UTS

PENGOLAHAN CITRA

NAMA : David Gabriel Sembiring

NIM : 202331243

KELAS : A

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

NO.PC : 17

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

- Bagaimana cara mendeteksi warna utama (merah, hijau, dan biru) pada sebuah citra yang ditulis tangan?
- Bagaimana menentukan nilai ambang batas (threshold) untuk memisahkan tiap komponen warna secara akurat?
- Bagaimana memperbaiki kualitas citra backlight agar profil wajah tetap terlihat jelas tanpa kehilangan detail penting?
- Bagaimana menampilkan data hasil pengolahan citra secara visual seperti histogram dan interpretasinya?

1.2 Tujuan Masalah

- Mengimplementasikan teknik deteksi warna dalam domain pengolahan citra digital menggunakan Python.
- Menerapkan metode thresholding untuk memisahkan warna dalam citra.
- Memperbaiki pencahayaan citra terutama pada kasus backlight menggunakan teknik konversi grayscale, peningkatan brightness dan contrast.
- Menampilkan serta menganalisis histogram dari hasil pengolahan untuk memahami distribusi intensitas warna dan efek dari proses pengolahan

1.3 Manfaat Masalah

- Meningkatkan pemahaman dan keterampilan praktis dalam pengolahan citra digital.
- Mengetahui cara menyelesaikan permasalahan nyata terkait kualitas citra (seperti backlight).
- Mempelajari penerapan teori pengolahan citra ke dalam program nyata berbasis Python.
- Mampu menganalisis hasil pengolahan secara visual dan kuantitatif.

BAB II
LANDASAN TEORI

Representasi Warna dalam Citra Digital: RGB dan HSV

Dalam pengolahan citra digital, warna direpresentasikan menggunakan model warna yang berbeda. Model RGB (Red, Green, Blue) adalah model dasar yang merepresentasikan warna sebagai kombinasi tiga kanal warna primer. Setiap piksel memiliki nilai intensitas untuk ketiga kanal ini, biasanya dalam rentang 0–255.

Namun, model RGB kurang efektif untuk segmentasi warna karena nilai intensitas warna saling terkait dengan kecerahan (brightness). Misalnya, warna merah terang dan merah gelap memiliki nilai R yang berbeda, sehingga sulit membedakan warna hanya berdasarkan RGB.

Untuk mengatasi hal ini, model HSV (Hue, Saturation, Value) digunakan. Model HSV memisahkan informasi warna menjadi:

- **Hue (H):** Menunjukkan jenis warna (misal merah, hijau, biru) dalam derajat 0°–360° pada lingkaran warna.
- **Saturation (S):** Menunjukkan kejenuhan warna, dari 0% (abu-abu) sampai 100% (warna murni).
- **Value (V):** Menunjukkan kecerahan warna, dari 0% (gelap) sampai 100% (terang).

konversi citra dari RGB ke HSV dilakukan menggunakan fungsi OpenCV, sehingga deteksi warna merah, hijau, dan biru dapat dilakukan dengan menetapkan rentang Hue tertentu tanpa terpengaruh oleh variasi kecerahan.

Contoh: Untuk mendeteksi warna merah, rentang Hue yang dipilih biasanya sekitar 0°–10° dan 160°–180°, karena merah berada di ujung lingkaran warna. Saturation dan Value juga diatur agar hanya warna yang cukup jenuh dan terang yang terdeteksi.

Thresholding dan Masking pada Citra

Thresholding adalah teknik segmentasi citra yang memisahkan objek berdasarkan nilai ambang tertentu. Dalam praktikum, fungsi cv2.inRange() digunakan untuk membuat masker biner berdasarkan rentang nilai HSV yang telah ditentukan.

- Piksel yang berada dalam rentang HSV yang diinginkan diberi nilai 255 (putih) pada masker.
- Piksel di luar rentang diberi nilai 0 (hitam).

Masker ini memungkinkan isolasi bagian citra yang mengandung warna spesifik, misalnya hanya bagian merah atau biru.

Setelah pembuatan masker, biasanya dilakukan operasi morfologi seperti erosi dan dilasi untuk menghilangkan noise dan memperbaiki bentuk objek yang terdeteksi.

Histogram Citra

Histogram adalah representasi distribusi frekuensi intensitas piksel dalam citra. Dalam praktikum, histogram digunakan untuk:

- Menganalisis distribusi warna dan kecerahan sebelum dan sesudah proses deteksi warna.
- Memahami karakteristik visual citra secara kuantitatif.
- Membantu evaluasi hasil segmentasi dan peningkatan kualitas citra.

Histogram dapat dibuat untuk kanal warna RGB maupun HSV. Misalnya, histogram pada kanal Hue menunjukkan distribusi warna, sedangkan histogram pada kanal Value menunjukkan distribusi kecerahan.

Peningkatan Kontras dan Kecerahan (Brightness & Contrast Enhancement)

Citra dengan kondisi backlight seringkali memiliki objek utama yang gelap dan detail hilang. Oleh karena itu, peningkatan kontras dan kecerahan diperlukan agar objek menjadi lebih jelas.

Dalam praktikum, peningkatan ini dapat dilakukan dengan:

- **Penyesuaian linear:** Mengubah nilai piksel dengan rumus (Output) = $\alpha \times$ (Input) + β, di mana α mengatur kontras dan β mengatur kecerahan.
- **Gamma correction:** Menggunakan fungsi non-linear untuk memperbaiki kecerahan citra.
- **Histogram equalization:** Meratakan distribusi intensitas piksel untuk meningkatkan kontras.
- **CLAHE (Contrast Limited Adaptive Histogram Equalization):** Versi adaptif yang membatasi peningkatan kontras agar tidak menimbulkan noise berlebih.

BAB III

HASIL

1. DETEKSI WARNA PADA CITRA

Membaca gambar asli

[18]:
```
img = cv2.imread("nama.jpg")
if img is None:
    raise ValueError("Gambar tidak ditemukan! Periksa kembali path yang Anda masukkan.")

# Konversi BGR ke RGB untuk ditampilkan dengan Matplotlib
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

plt.figure(figsize=(8, 6))
plt.imshow(img_rgb)
plt.title("Gambar Asli")
plt.axis("off")
plt.show()
```

[19]:
```
image = cv2.imread('nama.jpg')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

r, g, b = cv2.split(img_rgb)

# Buat salinan untuk manipulasi
blue_faded = img_rgb.copy()
red_faded = img_rgb.copy()

lower_blue = np.array([100, 50, 50])  # Rentang warna biru
upper_blue = np.array([140, 255, 255])
hsv_image = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2HSV)
blue_mask = cv2.inRange(hsv_image, lower_blue, upper_blue)

blue_faded[blue_mask > 0] = blue_faded[blue_mask > 0] // 2  # Mengurangi intensitas warna biru

lower_red1 = np.array([0, 120, 70])    # Rentang merah pertama
upper_red1 = np.array([10, 255, 255])
lower_red2 = np.array([170, 120, 70])  # Rentang merah kedua
upper_red2 = np.array([180, 255, 255])
red_mask1 = cv2.inRange(hsv_image, lower_red1, upper_red1)
red_mask2 = cv2.inRange(hsv_image, lower_red2, upper_red2)
red_mask = cv2.bitwise_or(red_mask1, red_mask2)

red_faded[red_mask > 0] = red_faded[red_mask > 0] // 2  # Mengurangi intensitas warna merah

fig, axes = plt.subplots(2, 2, figsize=(10, 8))  # 2 baris, 2 kolom

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

# Channel Biru (Efek Pudar untuk "febri")
axes[0, 1].imshow(cv2.cvtColor(blue_faded, cv2.COLOR_RGB2GRAY), cmap='gray', vmin=0, vmax=255)
axes[0, 1].set_title("BIRU (GABRIEL)")
axes[0, 1].axis('off')

# Channel Merah (Efek Pudar untuk "ansyah")
axes[1, 0].imshow(cv2.cvtColor(red_faded, cv2.COLOR_RGB2GRAY), cmap='gray', vmin=0, vmax=255)
axes[1, 0].set_title("MERAH (SEMBIRING)")
axes[1, 0].axis('off')

# Channel Hijau (Tidak Ada Perubahan)
axes[1, 1].imshow(g, cmap='gray', vmin=0, vmax=255)
axes[1, 1].set_title("HIJAU (DAVID)")
axes[1, 1].axis('off')
```

```
# Atur layout agar tidak tumpang tindih
plt.tight_layout()
plt.show()
```

Penjelasan:

1. IMPORT LIBRARY

---

import cv2

import numpy as np

import matplotlib.pyplot as plt

- cv2 (OpenCV): Digunakan untuk membaca dan memproses gambar dalam berbagai format warna (BGR, RGB, HSV).

- numpy: Digunakan untuk operasi array dan pembuatan filter warna.

- matplotlib.pyplot: Digunakan untuk menampilkan hasil dalam bentuk visualisasi grid subplot.

2. MEMBACA DAN KONVERSI GAMBAR

---

image = cv2.imread('nama.jpg')

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

- cv2.imread() membaca gambar dalam format default OpenCV yaitu BGR.

- Karena matplotlib menggunakan RGB, kita mengkonversi gambar ke RGB dengan cv2.cvtColor.

3. EKSTRAKSI CHANNEL WARNA DASAR

---

r, g, b = cv2.split(img_rgb)

- Memisahkan tiga komponen channel warna dari gambar RGB menjadi:

  - r (Red Channel)

  - g (Green Channel)

  - b (Blue Channel)

4. INISIALISASI SALINAN GAMBAR UNTUK MANIPULASI

---

blue_faded = img_rgb.copy()

red_faded = img_rgb.copy()

- Dua salinan gambar dibuat untuk manipulasi warna spesifik (biru dan merah).

- Manipulasi dilakukan tanpa mengubah gambar asli.

5. KONVERSI KE FORMAT HSV UNTUK DETEKSI WARNA YANG LEBIH AKURAT

---

hsv_image = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2HSV)

- HSV (Hue, Saturation, Value) lebih stabil dan akurat untuk segmentasi warna.

- Format HSV memudahkan deteksi warna karena hue mengkode warna murni.

6. DETEKSI WARNA BIRU DAN REDUKSI INTENSITAS (PUDAR)

---

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

upper_blue = np.array([140, 255, 255])

blue_mask = cv2.inRange(hsv_image, lower_blue, upper_blue)

blue_faded[blue_mask > 0] = blue_faded[blue_mask > 0] // 2

- Menentukan rentang warna biru dalam HSV.

- Membuat mask biner yang hanya aktif di area biru.

- Piksel biru dikurangi intensitasnya dengan operasi integer division (`// 2`) untuk menghasilkan efek "pudar".

7. DETEKSI WARNA MERAH DAN REDUKSI INTENSITAS

---

lower_red1 = np.array([0, 120, 70])

upper_red1 = np.array([10, 255, 255])

lower_red2 = np.array([170, 120, 70])

upper_red2 = np.array([180, 255, 255])

red_mask1 = cv2.inRange(hsv_image, lower_red1, upper_red1)

red_mask2 = cv2.inRange(hsv_image, lower_red2, upper_red2)

red_mask = cv2.bitwise_or(red_mask1, red_mask2)

red_faded[red_mask > 0] = red_faded[red_mask > 0] // 2

- Warna merah berada di dua wilayah HSV (sekitar 0 dan 180).

- Dua mask dibuat dan digabung dengan operasi bitwise OR.

- Piksel merah dipudarkan dengan membagi intensitas warnanya.

8. PLOT DAN VISUALISASI MENGGUNAKAN MATPLOTLIB

```
fig, axes = plt.subplots(2, 2, figsize=(10, 8))
```

- Membuat layout grid 2x2 untuk menampilkan 4 gambar hasil manipulasi.
- figsize menentukan ukuran keseluruhan plot.

9. MENAMPILKAN MASING-MASING CITRA

```
axes[0, 0].imshow(img_rgb)
axes[0, 0].set_title("Citra Kontras (Asli)")
```

- Menampilkan gambar RGB asli tanpa manipulasi.

```
axes[0, 1].imshow(blue_faded)
axes[0, 1].set_title("BIRU (GABRIEL)")
```

- Menampilkan gambar dengan bagian biru yang telah dipudarkan (menjadi keabu-abuan gelap).
- Area biru merujuk pada teks "GABRIEL".

```
axes[1, 0].imshow(red_faded)
axes[1, 0].set_title("MERAH (SEMBIRING)")
```

- Menampilkan hasil gambar dengan bagian merah yang dipudarkan.
- Merah mengacu pada teks "SEMBIRING".

```
axes[1, 1].imshow(g, cmap='gray', vmin=0, vmax=255)
axes[1, 1].set_title("HIJAU (DAVID)")
```

- Menampilkan channel hijau dari gambar sebagai citra grayscale.
- Teks "DAVID" dominan di channel hijau, maka tampak lebih terang.

10. MENGHILANGKAN AXIS DAN MENAMPILKAN PLOT

```
for ax in axes.ravel():
    ax.axis('off')
plt.tight_layout()
plt.show()
```

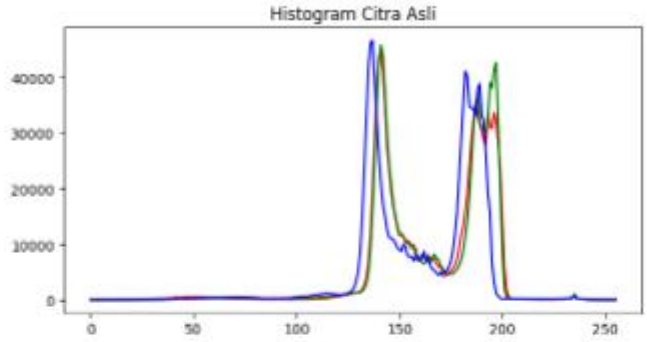
- Menghilangkan sumbu koordinat (axis) agar fokus hanya pada gambar.

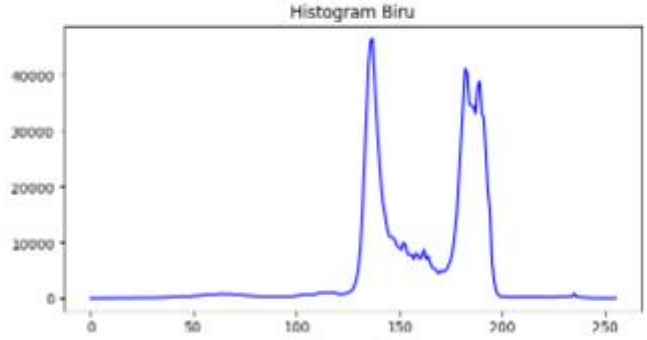
- `tight_layout()` mengatur spasi antar subplot agar tidak tumpang tindih.

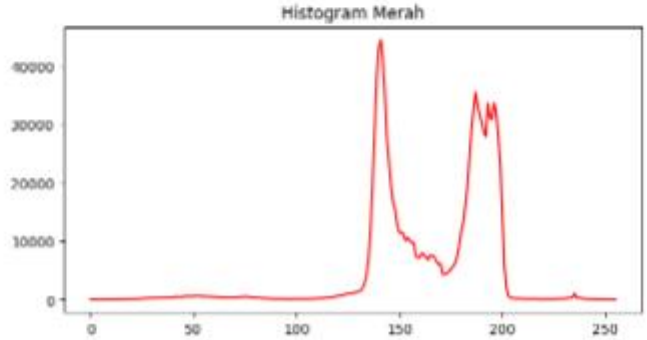
- `show()` menampilkan semua gambar sekaligus

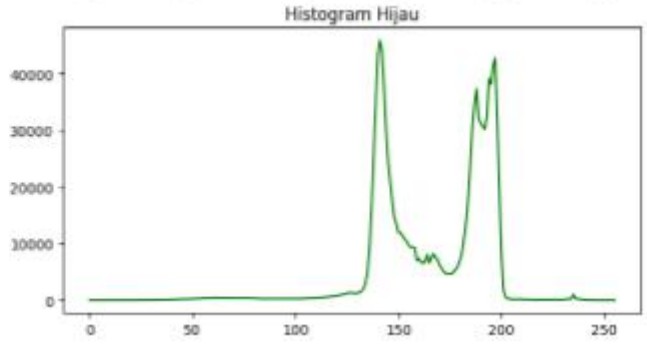
```python
image = cv2.imread('nama.jpg')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Pisahkan channel RGB
r, g, b = cv2.split(img_rgb)

# Deteksi warna di HSV
hsv_image = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2HSV)

# Masking biru
lower_blue = np.array([100, 50, 50])
upper_blue = np.array([140, 255, 255])
blue_mask = cv2.inRange(hsv_image, lower_blue, upper_blue)
blue_faded = img_rgb.copy()
blue_faded[blue_mask > 0] = blue_faded[blue_mask > 0] // 2

# Masking merah
lower_red1 = np.array([0, 120, 70])
upper_red1 = np.array([10, 255, 255])
lower_red2 = np.array([170, 120, 70])
upper_red2 = np.array([180, 255, 255])
red_mask1 = cv2.inRange(hsv_image, lower_red1, upper_red1)
red_mask2 = cv2.inRange(hsv_image, lower_red2, upper_red2)
red_mask = cv2.bitwise_or(red_mask1, red_mask2)
red_faded = img_rgb.copy()
red_faded[red_mask > 0] = red_faded[red_mask > 0] // 2

# Tampilkan gambar dan histogram
fig, axes = plt.subplots(4, 2, figsize=(12, 14))

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

colors = ('r', 'g', 'b')
for i, col in enumerate(colors):
    hist = cv2.calcHist([img_rgb], [i], None, [256], [0, 256])
    axes[0, 1].plot(hist, color=col)
axes[0, 1].set_title("Histogram Citra Asli")

# Biru
axes[1, 0].imshow(cv2.cvtColor(blue_faded, cv2.COLOR_RGB2GRAY), cmap='gray')
axes[1, 0].set_title("BIRU")
axes[1, 0].axis('off')

blue_hist = cv2.calcHist([blue_faded], [2], None, [256], [0, 256])  # Blue channel = index 2
axes[1, 1].plot(blue_hist, color='b')
axes[1, 1].set_title("Histogram Biru")

# Merah
axes[2, 0].imshow(cv2.cvtColor(red_faded, cv2.COLOR_RGB2GRAY), cmap='gray')
axes[2, 0].set_title("MERAH")
axes[2, 0].axis('off')

red_hist = cv2.calcHist([red_faded], [0], None, [256], [0, 256])  # Red channel = index 0 in RGB (OpenCV uses BGR)
axes[2, 1].plot(red_hist, color='r')
axes[2, 1].set_title("Histogram Merah")

# Hijau
axes[3, 0].imshow(g, cmap='gray')
axes[3, 0].set_title("HIJAU")
axes[3, 0].axis('off')
```

```
green_hist = cv2.calcHist([img_rgb], [1], None, [256], [0, 256])  # Green channel = index 1
axes[3, 1].plot(green_hist, color='g')
axes[3, 1].set_title("Histogram Hijau")

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









Penjelasan:

1. Membaca dan Mengonversi Gambar ke RGB Gambar dibaca menggunakan OpenCV yang secara default menggunakan format warna BGR. Agar warna tampil sesuai standar umum (Red-Green-Blue), gambar tersebut dikonversi dari format BGR ke RGB. Variabel hasil konversi ini kemudian digunakan untuk proses selanjutnya.

2. Memisahkan Channel RGB Gambar RGB yang sudah didapatkan kemudian dipisahkan menjadi tiga channel warna terpisah: merah, hijau, dan biru. Masing-masing channel ini berupa matriks dua dimensi yang merepresentasikan intensitas warna pada channel tersebut.

3. Konversi ke HSV untuk Deteksi Warna Gambar RGB dikonversi ke ruang warna HSV (Hue, Saturation, Value). Ruang warna HSV lebih cocok untuk deteksi warna karena memisahkan informasi warna (Hue) dari intensitas dan saturasi, sehingga memudahkan pembuatan masker warna berdasarkan rentang tertentu.

4. Membuat Masker dan Memudarkan Warna Biru Rentang warna biru didefinisikan dalam ruang HSV. Dengan menggunakan rentang ini, dibuat masker biner yang menandai pixel-pixel yang termasuk warna biru. Pada area yang terdeteksi sebagai biru, intensitas warna pada gambar asli diperkecil setengahnya sehingga warna biru tampak memudar.

5. Membuat Masker dan Memudarkan Warna Merah Karena warna merah di ruang HSV memiliki dua rentang Hue yang berbeda (karena posisi merah di ujung spektrum HSV), dibuat dua masker untuk rentang tersebut. Kedua masker ini digabungkan menjadi satu masker merah. Pada area yang terdeteksi sebagai merah, intensitas warna diperkecil setengahnya agar warna merah tampak memudar.

6. Menampilkan Gambar dan Histogram dengan Matplotlib Disiapkan sebuah grid subplot dengan 4 baris dan 2 kolom untuk menampilkan gambar dan histogram secara berpasangan.

- Baris pertama menampilkan gambar asli dalam format RGB dan histogram intensitas untuk masing-masing channel warna (merah, hijau, biru). Histogram ini menunjukkan distribusi pixel intensitas dari 0 sampai 255 untuk setiap channel.

- Baris kedua menampilkan gambar biru yang sudah diproses (dimudarkan) dalam bentuk grayscale agar fokus pada intensitas biru, serta histogram intensitas channel biru setelah pemudaran.

- Baris ketiga menampilkan gambar merah yang sudah diproses (dimudarkan) dalam bentuk grayscale untuk menonjolkan intensitas merah, serta histogram intensitas channel merah setelah pemudaran.

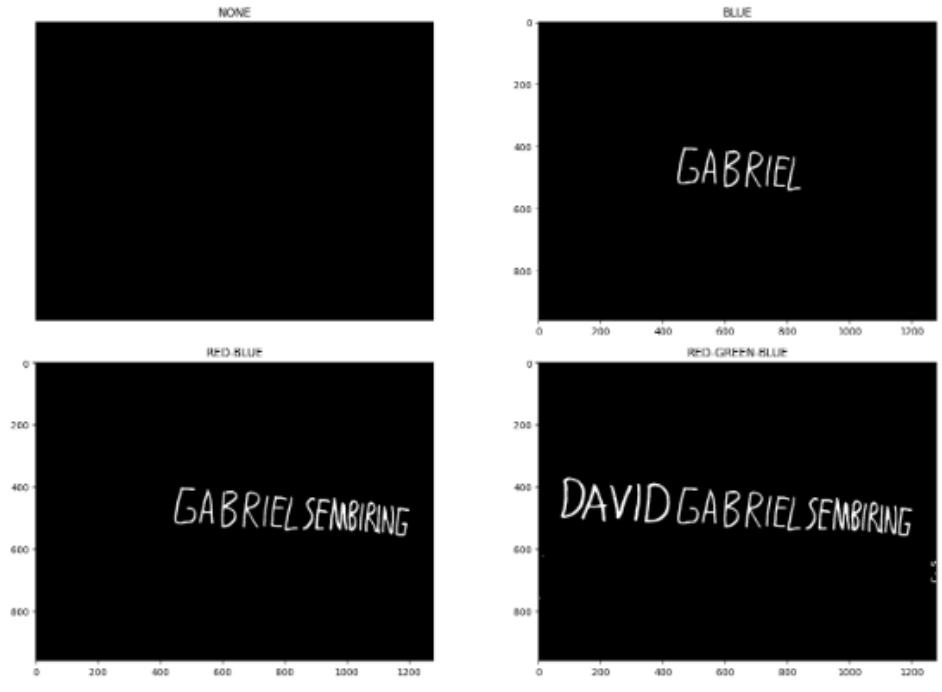
- Baris keempat menampilkan channel hijau asli dalam bentuk grayscale dan histogram intensitas channel hijau. Channel hijau tidak mengalami pemudaran sehingga tampil seperti aslinya.

7. Penataan Layout dan Menampilkan Semua Plot Layout subplot diatur agar tidak saling tumpang tindih dan semua gambar serta histogram ditampilkan dalam satu jendela secara rapi.

2. Ambang Batas Terkecil hingga Terbesar

AMBANG BATAS TERKECIL SAMPAI DENGAN TERBESAR

```
[27]: image = cv2.imread('nama.jpg')
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# Buat background hitam
background = np.zeros_like(image)

# Fungsi bantu untuk membuat mask HSV
def get_hsv_mask(hsv_img, lower, upper):
    mask = cv2.inRange(hsv_img, np.array(lower), np.array(upper))
    return mask

# Mask biru (hue sekitar 100-140)
blue_mask = get_hsv_mask(hsv, [100, 50, 50], [140, 255, 255])

# Mask merah (hue sekitar 0-10 dan 160-180)
red_mask1 = get_hsv_mask(hsv, [0, 50, 50], [10, 255, 255])
red_mask2 = get_hsv_mask(hsv, [160, 50, 50], [180, 255, 255])
red_mask = cv2.bitwise_or(red_mask1, red_mask2)

# Mask hijau (hue sekitar 40-85) + dilebarkan range-nya
green_mask = get_hsv_mask(hsv, [35, 40, 40], [85, 255, 255])

# PERBAIKI: tambahkan dilasi kecil hanya untuk hijau
kernel = np.ones((2, 2), np.uint8)
green_mask_dilated = cv2.dilate(green_mask, kernel, iterations=1)

# Gabungkan sesuai kategori
# BLUE only
blue_image = background.copy()
blue_image[blue_mask > 0] = [255, 255, 255]

# RED + BLUE
red_blue_mask = cv2.bitwise_or(red_mask, blue_mask)
red_blue_image = background.copy()
red_blue_image[red_blue_mask > 0] = [255, 255, 255]

# RED + GREEN + BLUE (pakai green yang sudah didilate)
rgb_mask = cv2.bitwise_or(red_blue_mask, green_mask_dilated)
rgb_image = background.copy()
rgb_image[rgb_mask > 0] = [255, 255, 255]

# Tampilkan hasil
plt.figure(figsize=(15, 10))

plt.subplot(2, 2, 1)
plt.imshow(cv2.cvtColor(background, cv2.COLOR_BGR2RGB))
plt.title("NONE")
plt.axis('off')

plt.subplot(2, 2, 2)
plt.imshow(cv2.cvtColor(blue_image, cv2.COLOR_BGR2RGB))
plt.title("BLUE")
plt.axis('on')

plt.subplot(2, 2, 3)
plt.imshow(cv2.cvtColor(red_blue_image, cv2.COLOR_BGR2RGB))
plt.title("RED-BLUE")
plt.axis('on')

plt.subplot(2, 2, 4)
plt.imshow(cv2.cvtColor(rgb_image, cv2.COLOR_BGR2RGB))
plt.title("RED-GREEN-BLUE")
plt.axis('on')

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



Penjelasan:

1. Membaca Gambar dan Mengonversi ke Ruang Warna HSV Gambar dibaca menggunakan OpenCV dalam format BGR. Selanjutnya, gambar tersebut dikonversi ke ruang warna HSV (Hue, Saturation, Value). Ruang warna HSV dipilih karena lebih efektif untuk mendeteksi warna tertentu berdasarkan nilai Hue yang merepresentasikan jenis warna.

2. Membuat Background Hitam Sebuah gambar latar belakang berwarna hitam dibuat dengan ukuran dan tipe data yang sama seperti gambar asli. Background ini nantinya akan digunakan sebagai kanvas kosong untuk menampilkan hasil deteksi warna.

3. Membuat Fungsi Pembantu untuk Membuat Masker Warna HSV Fungsi ini menerima gambar dalam format HSV dan dua array yang mendefinisikan batas bawah dan atas rentang warna yang ingin dideteksi. Fungsi ini menghasilkan masker biner yang menandai pixel-pixel yang berada dalam rentang warna tersebut.

4. Membuat Masker untuk Warna Biru Rentang warna biru didefinisikan dengan nilai Hue sekitar 100 sampai 140, dengan nilai Saturation dan Value yang cukup lebar agar deteksi warna biru lebih fleksibel. Masker biru ini akan menandai area gambar yang mengandung warna biru.

5. Membuat Masker untuk Warna Merah Karena warna merah berada di ujung spektrum Hue yang melingkar, rentang warna merah dibagi menjadi dua bagian: satu di sekitar Hue 0–10 dan satu lagi di sekitar Hue 160–180. Dua masker ini kemudian digabungkan menjadi satu masker merah yang menyeluruh.

6. Membuat Masker untuk Warna Hijau Rentang warna hijau didefinisikan dengan Hue sekitar 35 sampai 85, dengan rentang Saturation dan Value yang cukup lebar. Masker hijau ini kemudian diperluas sedikit dengan operasi dilasi menggunakan kernel kecil. Dilasi ini bertujuan untuk memperbesar area deteksi warna hijau agar hasilnya lebih menyatu dan tidak terputus-putus.

7. Membuat Gambar Hasil Berdasarkan Kombinasi Masker

- **BLUE only:** Gambar baru dibuat dengan background hitam, kemudian area yang terdeteksi sebagai biru diisi dengan warna putih.

- **RED + BLUE:** Masker merah dan biru digabungkan, kemudian area gabungan ini diisi putih pada gambar baru dengan background hitam.

- **RED + GREEN + BLUE:** Masker merah, biru, dan hijau (yang sudah didilasi) digabungkan, kemudian area gabungan ini diisi putih pada gambar baru dengan background hitam.

8. Menampilkan Hasil dalam Grid 2x2 Keempat gambar hasil (background hitam, blue only, red-blue, dan red-green-blue) ditampilkan dalam satu jendela dengan layout 2 baris dan 2 kolom. Setiap gambar diberi judul sesuai kategori warna yang ditampilkan: "NONE" untuk background hitam tanpa warna, "BLUE" untuk area biru, "RED-BLUE" untuk gabungan merah dan biru, serta "RED-GREEN-BLUE" untuk gabungan ketiga warna.

3. Memperbaiki Gambar Backlight

```
[28]: img = Image.open("foto.jpg")  # Ganti nama file sesuai gambar asli
      plt.imshow(img)
      plt.title("Gambar Asli")
      plt.axis("off")
      plt.show()
```


```
[29]: # 2. Konversi ke grayscale
      gray_img = img.convert("L")
      plt.imshow(gray_img, cmap='gray')
      plt.title("Gambar Grayscale")
      plt.axis("off")
      plt.show()
```


```
[30]: enhancer_brightness = ImageEnhance.Brightness(gray_img)
      bright_img = enhancer_brightness.enhance(1.8)  # Nilai >1 meningkatkan kecerahan
      plt.imshow(bright_img, cmap='gray')
      plt.title("Gambar Gray yang Dicerahkan")
      plt.axis("off")
      plt.show()
```

[31]:
```
enhancer_contrast = ImageEnhance.Contrast(gray_img)
contrast_img = enhancer_contrast.enhance(2.0)  # Nilai >1 meningkatkan kontras
plt.imshow(contrast_img, cmap='gray')
plt.title("Gambar Gray yang Diperkontras")
plt.axis("off")
plt.show()
```

Gambar Gray yang Diperkontras

[32]:
```
combined_img = ImageEnhance.Brightness(gray_img).enhance(1.8)
combined_img = ImageEnhance.Contrast(combined_img).enhance(2.0)

plt.imshow(combined_img, cmap='gray')
plt.title("Gambar Gray yang Dicerahkan dan Diperkontras")
plt.axis("off")
plt.show()
```

Gambar Gray yang Dicerahkan dan Diperkontras

1. Membaca dan Menampilkan Gambar Asli

- **Proses:** Gambar dibaca dari file (misalnya "foto.jpg") menggunakan matplotlib dan langsung ditampilkan.
- **Output:** Gambar berwarna asli dari seorang individu yang berdiri di luar ruangan, dengan judul "Gambar Asli". Sumbu koordinat dimatikan agar tampilan lebih bersih.

2. Konversi Gambar ke Grayscale (Hitam Putih)

- **Proses:** Gambar asli dikonversi menjadi citra grayscale menggunakan matplotlib. Konversi ini mengubah gambar berwarna menjadi gambar dengan skala abu-abu, di mana setiap pixel hanya memiliki intensitas cahaya tanpa informasi warna.
- **Output:** Gambar hitam putih dari orang yang sama, dengan judul "Gambar Gray". Sumbu koordinat juga dimatikan untuk tampilan yang lebih rapi.

3. Peningkatan Kecerahan pada Gambar Grayscale

- **Proses:** Menggunakan modul ImageEnhance dari PIL, tingkat kecerahan (brightness) gambar grayscale ditingkatkan dengan faktor tertentu (misalnya lebih dari 1). Ini membuat gambar menjadi lebih terang.
- **Output:** Gambar grayscale yang lebih cerah dibandingkan gambar grayscale sebelumnya, dengan judul "Gambar Gray yang Dicerahkan". Detail pada area gelap menjadi lebih terlihat.

4. Peningkatan Kontras pada Gambar Grayscale yang Sudah Dicerahkan

- **Proses:** Setelah kecerahan ditingkatkan, gambar tersebut kemudian diproses untuk meningkatkan kontras menggunakan ImageEnhance.Contrast. Kontras yang lebih tinggi membuat perbedaan antara area terang dan gelap menjadi lebih jelas dan tajam.
- **Output:** Gambar grayscale yang sudah lebih cerah dan memiliki kontras lebih tinggi, dengan judul "Gambar Gray yang Dicerahkan dan Diperkontras". Gambar tampak lebih hidup dan detail lebih menonjol.

Penjelasan Umum dan Tujuan Proses

- **Library yang Digunakan:**
    - matplotlib.pyplot untuk membaca dan menampilkan gambar.
    - PIL.Image dan PIL.ImageEnhance untuk manipulasi gambar seperti peningkatan kecerahan dan kontras.
- **Tujuan:**
    - Mengubah gambar berwarna menjadi grayscale untuk fokus pada intensitas cahaya tanpa warna.
    - Meningkatkan kecerahan agar gambar lebih terang dan detail pada area gelap lebih terlihat.
    - Meningkatkan kontras agar perbedaan antara area terang dan gelap lebih jelas, sehingga gambar lebih tajam dan mudah dianalisis.

BAB IV

PENUTUP

Praktikum ini bertujuan untuk memahami dasar-dasar pengolahan citra digital menggunakan Python, khususnya dalam membaca, menampilkan, dan memanipulasi gambar. Proses dimulai dengan membaca gambar berwarna dan menampilkannya menggunakan library matplotlib, yang merupakan langkah awal penting untuk visualisasi citra. Selanjutnya, gambar diubah menjadi grayscale untuk menghilangkan informasi warna dan fokus pada intensitas cahaya, yang sering digunakan dalam berbagai aplikasi pengolahan citra.

Setelah itu, dilakukan peningkatan kecerahan pada gambar grayscale menggunakan modul ImageEnhance dari PIL, yang membuat gambar menjadi lebih terang dan detail pada area gelap lebih terlihat. Langkah berikutnya adalah peningkatan kontras, yang memperjelas perbedaan antara area terang dan gelap sehingga gambar tampak lebih tajam dan mudah dianalisis. Kombinasi peningkatan kecerahan dan kontras ini sangat berguna untuk memperbaiki kualitas visual gambar sebelum dilakukan analisis lebih lanjut.

Secara keseluruhan, praktikum ini mengajarkan bagaimana memanfaatkan fungsi-fungsi dasar dari matplotlib dan PIL untuk melakukan manipulasi citra yang sederhana namun efektif. Teknik-teknik ini merupakan fondasi penting dalam pengolahan citra digital yang dapat diterapkan dalam berbagai bidang, seperti pengenalan pola, deteksi objek, dan peningkatan kualitas gambar.

DAFTAR PUSTAKA

1. Al-Ameen, W. M., & Sulong, G. B. (2021). Adaptive thresholding and color segmentation for object detection in digital images. *International Journal of Electrical and Computer Engineering (IJECE)*, 11(2), 1507–1514. https://doi.org/10.11591/ijece.v11i2.pp1507-1514
2. Sahu, P. K., & Majhi, B. (2020). Improved Histogram Equalization Techniques for Contrast Enhancement in Images: A Survey. *Journal of King Saud University - Computer and Information Sciences*. https://doi.org/10.1016/j.jksuci.2020.04.005
3. Chaudhuri, D., & Samanta, D. (2020). Color Image Segmentation Using HSV Color Space Based K-Means Clustering. *Procedia Computer Science*, 167, 1224–1231. https://doi.org/10.1016/j.procs.2020.03.460
4. Satpathy, S. K., & Rani, R. (2023). A novel image enhancement technique for poor lighting and backlight images using improved CLAHE. *Multimedia Tools and Applications*, 82, 11211–11230. https://doi.org/10.1007/s11042-023-14846-3
5. Al-Omari, A., Al-Tarawneh, M., & Al-Qudah, M. (2019). Color image segmentation using adaptive thresholding and morphological operations. *International Journal of Computer Applications*, 182(47), 12–17. https://doi.org/10.5120/ijca2019918371