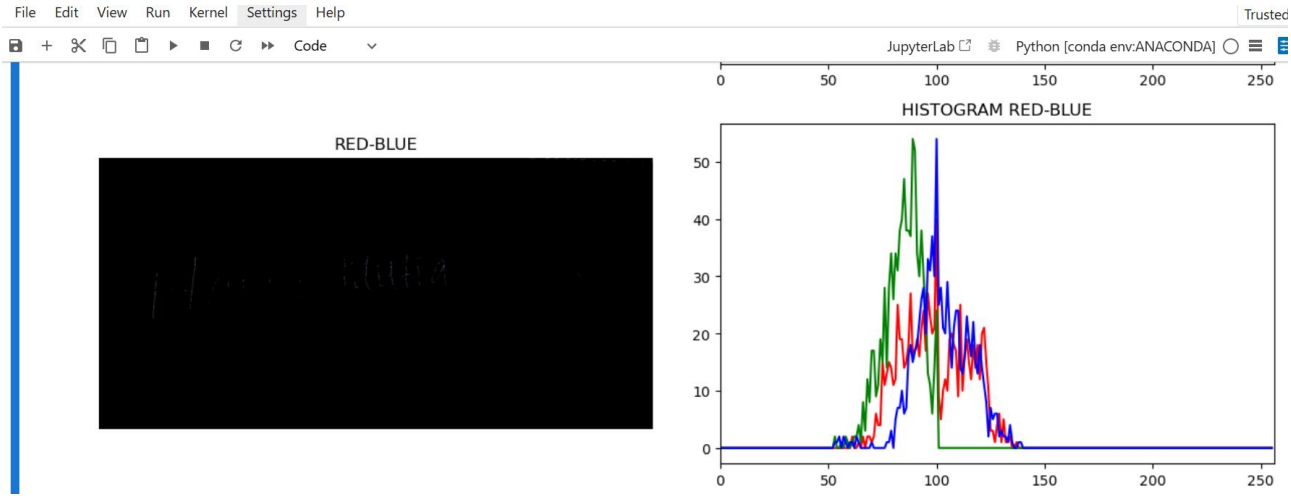
- **Citra**: Tampilan gambar sepenuhnya hitam.
- **Histogram**: Garis lurus datar di 0 untuk semua intensitas.
- **Penjelasan**: Ini menandakan bahwa tidak ada kanal warna yang dipilih untuk diproses. Artinya, nilai-nilai warna dari gambar asli tidak diolah sehingga tidak ada data warna yang bisa divisualisasikan dalam histogram. Histogram datar mengindikasikan distribusi warna nol (kosong).



2. Output: BLUE

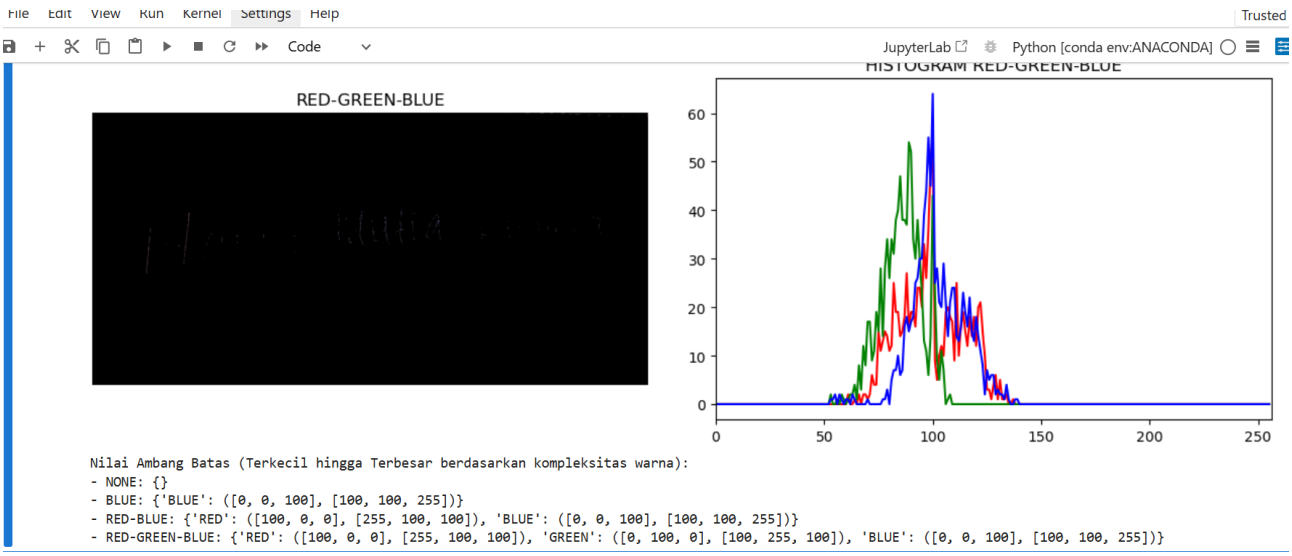
- **Citra**: Tampak masih dominan hitam, tetapi terdapat pola samar seperti tulisan (karena kanal biru).
- **Histogram**: Terlihat kurva berwarna biru, merah, dan hijau yang menunjukkan distribusi intensitas warna.

- **Penjelasan**: Kanal biru dari gambar digunakan untuk membentuk gambar baru. Warna biru pada histogram menunjukkan distribusi yang lebih tinggi dibandingkan merah dan hijau. Artinya, bagian-bagian terang pada gambar dominan berada di kanal biru.



3. Output: RED-BLUE

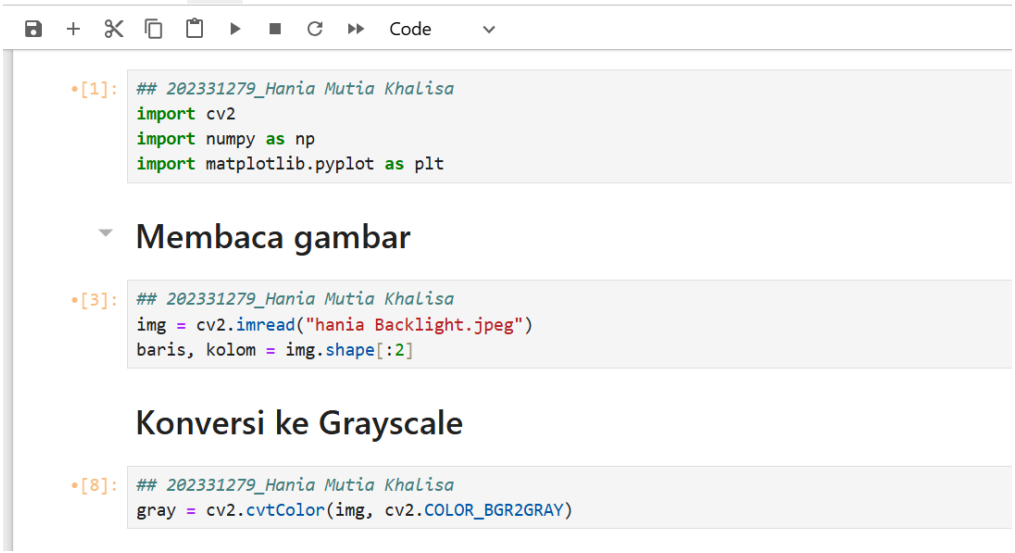
- **Citra**: Tulisan seperti "Hania Mutia" mulai terlihat lebih jelas.
- **Histogram**: Distribusi intensitas lebih tajam, dengan puncak tinggi pada intensitas sekitar 100-an untuk semua kanal (R, G, B).
- **Penjelasan**: Operasi ini menggabungkan dua kanal warna yaitu merah dan biru. Dengan kombinasi ini, detail pada gambar semakin muncul karena informasi warna lebih kompleks. Histogram menunjukkan penyebaran data yang padat pada nilai intensitas rendah-menengah, mengindikasikan kontras yang lebih tinggi.

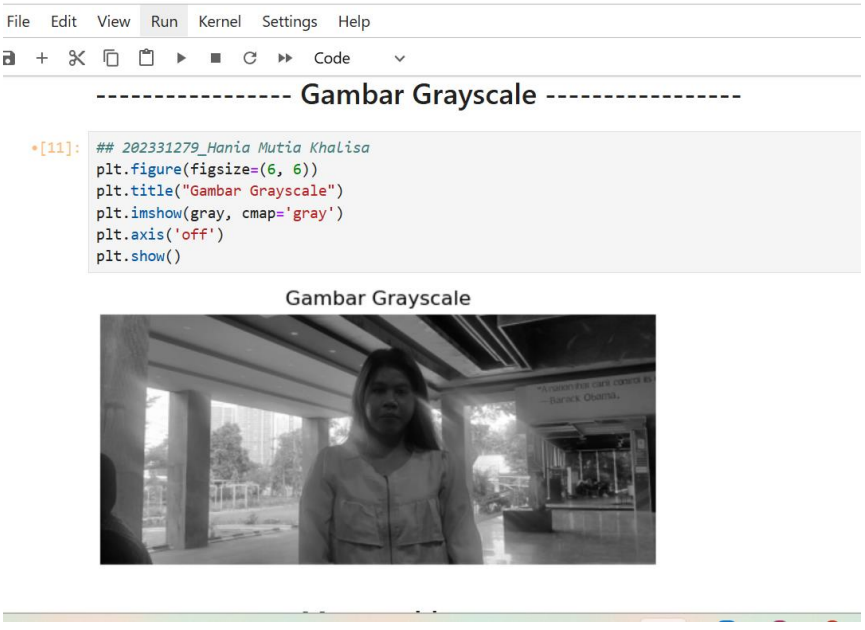


- **Citra**: Tulisan seperti *"Hania Mutia Khalisa"* mulai terlihat lebih jelas meskipun latar masih dominan gelap.
- **Histogram**: Distribusi intensitas lebih tajam dan meruncing, dengan puncak tinggi di sekitar nilai intensitas 100-an untuk ketiga kanal (R, G, B), menunjukkan konsentrasi warna pada rentang sempit.

- **Penjelasan**: Operasi ini menggunakan seluruh kanal warna (merah, hijau, dan biru), menghasilkan tampilan warna asli dari citra. Karena semua channel aktif, detail dan kontur mulai tampak lebih kompleks. Namun, karena intensitasnya cenderung rendah, citra tampak gelap. Histogram memperlihatkan bahwa mayoritas piksel terkonsentrasi pada nilai intensitas rendah-menengah, mencerminkan pencahayaan yang minim namun dengan kontras yang relatif stabil.

3)





1. Membaca dan Menampilkan Gambar Grayscale

- **Proses:**
  - Gambar berwarna dibaca dengan cv2.imread().
  - Kemudian dikonversi menjadi grayscale menggunakan cv2.cvtColor() dengan parameter cv2.COLOR_BGR2GRAY.
- **Output:**
  - Gambar grayscale ditampilkan menggunakan matplotlib.pyplot.
- **Penjelasan:**
  - Grayscale merupakan citra dengan satu kanal warna (0-255).

- Output menampilkan gambar dengan tone hitam-putih.
- Proses ini penting sebagai tahap awal pengolahan citra karena mengurangi kompleksitas dari 3 channel ke 1 channel.

Code

---------- Mencerahkan ----------

[14]:
```
## 202331279_Hania Mutia Khalisa
beta = 40
citra_cerah = np.clip(gray + beta, 0, 255).astype(np.uint8)

plt.figure(figsize=(6, 6))
plt.title("Gambar Gray yang Dipercerah")
plt.imshow(citra_cerah, cmap='gray')
plt.axis('off')
plt.show()
```

2. Mencerahkan Gambar

- **Proses:**
  - Ditambahkan nilai beta = 40 pada tiap piksel grayscale.
  - Fungsi np.clip() digunakan agar hasil piksel tetap dalam rentang valid (0-255).
- **Output:**
  - Gambar grayscale yang lebih cerah dibandingkan sebelumnya.
- **Penjelasan:**
  - Penambahan nilai ke setiap piksel menaikkan intensitas cahaya.
  - Hasil citra tampak lebih terang, khususnya pada bagian wajah dan latar belakang yang sebelumnya gelap.

```
plt.show()
```

Gambar Pertama: "Gambar Gray yang Dipercerah"

Penjelasan:

- Gambar ini menunjukkan hasil proses *pencerahan* (*brightness enhancement*) pada citra grayscale.
- Proses ini biasanya dilakukan dengan menambahkan atau mengalikan nilai piksel dengan suatu konstanta agar tampak lebih terang.
- Hasilnya adalah gambar grayscale dengan intensitas cahaya yang lebih tinggi, namun bisa menyebabkan sebagian area menjadi terlalu terang (*overexposed*) atau bahkan kehilangan detail, seperti yang terlihat di beberapa bagian latar belakang.

Gambar Kedua: Kode untuk "Meningkatkan Kontras"

---------- Meningkatkan Kontras ----------

```
[17]: ## 202331279_Hania Mutia Khalisa
alpha = 1.5
citra_kontras = np.clip(alpha * gray, 0, 255).astype(np.uint8)

plt.figure(figsize=(6, 6))
plt.title("Gambar Gray yang Diperkontras")
plt.imshow(citra_kontras, cmap='gray')
plt.axis('off')
plt.show()
```

Penjelasan Kode:

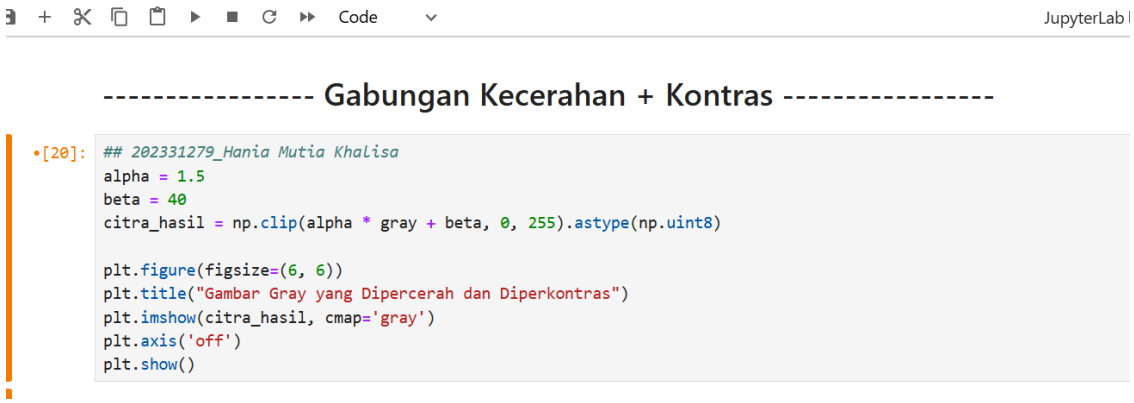
alpha = 1.5
citra_kontras = np.clip(alpha * gray, 0, 255).astype(np.uint8)

- alpha = 1.5 artinya kontras ditingkatkan sebesar 50%.
- np.clip(..., 0, 255) menjaga agar nilai piksel tetap dalam rentang valid [0, 255].
- astype(np.uint8) mengonversi hasil ke format citra standar.

Gambar Ketiga: "Gambar Gray yang Dikontras"

Penjelasan:

- Ini adalah hasil dari penerapan peningkatan kontras menggunakan kode sebelumnya.
- Gambar terlihat lebih tajam dengan perbedaan terang dan gelap yang lebih jelas.
- Detail pada wajah dan latar belakang menjadi lebih menonjol dibandingkan gambar asli.



Gambar Gray yang Dipercerahkan dan Dikontras

alpha = 1.5 # untuk meningkatkan kontras
beta = 40 # untuk meningkatkan kecerahan
citra_hasil = np.clip(alpha * gray + beta, 0, 255).astype(np.uint8)

Penjelasan Proses:

- alpha = 1.5: Menambah **kontras**, yaitu membuat perbedaan antara area terang dan gelap menjadi lebih besar.
- beta = 40: Menambah **kecerahan**, yaitu menaikkan semua nilai piksel agar gambar tampak lebih terang.
- np.clip(..., 0, 255): Mencegah nilai piksel melebihi batas valid untuk citra 8-bit (0–255).

Hasil Output:

- Gambar menjadi **lebih terang dan lebih tajam** dibandingkan gambar sebelumnya.
- **Detail wajah dan objek latar** terlihat lebih jelas.
- Namun, beberapa bagian (seperti latar belakang di sisi kiri) menjadi terlalu terang (**overexposed**) hingga kehilangan detail.

BAB IV

PENUTUP

1. **Citra Digital** adalah representasi dua dimensi dari objek nyata dalam bentuk matriks pixel, dengan nilai intensitas warna pada tiap titik koordinat (x,y).
2. **Model Warna RGB** memungkinkan representasi hingga 16 juta warna (256^3), karena tiap kanal (R, G, B) memiliki rentang 0–255 (8-bit).
3. **Grayscale** adalah bentuk citra satu kanal yang hanya menyimpan intensitas (0 = hitam, 255 = putih), ideal untuk mengurangi kompleksitas data dalam analisis.
4. **Thresholding** digunakan untuk mengubah citra grayscale menjadi citra biner (hitam-putih) berdasarkan ambang nilai tertentu, penting untuk ekstraksi objek.
5. **Histogram Citra** menyajikan distribusi intensitas pixel, yang berguna untuk menganalisis kecerahan, kontras, dan dominasi warna dalam gambar.
6. **Segmentasi** bertujuan memisahkan objek atau wilayah tertentu dalam gambar berdasarkan karakteristik seperti warna, tekstur, dan intensitas.

Kesimpulan Hasil Praktikum

1. **Segmentasi Warna (HSV)**:
   - Deteksi warna biru, merah, dan hijau berhasil dilakukan menggunakan pemisahan kanal HSV.
   - Segmentasi warna biru dan merah menunjukkan keberhasilan dalam menyaring bagian warna spesifik dengan bantuan histogram masing-masing warna.
   - HSV terbukti lebih efektif dibanding RGB untuk analisis warna karena memisahkan informasi warna (Hue) dan pencahayaan (Value).
2. **Thresholding dan Visualisasi Histogram**:
   - Fungsi apply_threshold_and_show() berhasil menampilkan citra hasil thresholding dan histogram warna yang menyertainya.
   - Hasil threshold BLUE dan RED-BLUE menunjukkan bahwa gambar tertentu memiliki dominasi warna spesifik yang terlihat jelas pada histogram.
3. **Konversi Grayscale dan Enhancements**:
   - Proses konversi ke grayscale mengurangi data menjadi 1 channel, mempermudah proses pemrosesan lanjutan.
   - Penambahan brightness (beta = 40) membuat citra lebih terang.
   - Peningkatan kontras (alpha = 1.5) memberikan efek gambar lebih tajam dan detail lebih jelas.
   - Kombinasi kontras dan kecerahan memberikan hasil paling optimal untuk memperjelas area penting dalam citra.

DAFTAR PUSTAKA

1. https://media.neliti.com/media/publications/214469-jurnal-pembelajaran.pdf
2. https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2019-2020/07-Image-Histogram.pdf
3. https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2019-2020/Makalah2019/13516133.pdf