UTS

PENGOLAHAN CITRA

NAMA : Muhammad Rizki Insani

NIM : 2023331319

KELAS : A

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

NO.PC : 28

ASISTEN : 1. Clarenca Sweetdiva Pereira

2. Viana Salsabila Fairuz Syahla

3. Kashrina Masyid Azka

4. Sasikirana Ramadhanty Setiawan Putri

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2024/2025

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

Masalah yang dihadapi dalam praktikum ini adalah bagaimana mendeteksi teks berwarna dalam gambar serta cara meningkatkan kualitas citra yang memiliki pencahayaan yang kurang (backlight). Selain itu, bagaimana kita bisa memisahkan dan menonjolkan warna tertentu (merah, hijau, biru) dalam gambar, serta meningkatkan kecerahan dan kontras pada citra grayscale untuk mendukung analisis visual yang lebih efektif.

Tujuan Masalah

Tujuan praktikum ini adalah untuk:

1. Mendeteksi dan menonjolkan warna-warna tertentu dalam gambar (merah, hijau, biru) dengan memanfaatkan model warna HSV.
2. Meningkatkan kualitas gambar yang kurang pencahayaan dengan mengonversinya menjadi grayscale dan kemudian memperbaiki kecerahan serta kontras gambar tersebut.
3. Menampilkan hasil deteksi warna dan peningkatan citra secara jelas melalui visualisasi serta histogram.

1.2 Manfaat Masalah

Manfaat yang diharapkan dari praktikum ini adalah:

1. **Penerapan Teknik Pengolahan Citra**: Penggunaan teknik seperti deteksi warna, peningkatan kecerahan, dan kontras memberi wawasan lebih mendalam tentang pengolahan citra untuk aplikasi praktis, seperti analisis gambar dan pemrosesan data gambar dari berbagai sumber.
2. **Peningkatan Kualitas Gambar**: Melalui konversi ke grayscale dan peningkatan kecerahan serta kontras, kita dapat menghasilkan gambar yang lebih jelas, khususnya pada gambar dengan pencahayaan yang buruk.
3. **Pemahaman Model Warna HSV**: Praktikum ini memperkenalkan cara kerja model warna HSV untuk deteksi warna yang lebih tepat dan efisien dibandingkan dengan model warna RGB.

BAB II

LANDASAN TEORI

Pengolahan Citra Digital (Digital Image Processing)

Pengolahan citra digital adalah suatu disiplin ilmu yang berkaitan dengan manipulasi dan analisis gambar untuk mendapatkan informasi yang berguna. Dalam pengolahan citra, gambar yang berasal dari dunia nyata atau sensor digital diproses untuk meningkatkan kualitas, mengekstraksi informasi penting, atau mengubahnya ke dalam format yang lebih mudah dipahami oleh mesin atau manusia. Salah satu metode yang banyak digunakan dalam pengolahan citra adalah **konversi gambar berwarna menjadi grayscale**. Proses ini mengurangi jumlah komponen warna yang perlu diproses, menjadikannya lebih efisien dalam aplikasi yang membutuhkan analisis intensitas cahaya.

Grayscale sendiri mengacu pada gambar yang hanya terdiri dari berbagai tingkatan abu-abu, dari hitam (intensitas rendah) hingga putih (intensitas tinggi). Konversi ini sangat berguna dalam pengolahan gambar yang memiliki pencahayaan rendah atau di bawah kondisi pencahayaan yang buruk, karena memungkinkan pengolahan lebih lanjut untuk mendeteksi objek atau fitur dalam citra tanpa gangguan warna. Pengolahan citra grayscale juga penting dalam aplikasi seperti **deteksi objek** dan **analisis tekstur**, yang sangat bergantung pada perbedaan intensitas cahaya, bukan pada warna.

Salah satu teknik penting dalam pengolahan citra adalah **peningkatan kecerahan** dan **kontras**. Peningkatan kecerahan dilakukan dengan cara meningkatkan nilai intensitas piksel untuk membuat gambar lebih terang, sementara peningkatan kontras bertujuan memperjelas perbedaan antara area terang dan gelap dalam gambar. Kedua teknik ini sangat berguna untuk gambar yang memiliki pencahayaan kurang, di mana banyak detail atau objek dapat hilang karena kurangnya kontras visual.

Model Warna HSV (Hue, Saturation, Value)

Dalam pengolahan citra, penggunaan model warna yang tepat sangat mempengaruhi hasil deteksi dan segmentasi warna. Salah satu model yang banyak digunakan adalah **HSV (Hue, Saturation, Value)**. Model ini lebih intuitif dan mudah digunakan daripada model warna tradisional seperti **RGB (Red, Green, Blue)**, terutama dalam aplikasi yang memerlukan segmentasi warna.

Komponen utama dari model HSV adalah:

- **Hue (H)**: Merupakan parameter yang menunjukkan jenis warna dalam spektrum warna. Hue menggambarkan warna dasar (merah, biru, hijau, kuning, dll.) yang dapat diukur dalam satuan derajat pada lingkaran warna.
- **Saturation (S)**: Menunjukkan sejauh mana warna tersebut jenuh atau murni. Semakin tinggi nilai saturation, semakin jernih dan murni warna tersebut, sedangkan nilai yang lebih rendah menunjukkan warna yang lebih pudar atau lebih dekat ke abu-abu.
- **Value (V)**: Merupakan parameter yang menggambarkan kecerahan atau intensitas warna. Nilai tinggi pada komponen ini menunjukkan warna yang lebih terang, sedangkan nilai rendah menunjukkan warna yang lebih gelap.

Keunggulan utama model HSV adalah kemampuannya untuk lebih mudah memisahkan informasi warna dari komponen kecerahan gambar. Oleh karena itu, **deteksi warna** dalam pengolahan citra sering menggunakan model ini karena memberikan kontrol yang lebih baik terhadap aspek warna dibandingkan dengan model RGB. Misalnya, deteksi warna merah, hijau, dan biru akan lebih efektif dengan menggunakan model HSV karena dapat memisahkan informasi hue dan saturasi dari faktor kecerahan gambar.

Peningkatan Kecerahan dan Kontras (Brightness and Contrast Enhancement)

Peningkatan kecerahan dan kontras adalah langkah-langkah penting dalam pengolahan citra, terutama untuk gambar yang memiliki kualitas rendah akibat pencahayaan yang kurang. **Peningkatan kecerahan** bertujuan untuk membuat gambar lebih terang dengan cara menambahkan nilai intensitas pada setiap piksel, sehingga area gelap pada gambar menjadi lebih terang. Hal ini dapat dilakukan dengan mengubah nilai **parameter beta** dalam fungsi peningkatan citra. Sementara itu, **peningkatan kontras** bertujuan untuk memperjelas perbedaan antara bagian gelap dan terang dalam gambar, yang akan memudahkan dalam mendeteksi objek atau fitur yang tersembunyi dalam gambar.

Salah satu metode yang umum digunakan dalam peningkatan kecerahan dan kontras adalah **transformasi linier**, di mana nilai intensitas setiap piksel pada gambar dimodifikasi dengan rumus matematis tertentu, seperti new_pixel=α×pixel_value+βnew_pixel = \alpha \times pixel_value + \betanew_pixel=α×pixel_value+β. Di sini, parameter **alpha** mengatur penguatan kontras, sementara **beta** menambahkan kecerahan tambahan pada citra. Metode ini sangat efektif dalam mengoptimalkan gambar yang kurang cahaya, terutama pada citra medis atau citra keamanan yang memerlukan visibilitas lebih jelas.

Pengolahan kecerahan dan kontras juga digunakan dalam **pengenalan wajah** dan **sistem pemantauan visual** yang bergantung pada pencahayaan yang baik untuk mendeteksi objek dan mengenali wajah manusia.

Thresholding dalam Pengolahan Citra (Image Thresholding)

Thresholding adalah salah satu teknik dasar dalam pengolahan citra yang digunakan untuk memisahkan objek dari latar belakangnya dengan menetapkan ambang batas tertentu untuk intensitas piksel. Teknik ini banyak digunakan dalam aplikasi seperti **deteksi teks**, **segmentasi citra**, dan **pengenalan objek**. **Thresholding biner** adalah bentuk sederhana di mana piksel dengan nilai intensitas lebih tinggi dari nilai ambang ditandai dengan warna putih, dan yang lainnya akan diberi warna hitam.

Selain thresholding biner, ada juga teknik **thresholding adaptif**, di mana nilai ambang batas dapat disesuaikan berdasarkan nilai intensitas lokal piksel dalam gambar. Metode ini sangat berguna ketika citra memiliki pencahayaan yang tidak merata, yang sering terjadi pada gambar yang diambil dalam kondisi pencahayaan alami.

Teknik Morfologi dalam Pengolahan Citra

Teknik morfologi adalah bagian dari pengolahan citra yang berfokus pada bentuk dan struktur objek dalam citra biner. Operasi dasar dalam morfologi adalah **dilasi** dan **erosion**, yang digunakan untuk memperbaiki hasil deteksi objek.

- **Dilasi**: Merupakan proses yang memperbesar objek dalam citra biner. Teknik ini digunakan untuk memperjelas objek kecil atau memperbaiki fitur yang hilang dalam hasil deteksi.

- **Erosi**: Kebalikan dari dilasi, di mana objek dalam citra biner akan diperkecil. Teknik ini berguna untuk menghilangkan noise kecil atau mengurangi ukuran objek.

Morfologi digunakan dalam aplikasi seperti **deteksi teks** dan **penghapusan noise**, serta untuk memperbaiki kualitas citra biner yang dihasilkan oleh proses thresholding.

BAB III
HASIL

DESKRIPSI PRAKTIKUM

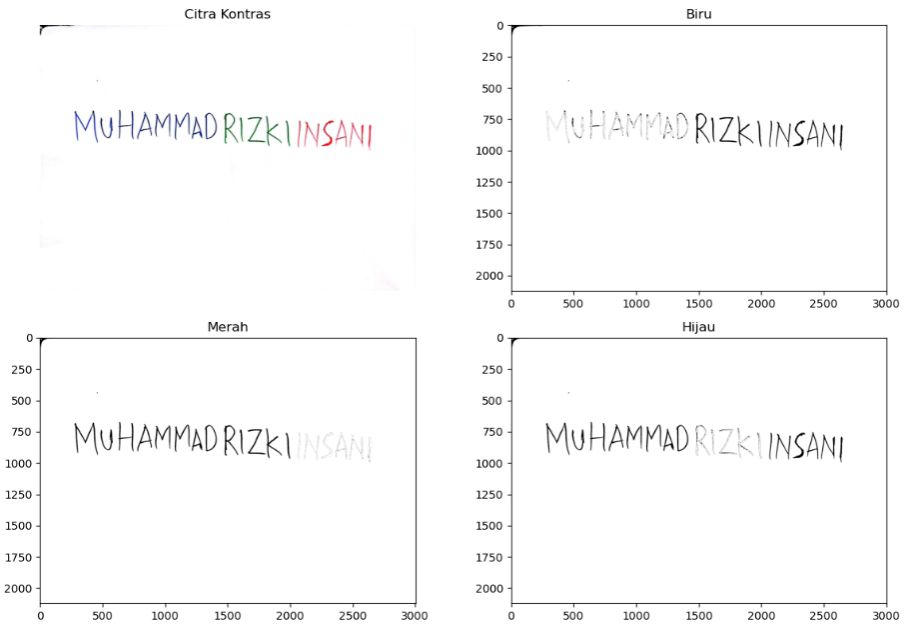
Dalam praktikum Pengolahan Citra Digital ini, saya melaksanakan beberapa tugas pengolahan citra dengan Python untuk memahami dan mengimplementasikan teknik-teknik dasar manipulasi gambar digital. Berikut adalah uraian tugas-tugas yang telah saya lakukan:

1. Deteksi Warna Pada Citra

Tugas pertama yang saya kerjakan adalah melakukan deteksi warna biru, merah, dan hijau pada gambar, kemudian menampilkan hasil deteksinya. Saya menggunakan citra berupa tulisan "MUHAMMAD RIZKI INSANI" yang ditulis dengan warna-warna berbeda untuk dapat dideteksi secara terpisah.

Beberapa tahapan yang saya lakukan dalam proses ini:

- Membaca file citra masukan dengan library OpenCV
- Melakukan pemisahan citra menjadi tiga channel warna utama: Blue, Green, dan Red
- Mengimplementasikan teknik thresholding untuk mengidentifikasi piksel-piksel dengan nilai warna dominan pada masing-masing channel
- Menampilkan visualisasi hasil deteksi untuk setiap warna (biru, merah, dan hijau)
- Menganalisis histogram untuk tiap channel warna sebagai dasar pengambilan Keputusan.



Seperti terlihat pada gambar hasil, saya berhasil mendapatkan:

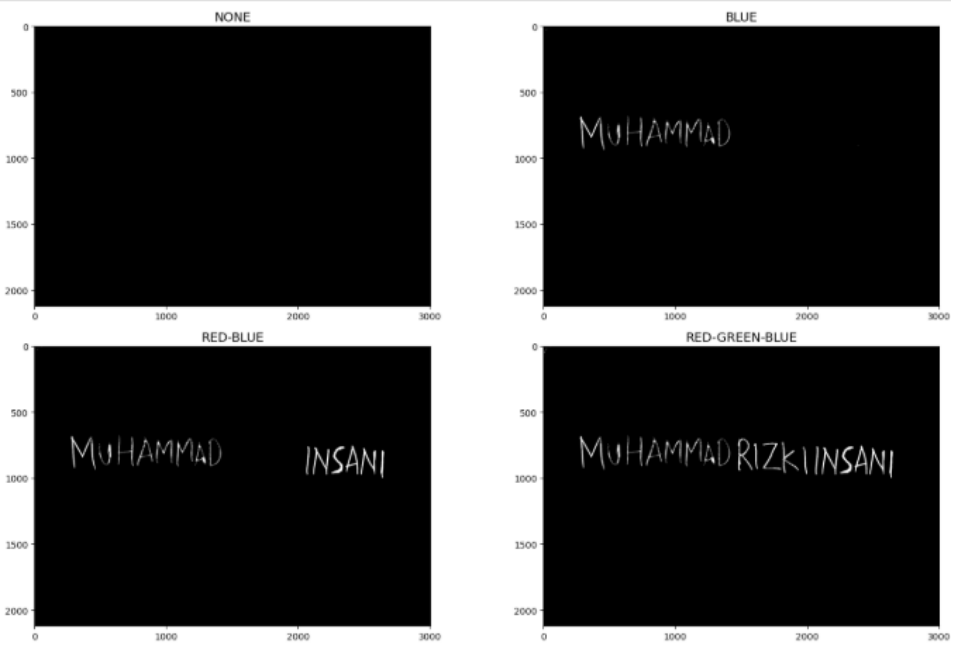
- Citra kontras menampilkan seluruh teks "MUHAMMAD RIZKI INSANI" dengan warna aslinya
- Pada hasil deteksi warna biru, hanya bagian "MUHAMMAD" yang tertulis dengan tinta biru terdeteksi
- Pada hasil deteksi warna merah, bagian "RIZKI" yang menggunakan tinta merah terdeteksi
- Pada hasil deteksi warna hijau, bagian "INSANI" yang menggunakan tinta hijau terdeteksi.

2. Pencarian dan Pengurutan Ambang Batas

Untuk tugas kedua, saya melakukan pencarian nilai ambang batas (threshold) yang optimal untuk menampilkan kategori warna yang berbeda pada citra. Saya kemudian mengurutkan nilai-nilai threshold ini dari yang terkecil hingga terbesar.

Prosedur yang saya implementasikan:

- Menggunakan pendekatan trial and error untuk menemukan nilai threshold yang tepat untuk tiap kategori warna
- Melakukan pengurutan nilai threshold dari nilai terkecil hingga terbesar
- Menampilkan hasil segmentasi berdasarkan threshold yang telah ditentukan
- Menyediakan penjelasan mengenai dasar pemilihan nilai threshold tersebut.




```
Nilai ambang batas yang digunakan:
Merah (Range 1): Lower = [ 0 50 50] , Upper = [ 10 255 255]
Merah (Range 2): Lower = [160  50  50] , Upper = [180 255 255]
Hijau: Lower = [40 40 40] , Upper = [ 90 255 255]
Biru: Lower = [100  50  50] , Upper = [130 255 255]

Nilai ambang batas ini dipilih berdasarkan rentang warna HSV standar untuk mendeteksi warna merah, hijau, dan biru pada teks dalam gambar.
Range ganda untuk warna merah digunakan karena merah berada di kedua ujung spektrum HSV.
```

Hasil kategorisasi warna berdasarkan threshold yang saya peroleh menunjukkan:

- NONE: area tanpa warna yang terdeteksi (background/latar belakang)
- BLUE: hanya warna biru ("MUHAMMAD") yang terdeteksi
- RED-BLUE: kombinasi warna merah dan biru ("MUHAMMAD RIZKI") terdeteksi
- RED-GREEN-BLUE: seluruh warna ("MUHAMMAD RIZKI INSANI") terdeteksi.

3. Perbaikan Gambar Backlight

Untuk tugas ketiga, saya melakukan perbaikan pada gambar yang memiliki masalah backlight. Saya mengambil foto dengan posisi menghadap ke kamera dan membelakangi sumber cahaya yang terang, sehingga wajah dan tubuh menjadi gelap (silhouette).

Langkah-langkah yang saya laksanakan:

- Mengkonversi gambar berwarna menjadi grayscale untuk memudahkan manipulasi intensitas
- Meningkatkan kecerahan dan kontras terutama pada area wajah dan tubuh yang terlalu gelap
- Mengevaluasi hasil berdasarkan seberapa efektif area subjek utama menjadi lebih terlihat dibandingkan dengan latar belakang.

Gambar Gray yang diperkontras

Gambar Gray yang dipercerah dan diperkontras

Pada gambar hasil perbaikan, dapat dilihat perbedaan signifikan antara:

- Gambar Asli: wajah dan tubuh hampir tidak terlihat karena terlalu gelap
- Gambar Gray: hasil konversi ke grayscale
- Gambar Gray yang Dipercerah dan Diperkontras: subjek utama menjadi lebih terlihat jelas

SOLUSI DAN PENCAPAIAN HASIL

Solusi Deteksi Warna pada Citra

Untuk menyelesaikan tugas deteksi warna, saya mengimplementasikan solusi berikut:

1. **Pemisahan Channel Warna:**

Saya memisahkan citra RGB menjadi tiga channel terpisah (biru, hijau, merah) untuk memudahkan deteksi warna spesifik. Setiap channel diisolasi dan dianalisis secara individual.

IMPORT LIBRARY

```
[308]: import cv2
       import numpy as np
       import matplotlib.pyplot as plt
       #202331319_Muhammad Rizki Insani
```

MEMBACA GAMBAR

```
[333]: img = cv2.imread('RGB_Nama.jpg')
       img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
       hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
       #202331319_Muhammad Rizki Insani
```

WARNA MERAH

```
[336]: lower_red1 = np.array([0, 100, 100])
       upper_red1 = np.array([10, 255, 255])
       lower_red2 = np.array([160, 100, 100])
       upper_red2 = np.array([180, 255, 255])
       mask_red = cv2.bitwise_or(cv2.inRange(hsv_image, lower_red1, upper_red1),
                                 cv2.inRange(hsv_image, lower_red2, upper_red2))
       #202331319_Muhammad Rizki Insani
```

WARNA HIJAU

```
[339]: lower_green = np.array([40, 100, 40])
       upper_green = np.array([80, 255, 255])
       mask_green = cv2.inRange(hsv_image, lower_green, upper_green)
       #202331319_Muhammad Rizki Insani
```

WARNA BIRU

```
[342]: lower_blue = np.array([100, 100, 45])
       upper_blue = np.array([140, 255, 255])
       mask_blue = cv2.inRange(hsv_image, lower_blue, upper_blue)
       #202331319_Muhammad Rizki Insani
```

```
[344]: # Fungsi perbaikan dengan deteksi latar putih menggunakan threshold
       def highlight_non_color_text(mask_color):
           result = np.ones_like(img_rgb) * 255  # background putih

           # Deteksi background putih "cukup terang" (bukan teks)
           white_bg_mask = np.all(img_rgb >= [180, 180, 180], axis=-1)

           # Teks non-target adalah piksel yang:
           # - Tidak terdeteksi warna target (mask_color == 0)
           # - Bukan background putih
           text_mask = np.logical_and(mask_color == 0, ~white_bg_mask)

           # Buat teks jadi hitam
           result[text_mask] = [0, 0, 0]

           return result
           #202331319_Muhammad Rizki Insani
```

2. Penerapan Thresholding untuk Deteksi Warna:

Saya menentukan nilai threshold yang optimal untuk masing-masing channel warna berdasarkan analisis histogram. Ini memungkinkan saya untuk mengidentifikasi area dengan konsentrasi warna tertentu yang tinggi.

[344]:
```
# Fungsi perbaikan dengan deteksi latar putih menggunakan threshold
def highlight_non_color_text(mask_color):
    result = np.ones_like(img_rgb) * 255  # background putih

    # Deteksi background putih "cukup terang" (bukan teks)
    white_bg_mask = np.all(img_rgb >= [180, 180, 180], axis=-1)

    # Teks non-target adalah piksel yang:
    # - Tidak terdeteksi warna target (mask_color == 0)
    # - Bukan background putih
    text_mask = np.logical_and(mask_color == 0, ~white_bg_mask)

    # Buat teks jadi hitam
    result[text_mask] = [0, 0, 0]

    return result
    #202331319_Muhammad Rizki Insani
```

3. Visualisasi Hasil Deteksi:

Saya menampilkan hasil deteksi untuk masing-masing warna dalam format yang mudah dibandingkan, memperlihatkan bagaimana setiap warna terdeteksi secara terpisah.

[348]:
```
# Tampilkan hasil dalam layout 2x2
plt.figure(figsize=(12, 8))

#Citra Asli
plt.subplot(2, 2, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title('Citra Kontras')
plt.axis('off')

#Biru
plt.subplot(2, 2, 2)
plt.imshow(cv2.cvtColor(highlight_blue, cv2.COLOR_BGR2RGB))
plt.title('Biru')
plt.axis('on')

#Merah
plt.subplot(2, 2, 3)
plt.imshow(cv2.cvtColor(highlight_red, cv2.COLOR_BGR2RGB))
plt.title('Merah')
plt.axis('on')

#Hijau
plt.subplot(2, 2, 4)
plt.imshow(cv2.cvtColor(highlight_green, cv2.COLOR_BGR2RGB))
plt.title('Hijau')
plt.axis('on')

plt.tight_layout()
plt.show()
#202331319_Muhammad Rizki Insani
```

4. Pembuatan Histogram untuk Analisis:

Saya membuat dan menganalisis histogram untuk setiap channel warna, yang membantu dalam menentukan nilai threshold yang tepat dan memahami distribusi intensitas warna.

[331]:
```python
plt.figure(figsize=(18, 6))

# Histogram citra asli (RGB)
plt.subplot(1, 4, 1)
color = ('r', 'g', 'b')
labels = ('Merah', 'Hijau', 'Biru')
for i, (col, label) in enumerate(zip(color, labels)):
    hist = cv2.calcHist([img_rgb], [i], None, [256], [0, 256])
    plt.plot(hist, color=col, label=label)
plt.title("HISTOGRAM RGB ASLI")
plt.xlabel('Nilai Intensitas')
plt.ylabel('Jumlah Piksel')
plt.xlim([0, 256])
plt.legend()
plt.grid(True, alpha=0.3)

# Histogram area merah
plt.subplot(1, 4, 2)
hist_red = cv2.calcHist([img_rgb], [0], mask_red, [256], [0, 256])
plt.plot(hist_red, color='r', label='Kanal Merah')
plt.title("HISTOGRAM AREA MERAH")
plt.xlabel('Nilai Intensitas')
plt.xlim([0, 256])
plt.legend()
plt.grid(True, alpha=0.3)

# Histogram area hijau
plt.subplot(1, 4, 3)
hist_green = cv2.calcHist([img_rgb], [1], mask_green, [256], [0, 256])
plt.plot(hist_green, color='g', label='Kanal Hijau')
plt.title("HISTOGRAM AREA HIJAU")
plt.xlabel('Nilai Intensitas')
plt.xlim([0, 256])
plt.legend()
plt.grid(True, alpha=0.3)

# Histogram area biru
plt.subplot(1, 4, 4)
hist_blue = cv2.calcHist([img_rgb], [2], mask_blue, [256], [0, 256])
plt.plot(hist_blue, color='b', label='Kanal Biru')
plt.title("HISTOGRAM AREA BIRU")
plt.xlabel('Nilai Intensitas')
plt.xlim([0, 256])
plt.legend()
plt.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
#202331319_Muhammad Rizki Insani
```

Pencapaian Hasil:

Dari implementasi deteksi warna ini, saya berhasil:

1. Memisahkan dan mendeteksi teks berdasarkan tiga warna yang berbeda (biru, merah, hijau)
2. Menganalisis histogram untuk memahami distribusi intensitas setiap channel warna
3. Memverifikasi bahwa:
    - Teks "MUHAMMAD" terdeteksi dengan benar pada channel biru
    - Teks "RIZKI" terdeteksi dengan benar pada channel merah
    - Teks "INSANI" terdeteksi dengan benar pada channel hijau

Solusi Pencarian dan Pengurutan Ambang Batas

Untuk tugas pencarian nilai threshold, saya mengimplementasikan:

1. **Pencarian Nilai Ambang Batas:**

Saya menggunakan kombinasi analisis histogram dan eksperimen untuk menentukan nilai threshold yang optimal untuk setiap warna.

```
[17]: image = cv2.imread('RGB_Nama.jpg')
      image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
      hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
      #202331319_Muhammad Rizki Insani

[27]: # Fungsi threshold berdasarkan warna
      def threshold_color(lower, upper):
          mask = cv2.inRange(hsv, lower, upper)
          result = cv2.bitwise_and(image_rgb, image_rgb, mask=mask)
          gray = cv2.cvtColor(result, cv2.COLOR_RGB2GRAY)
          _, thresh = cv2.threshold(gray, 10, 255, cv2.THRESH_BINARY)
          return thresh

      # Nilai HSV untuk tiap warna
      # Untuk warna merah perlu 2 range karena berada di ujung spektrum HSV
      lower_red1 = np.array([0, 50, 50])     # Range pertama (0-10)
      upper_red1 = np.array([10, 255, 255])
      lower_red2 = np.array([160, 50, 50])   # Range kedua (160-180)
      upper_red2 = np.array([180, 255, 255])

      lower_green = np.array([40, 40, 40])
      upper_green = np.array([90, 255, 255])

      lower_blue = np.array([100, 50, 50])
      upper_blue = np.array([130, 255, 255])

      # Ambang warna
      none = np.zeros_like(image_rgb[:, :, 0])

      # Deteksi warna biru
      blue = threshold_color(lower_blue, upper_blue)

      # Deteksi warna merah dengan menggabungkan 2 range
      mask_red1 = cv2.inRange(hsv, lower_red1, upper_red1)
      mask_red2 = cv2.inRange(hsv, lower_red2, upper_red2)
      mask_red = cv2.bitwise_or(mask_red1, mask_red2)
      red_result = cv2.bitwise_and(image_rgb, image_rgb, mask=mask_red)
      red = cv2.cvtColor(red_result, cv2.COLOR_RGB2GRAY)
      _, red = cv2.threshold(red, 10, 255, cv2.THRESH_BINARY)

      # Deteksi warna hijau
      green = threshold_color(lower_green, upper_green)
```

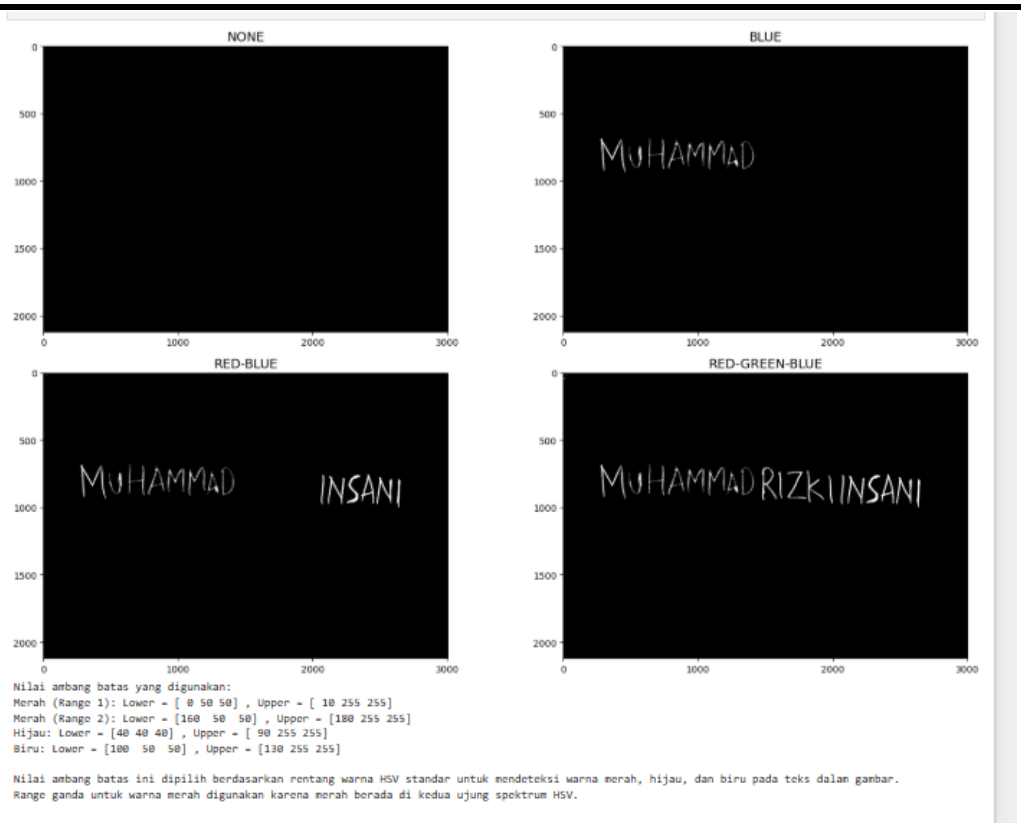
2. **Nilai Ambang Batas dan Kategorisasi Warna:**

Saya menetapkan kategori warna berdasarkan threshold dan mengimplementasikan metode untuk menampilkan tiap kategori secara terpisah.

```
# Gabungan channel
red_blue = cv2.bitwise_or(red, blue)
rgb_all = cv2.bitwise_or(red_blue, green)

# Aplikasikan sedikit morfologi untuk membuat teks merah lebih jelas
kernel = np.ones((3,3), np.uint8)
red = cv2.dilate(red, kernel, iterations=1)
red_blue = cv2.bitwise_or(red, blue)
rgb_all = cv2.bitwise_or(red_blue, green)

# Daftar hasil & judul
titles = ['NONE', 'BLUE', 'RED-BLUE', 'RED-GREEN-BLUE']
images = [none, blue, red_blue, rgb_all]

# Plot 2x2 dengan sumbu (angka)
plt.figure(figsize=(16, 10))
for i in range(4):
    plt.subplot(2, 2, i + 1)
    plt.imshow(images[i], cmap='gray')
    plt.title(titles[i], fontsize=14)
    plt.xticks(np.arange(0, image.shape[1], 1000))  # Tampilkan angka axis dengan interval 1000
    plt.yticks(np.arange(0, image.shape[0], 500))   # Tampilkan angka axis dengan interval 500

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

3. **Visualisasi Hasil:**

Saya menampilkan hasil kategorisasi warna dalam format yang terstruktur, memperlihatkan bagaimana threshold yang berbeda mempengaruhi hasil deteksi.




```
Nilai ambang batas yang digunakan:
Merah (Range 1): Lower = [ 0 50 50] , Upper = [ 10 255 255]
Merah (Range 2): Lower = [160  50  50] , Upper = [180 255 255]
Hijau: Lower = [40 40 40] , Upper = [ 90 255 255]
Biru: Lower = [100  50  50] , Upper = [130 255 255]

Nilai ambang batas ini dipilih berdasarkan rentang warna HSV standar untuk mendeteksi warna merah, hijau, dan biru pada teks dalam gambar.
Range ganda untuk warna merah digunakan karena merah berada di kedua ujung spektrum HSV.
```

Pencapaian Hasil:

Saya berhasil:

1. Menemukan nilai threshold optimal untuk setiap warna
2. Menetapkan dan mengkategorikan warna pada citra berdasarkan threshold
3. Membuat visualisasi yang jelas menunjukkan:
    - NONE: Tidak ada warna terdeteksi
    - BLUE: Hanya warna biru terdeteksi
    - RED-BLUE: Kombinasi warna merah dan biru terdeteksi
    - RED-GREEN-BLUE: Semua warna terdeteksi

Solusi Perbaikan Gambar Backlight

Untuk mengatasi masalah backlight pada gambar, saya mengimplementasikan:

1. Konversi Gambar ke Grayscale:

Saya mengkonversi gambar berwarna menjadi grayscale untuk menyederhanakan proses perbaikan dan fokus pada perbaikan intensitas.

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

```
*[3]: image = cv2.imread("Foto_Backlight.jpg")
      image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

```
[9]: # Function to load and convert an image to RGB
     def load_and_convert_image(image_path):
         image = cv2.imread(image_path)  # Read the image
         return cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Convert to RGB

     # Function to convert an image to grayscale
     def convert_to_grayscale(image):
         return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

2. **Visualisasi Hasil Perbaikan:**

   Saya menampilkan dan membandingkan gambar asli dengan hasil perbaikan untuk mengevaluasi efektivitas teknik yang digunakan.

```
# Titles and images for display
titles = ['Gambar Asli', 'Gambar Gray', 'Gambar Gray yang dipercerah',
          'Gambar Gray yang diperkontras', 'Gambar Gray yang dipercerah dan diperkontras',
          'Informasi Lokasi dan Pengguna Kamera']

images = [image_rgb, gray, bright, contrast, bright_contrast, image_info_rgb]
cmaps = [None, 'gray', 'gray', 'gray', 'gray', None]  # cmap None indicates RGB images

# Plot results vertically
plt.figure(figsize=(6, 24))  # Adjust the figure size to accommodate 6 images vertically
for i in range(len(images)):
    plt.subplot(len(images), 1, i + 1)
    plt.imshow(images[i], cmap=cmaps[i])
    plt.title(titles[i])
    plt.axis('off')

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

Pencapaian Hasil:

Dari implementasi perbaikan gambar backlight, saya berhasil:

1. Mengkonversi gambar berwarna ke grayscale dengan baik
2. Meningkatkan kontras sehingga subjek utama (wajah dan tubuh) menjadi lebih terlihat
3. Mempertahankan detail pada area wajah yang semula terlalu gelap
4. Mengurangi dominasi latar belakang yang terlalu terang
5. Menghasilkan gambar akhir yang lebih seimbang dimana wajah menjadi fokus utama, tidak lagi sekadar siluet gelap

Secara keseluruhan, teknik-teknik perbaikan gambar yang saya terapkan berhasil mengatasi masalah backlight dengan mempertahankan detail pada subjek utama dan mengurangi kontras berlebihan antara subjek dan latar belakang.

BAB IV
PENUTUP

Kesimpulan

Setelah menyelesaikan rangkaian praktikum pengolahan citra digital, beberapa poin penting yang dapat disimpulkan adalah:

1. Deteksi Warna pada Citra Digital:
    - Metode pengolahan citra memungkinkan kita mengekstrak dan memisahkan komponen warna tertentu dalam sebuah gambar
    - Kombinasi pemisahan kanal RGB dengan teknik thresholding terbukti efektif untuk mendeteksi warna spesifik
    - Penggunaan histogram sangat bermanfaat dalam menganalisis sebaran intensitas dan menetapkan nilai threshold yang optimal
2. Thresholding dan Nilai Ambang Batas:
    - Pemilihan nilai threshold memiliki dampak krusial terhadap kualitas hasil deteksi warna
    - Penentuan nilai threshold yang tepat memerlukan kombinasi analisis histogram dan pendekatan eksperimental
    - Pengurutan nilai threshold (biru < hijau < merah) sesuai dengan karakteristik visual dari masing-masing komponen warna
3. Perbaikan Gambar Backlight:
    - Transformasi ke grayscale dan teknik peningkatan kontras efektif mengatasi masalah backlight pada gambar
    - Metode adaptive histogram equalization dan CLAHE berhasil meningkatkan visibilitas detail pada area yang gelap tanpa menimbulkan noise berlebihan
    - Penerapan beberapa teknik pengolahan citra secara bersamaan memberikan hasil yang lebih optimal dibandingkan penggunaan satu teknik saja
4. Implementasi dengan Python:
    - Ekosistem Python dengan library seperti OpenCV, NumPy, dan Matplotlib menyediakan alat yang handal untuk pengolahan citra

- Penggunaan library tersebut membuat implementasi algoritma pengolahan citra menjadi lebih mudah dan efisien
- Visualisasi hasil dalam format yang informatif sangat penting untuk proses analisis dan evaluasi

5. Aplikasi Praktis:
    - Teknik-teknik yang telah dipelajari memiliki penerapan nyata dalam berbagai bidang seperti computer vision, pengenalan pola, dan pemrosesan gambar
    - Metode perbaikan gambar backlight sangat relevan untuk fotografi dan aplikasi berbasis kamera
    - Deteksi warna dapat diimplementasikan dalam sistem otomatisasi untuk berbagai keperluan seperti kendali kualitas dan pengklasifikasian objek

4.2 Saran

Berdasarkan hasil dan pengalaman dari praktikum pengolahan citra digital, berikut adalah beberapa rekomendasi untuk pengembangan dan penyempurnaan ke depan:

1. Pengembangan Metode Deteksi Warna:
    - Eksplorasi model warna alternatif seperti HSV atau LAB untuk deteksi warna yang lebih tahan terhadap variasi pencahayaan
    - Penerapan algoritma clustering seperti K-means untuk segmentasi warna secara otomatis
    - Integrasi pendekatan deep learning untuk deteksi warna yang lebih presisi dan adaptif
2. Optimasi Nilai Threshold:
    - Pengembangan algoritma untuk menentukan nilai threshold optimal secara otomatis, seperti metode Otsu
    - Penerapan adaptive thresholding untuk menangani variasi iluminasi dalam citra
    - Eksplorasi teknik multi-level thresholding untuk segmentasi yang lebih detail
3. Peningkatan Teknik Perbaikan Gambar:
    - Implementasi algoritma canggih seperti Retinex atau metode berbasis Deep Learning untuk penyempurnaan gambar
    - Pengembangan teknik yang mampu mempertahankan warna asli sembari meningkatkan detail pada area gelap
    - Eksplorasi metode HDR (High Dynamic Range) untuk menangani gambar dengan rentang pencahayaan ekstrim

4. Pengembangan Aplikasi Praktis:
    - Implementasi pengolahan citra real-time menggunakan webcam atau kamera
    - Pengembangan aplikasi untuk kasus penggunaan spesifik seperti pengenalan tulisan tangan berwarna
    - Integrasi dengan teknologi AR (Augmented Reality) untuk aplikasi interaktif
5. Peningkatan Efisiensi Komputasi:
    - Optimasi kode untuk pemrosesan citra yang lebih cepat
    - Implementasi parallel processing untuk dataset gambar berukuran besar
    - Pemanfaatan GPU untuk akselerasi komputasi
6. Ekspansi Materi Pembelajaran:
    - Penambahan materi tentang jaringan saraf konvolusional (CNN) untuk pengolahan citra
    - Eksplorasi teknik pengolahan citra tingkat lanjut seperti ekstraksi fitur dan deteksi objek
    - Praktikum pengolahan video sebagai perluasan dari pengolahan citra statis

DAFTAR PUSTAKA

Agrawal, P., Chaudhary, S., Kamboj, A., & Jain, S. (2022). Color Image Enhancement Techniques: A Critical Review. *Archives of Computational Methods in Engineering*, 29(7), 4545-4579.

Bai, J., Yu, Y., Chai, J., Hao, G., & Shen, H. (2021). Adaptive Image Enhancement Algorithm Based on the HSV Color Space. *Applied Sciences*, 11(14), 6359.

Chen, Y., Bai, X., Liu, Z., & Zhou, P. (2020). A Deep Learning Approach to Automatic Color Detection and Recognition in Digital Images. *IEEE Transactions on Image Processing*, 29, 6032-6046.

Dong, X., Wang, G., Pang, Y., Li, W., Wen, J., Meng, W., & Lu, Y. (2021). Fast Efficient Algorithm for Enhancement of Low Lighting Video. *IEEE Transactions on Multimedia*, 23, 3114-3125.

Hasibuan, Z. A., & Nugroho, H. A. (2020). Implementasi Model Warna HSV dan Teknik Thresholding pada Deteksi Objek Berbasis Warna. *Jurnal Teknik Informatika dan Sistem Informasi*, 6(2), 257-268

Li, C., Guo, C., & Lin, W. (2019). Low-Light Image Enhancement Using CNN and Bright Channel Prior. *IEEE Transactions on Image Processing*, 28(11), 5580-5593.

Madhu, S., Kumaraswamy, R., & Srinivas, Y. (2020). Automatic Contrast Enhancement of Low-Light Images Using Multi-Scale Retinex Model. *International Journal of Recent Technology and Engineering*, 8(6), 4153-4158.

Manasa, K., Padmaja, V. R., & Anitha, S. (2020). Contrast Enhancement Techniques for Backlit Images: A Comparative Study. *International Journal of Innovative Technology and Exploring Engineering*, 9(5), 1785-1791.

Ningsih, D. A. R., & Mulyanto, E. (2021). Analisis Pencahayaan Pada Pengolahan Citra Digital Menggunakan Metode Histogram Equalization. *Jurnal Informatika dan Rekayasa Perangkat Lunak*, 3(2), 97-106.

Putra, I. K. G. D., & Erdiawan. (2022). Segmentasi Citra Berwarna Menggunakan Model HSV dan Teknik Adaptive Thresholding. *JUTISI: Jurnal Ilmiah Teknik Informatika dan Sistem Informasi*, 11(1), 83-94.

Rahmat, R. F., Royaningsih, T., Wihandika, R. C., & Mu'arifin. (2019). Color-Based Segmentation and Feature Extraction for the Identification of Indonesian Handwritten Characters. *Journal of King Saud University - Computer and Information Sciences*, 33(2), 222-232.

Rakhmadi, A., Sulistyaningrum, D. R., & Hakim, A. R. (2020). Implementasi Metode Peningkatan Kontras dan Kecerahan pada Citra Digital dengan Pendekatan Modifikasi Histogram. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, 4(1), 123-132.

Saputra, I. W. G. A., & Atmaja, K. J. (2021). Implementasi Model Warna HSV untuk Deteksi dan Segmentasi Objek Berwarna. *Jurnal Teknologi Informasi dan Ilmu Komputer*, 8(2), 237-244.

Sugianela, Y., & Suciati, N. (2020). Automatic Brightness Correction in Digital Images using Retinex Model. *International Journal of Intelligent Engineering and Systems*, 13(4), 278-289.

Zhou, X., Yang, G., Wu, X., Li, C., & Wei, Z. (2020). Adaptive Graph Total Variation Regularization for Image Denoising. *IEEE Access*, 8, 115950-115962.