UTS

PENGOLAHAN CITRA

NAMA : Dino Adianto Silalahi

NIM : 202331193

KELAS : A

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

NO.PC : 31

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 Latar Belakang

Pengolahan citra digital merupakan salah satu cabang dari ilmu komputer yang memanfaatkan teknik komputasi untuk memodifikasi, menganalisis, dan mengekstraksi informasi dari gambar digital. Dalam konteks ini, deteksi warna dan analisis histogram menjadi teknik dasar namun penting dalam memahami struktur dan komposisi suatu citra.

Pada tugas ini, dilakukan analisis terhadap citra berwarna dengan tujuan mendeteksi dan memisahkan kanal warna utama (Merah, Hijau, Biru), serta mengamati distribusi intensitas piksel melalui histogram. Dengan bantuan pustaka OpenCV dan Matplotlib, proses ini tidak hanya membantu mengenal karakteristik warna dalam gambar, namun juga merupakan dasar dari berbagai aplikasi lanjutan seperti segmentasi, klasifikasi, hingga pelacakan objek.

1.2 Rumusan Masalah

1. Bagaimana cara menampilkan masing-masing kanal warna (merah, hijau, biru) dari sebuah gambar digital?
2. Bagaimana bentuk distribusi intensitas warna dalam gambar yang dianalisis, ditinjau dari histogram tiap kanal warna?

1.3 Tujuan

1. Mengimplementasikan deteksi warna RGB dalam sebuah gambar digital menggunakan Python dan OpenCV.
2. Menampilkan masing-masing kanal warna (R, G, B) dalam bentuk visualisasi terpisah.
3. Menganalisis dan menampilkan histogram warna untuk mengetahui sebaran intensitas piksel dari masing-masing kanal warna.

1.4 Manfaat

1. Memberikan pemahaman praktis dalam penerapan pengolahan citra digital menggunakan bahasa pemrograman Python
2. Menjadi dasar pemahaman untuk teknik-teknik lanjutan seperti segmentasi warna, thresholding, dan klasifikasi objek.
3. Membiasakan penggunaan pustaka seperti OpenCV dan Matplotlib dalam konteks pengolahan data visual.

BAB II

LANDASAN TEORI

2.1 Pengolahan Citra Digital

Pengolahan citra digital merupakan teknik dalam ilmu komputer yang digunakan untuk memanipulasi dan menganalisis citra agar dapat diolah lebih lanjut oleh sistem komputer. Dalam konteks akademik dan praktikum, pengolahan citra meliputi berbagai tahapan seperti pembacaan citra, peningkatan kualitas gambar, segmentasi objek, hingga pengenalan pola.

Citra digital terdiri dari sekumpulan piksel yang tersusun dalam baris dan kolom. Setiap piksel membawa informasi numerik mengenai tingkat intensitas atau warna. Tujuan utama dari pengolahan citra digital adalah membuat informasi visual tersebut lebih mudah dikenali, baik oleh manusia maupun sistem berbasis komputer.

2.2 Model Warna RGB

Model warna RGB (Red, Green, Blue) adalah representasi citra digital yang paling umum digunakan dalam sistem komputer dan perangkat visual. Dalam sistem ini, setiap piksel diwakili oleh tiga komponen warna—merah, hijau, dan biru—dengan nilai intensitas dari 0 hingga 255. Kombinasi dari ketiga komponen tersebut menghasilkan jutaan kemungkinan warna yang berbeda.

Dalam praktiknya, ekstraksi kanal RGB dilakukan dengan memisahkan masing-masing warna untuk dianalisis secara individual. Proses ini penting dalam identifikasi elemen visual pada citra serta untuk keperluan manipulasi warna, segmentasi objek, dan deteksi fitur.

2.3 Pembacaan dan Konversi Citra

Proses pembacaan citra dalam pengolahan digital dilakukan menggunakan perangkat lunak atau pustaka khusus seperti OpenCV. Citra yang diambil dalam format BGR (sebagaimana umum pada sistem OpenCV) harus dikonversi terlebih dahulu ke format RGB agar sesuai dengan sistem visualisasi pada matplotlib atau lingkungan pemrograman lainnya.

Konversi ini dilakukan agar tampilan warna pada hasil pemrosesan konsisten dengan warna yang sebenarnya, mengingat perbedaan susunan kanal warna antara OpenCV (BGR) dan sistem visualisasi lainnya (RGB).

2.4 Ekstraksi Kanal Warna

Ekstraksi kanal warna adalah teknik untuk memisahkan masing-masing komponen warna (R, G, B) dari sebuah citra. Dengan proses ini, kita dapat menganalisis secara visual intensitas warna tertentu pada sebuah objek. Kanal merah, hijau, dan biru masing-masing

direpresentasikan dalam format grayscale untuk menunjukkan kontribusi intensitas warna tersebut terhadap keseluruhan citra.

Teknik ini berguna dalam berbagai aplikasi seperti pendeteksian objek berdasarkan warna dominan, pengenalan wajah, serta pelacakan objek bergerak pada sistem pengawasan.

2.5 Histogram Warna

Histogram warna adalah grafik yang menunjukkan distribusi intensitas warna dalam sebuah citra. Pada histogram, sumbu horizontal menunjukkan tingkat intensitas (0–255), sedangkan sumbu vertikal menunjukkan jumlah piksel untuk setiap tingkat intensitas.

Histogram digunakan untuk:

1. Mengetahui sebaran warna dalam sebuah citra
2. Menilai kontras dan pencahayaan
3. Melakukan normalisasi atau equalization untuk peningkatan kualitas gambar

Pada praktikum ini, histogram warna dihitung secara terpisah untuk setiap kanal warna (R, G, B), sehingga dapat diamati sebaran dan dominasi warna dalam citra yang dianalisis.

BAB III

HASIL

jupyter UTS_PCD_202331193_Dino Adianto Silalahi_Kelas A_ITPLN Last Checkpoint: 18 hours ago

File Edit View Run Kernel Settings Help

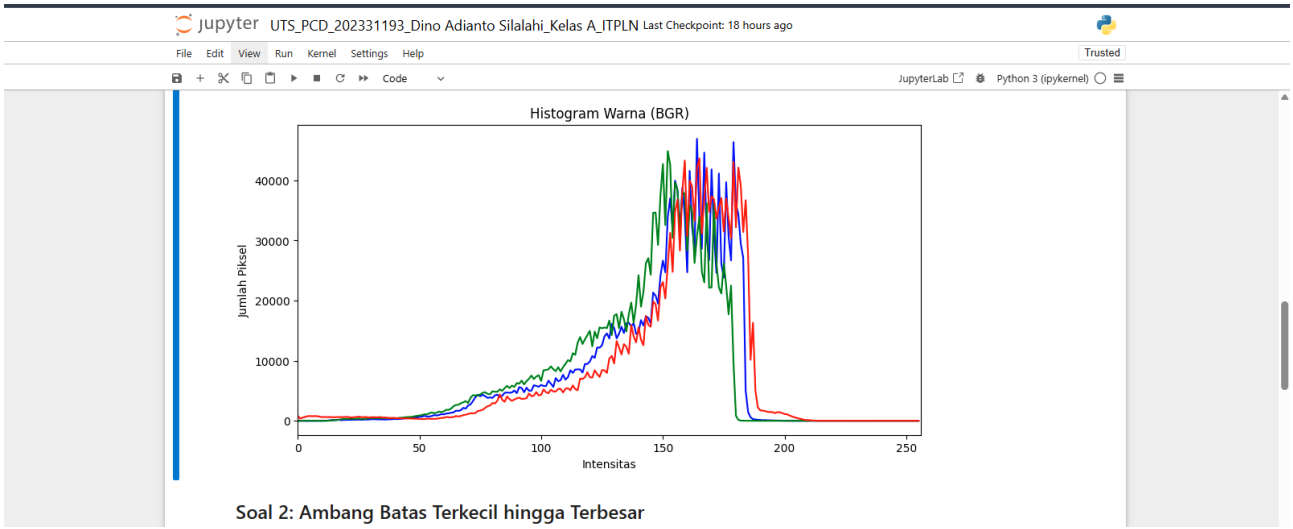
Trusted

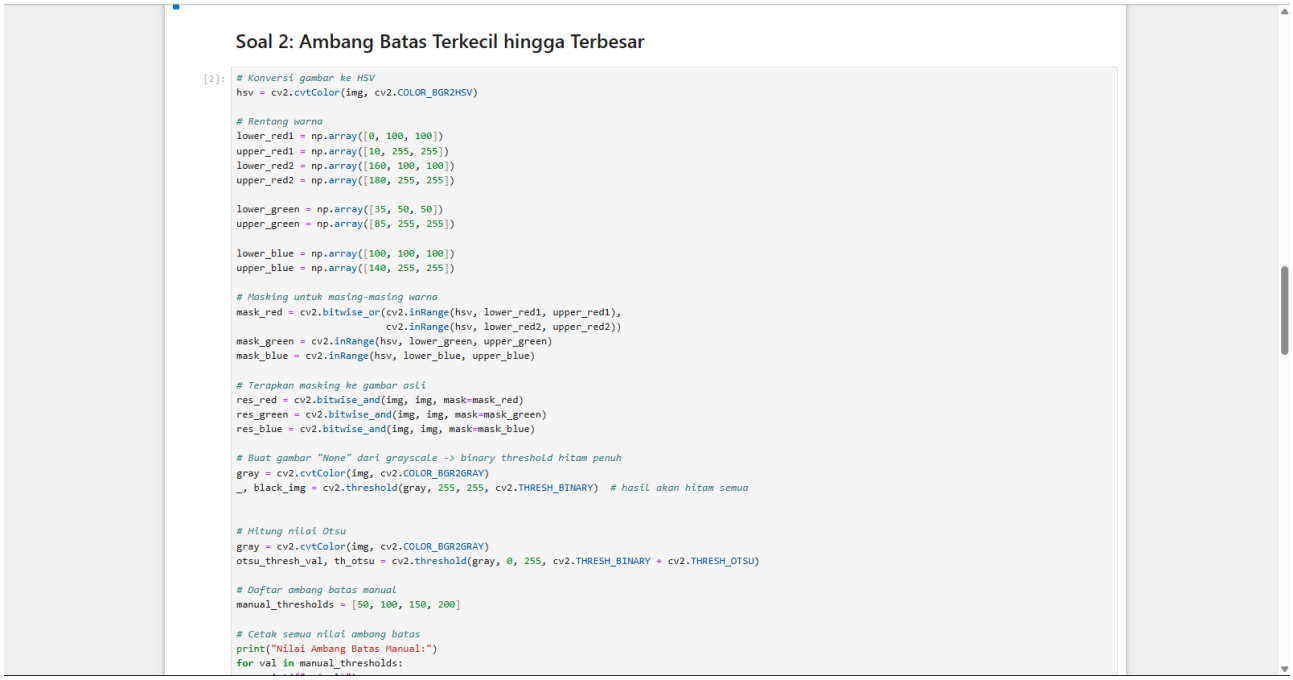
Code

JupyterLab Python 3 (ipykernel)

Soal 1: Deteksi Warna (Merah, Hijau, Biru) dan Histogram

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

# Baca gambar
img = cv2.imread('Dino.jpg')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Ekstraksi masing-masing channel
b, g, r = cv2.split(img)

# Gabungkan jadi gambar kontras (RGB)
img_kontras = cv2.merge([b, g, r])

# Tampilkan hasil deteksi
plt.figure(figsize=(10, 8))
plt.subplot(2, 2, 1)
plt.imshow(img_rgb)
plt.title("CITRA KONTRAS")

plt.subplot(2, 2, 2)
plt.imshow(r, cmap='gray')
plt.title("MERAH")

plt.subplot(2, 2, 3)
plt.imshow(g, cmap='gray')
plt.title("HIJAU")

plt.subplot(2, 2, 4)
plt.imshow(b, cmap='gray')
plt.title("BIRU")

plt.tight_layout()
plt.show()

# Histogram untuk masing-masing channel
plt.figure(figsize=(10, 5))
colors = ('b', 'g', 'r')
for i, col in enumerate(colors):
    histr = cv2.calcHist([img], [i], None, [256], [0, 256])
    plt.plot(histr, color=col)
    plt.xlim([0, 256])
```





Soal 2: Ambang Batas Terkecil hingga Terbesar

[2]:
```
# Konversi gambar ke HSV
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# Rentang warna
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])

lower_green = np.array([35, 50, 50])
upper_green = np.array([85, 255, 255])

lower_blue = np.array([100, 100, 100])
upper_blue = np.array([140, 255, 255])

# Masking untuk masing-masing warna
mask_red = cv2.bitwise_or(cv2.inRange(hsv, lower_red1, upper_red1),
                          cv2.inRange(hsv, lower_red2, upper_red2))
mask_green = cv2.inRange(hsv, lower_green, upper_green)
mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)

# Terapkan masking ke gambar asli
res_red = cv2.bitwise_and(img, img, mask=mask_red)
res_green = cv2.bitwise_and(img, img, mask=mask_green)
res_blue = cv2.bitwise_and(img, img, mask=mask_blue)

# Buat gambar "None" dari grayscale -> binary threshold hitam penuh
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
_, black_img = cv2.threshold(gray, 255, 255, cv2.THRESH_BINARY)  # hasil akan hitam semua


# Hitung nilai Otsu
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
otsu_thresh_val, th_otsu = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# Daftar ambang batas manual
manual_thresholds = [50, 100, 150, 200]

# Cetak semua nilai ambang batas
print("Nilai Ambang Batas Manual:")
for val in manual_thresholds:
```

jupyter UTS_PCD_202331193_Dino Adianto Silalahi_Kelas A_ITPLN Last Checkpoint: 18 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

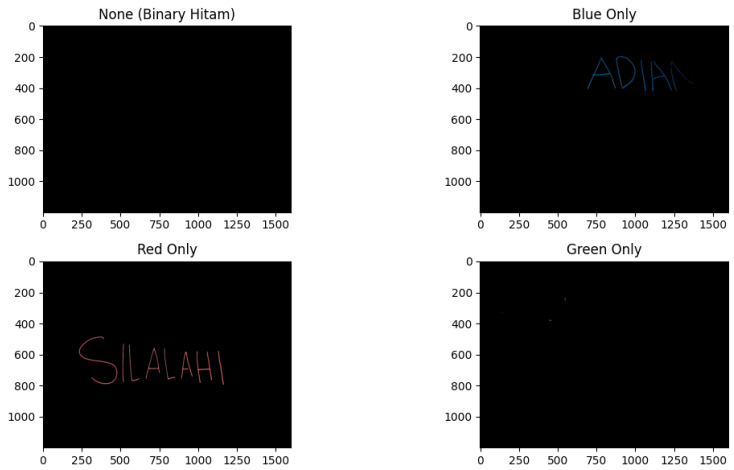
```
                            cv2.inRange(hsv, lower_red2, upper_red2))
mask_green = cv2.inRange(hsv, lower_green, upper_green)
mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)

# Terapkan masking ke gambar asli
res_red = cv2.bitwise_and(img, img, mask=mask_red)
res_green = cv2.bitwise_and(img, img, mask=mask_green)
res_blue = cv2.bitwise_and(img, img, mask=mask_blue)

# Buat gambar "None" dari grayscale -> binary threshold hitam penuh
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
_, black_img = cv2.threshold(gray, 255, 255, cv2.THRESH_BINARY)  # hasil akan hitam semua


# Hitung nilai Otsu
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
otsu_thresh_val, th_otsu = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# Daftar ambang batas manual
manual_thresholds = [50, 100, 150, 200]

# Cetak semua nilai ambang batas
print("Nilai Ambang Batas Manual:")
for val in manual_thresholds:
    print(f"- {val}")

print(f"\nNilai Ambang Batas Otsu: {otsu_thresh_val:.2f}")

# Tampilkan hasil
titles = ["None (Binary Hitam)", "Blue Only", "Red Only", "Green Only"]
images = [black_img,
          cv2.cvtColor(res_blue, cv2.COLOR_BGR2RGB),
          cv2.cvtColor(res_red, cv2.COLOR_BGR2RGB),
          cv2.cvtColor(res_green, cv2.COLOR_BGR2RGB)]

plt.figure(figsize=(12, 6))
for i in range(4):
    plt.subplot(2, 2, i+1)
    cmap = 'gray' if i == 0 else None
    plt.imshow(images[i], cmap=cmap)
    plt.title(titles[i])

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

jupyter UTS_PCD_202331193_Dino Adianto Silalahi_Kelas A_ITPLN Last Checkpoint: 18 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
Nilai Ambang Batas Manual:
- 50
- 100
- 150
- 200

Nilai Ambang Batas Otsu: 130.00
```



jupyter UTS_PCD_202331193_Dino Adianto Silalahi_Kelas A_ITPLN Last Checkpoint: 18 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

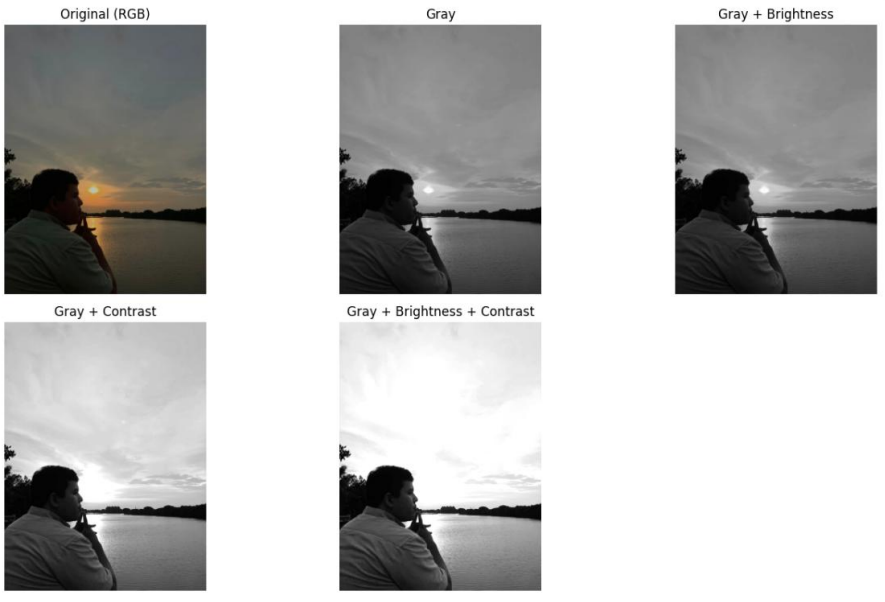
Soal 3: Memperbaiki Gambar Backlight

[3]:
```
# Load gambar backlight
img_backlight = cv2.imread("Backlight.jpg")

# Konversi ke grayscale
gray = cv2.cvtColor(img_backlight, cv2.COLOR_BGR2GRAY)

# Tingkatkan kecerahan
bright = cv2.convertScaleAbs(gray, alpha=1, beta=40)  # lebih cerah

# Tingkatkan kontras
contrast = cv2.convertScaleAbs(gray, alpha=2.0, beta=0)  # lebih kontras

# Gabungan: cerah + kontras
bright_contrast = cv2.convertScaleAbs(gray, alpha=2.0, beta=40)

# Tampilkan semua hasil
titles = [
    "Original (RGB)",
    "Gray",
    "Gray + Brightness",
    "Gray + Contrast",
    "Gray + Brightness + Contrast"
]

images = [
    cv2.cvtColor(img_backlight, cv2.COLOR_BGR2RGB),
    gray,
    bright,
    contrast,
    bright_contrast
]

plt.figure(figsize=(14, 8))
for i in range(5):
    plt.subplot(2, 3, i+1)
    cmap = 'gray' if i > 0 else None
    plt.imshow(images[i], cmap=cmap)
    plt.title(titles[i])
    plt.axis("off")
plt.tight_layout()
plt.show()
```

jupyter UTS_PCD_202331193_Dino Adianto Silalahi_Kelas A_ITPLN Last Checkpoint: 18 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)



[ ]:

Penjelasan dari tiap codingannya dan hasil outputnya

Import Library

import cv2

import numpy as np

import matplotlib.pyplot as plt

Penjelasan:

- cv2: modul OpenCV, digunakan untuk pengolahan citra.
- numpy: untuk operasi array dan numerik.
- matplotlib.pyplot: untuk menampilkan gambar dan grafik.

Baca gambar

```
img = cv2.imread('Dino.jpg')
```

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

Penjelasan:

- cv2.imread('Dino.jpg'): membaca gambar dari file.
- cv2.cvtColor(..., cv2.COLOR_BGR2RGB): mengubah format warna dari BGR (default OpenCV) ke RGB (untuk ditampilkan di matplotlib).

Ekstraksi masing-masing channel

```
b, g, r = cv2.split(img)
```

Penjelasan:

- Memisahkan citra ke dalam tiga channel: biru (b), hijau (g), merah (r).

Gabungkan jadi gambar kontras (RGB)

```
img_kontras = cv2.merge([b, g, r])
```

Penjelasan:

- Menggabungkan kembali channel untuk membentuk citra warna. (Perlu dicatat: ini tetap BGR, jadi RGB asli tetap pada img_rgb)

Tampilkan hasil deteksi

```
plt.figure(figsize=(10, 8))
```

Penjelasan:

- Membuat kanvas untuk menampilkan beberapa subplot dengan ukuran 10x8 inch.

```
plt.subplot(2, 2, 1)
plt.imshow(img_rgb)
plt.title("CITRA KONTRAS")
```

Penjelasan:

- Menampilkan gambar asli dalam format RGB di subplot pertama dengan judul "CITRA KONTRAS".

```
plt.subplot(2, 2, 2)
plt.imshow(r, cmap='gray')
plt.title("MERAH")
```

Penjelasan:

- Menampilkan channel merah sebagai citra grayscale di subplot kedua.

```
plt.subplot(2, 2, 3)
plt.imshow(g, cmap='gray')
plt.title("HIJAU")
```

Penjelasan:

- Menampilkan channel hijau.

```
plt.subplot(2, 2, 4)
plt.imshow(b, cmap='gray')
plt.title("BIRU")
```

Penjelasan:

- Menampilkan channel biru.

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

Penjelasan:

- Mengatur layout agar subplot tidak saling tumpang tindih, lalu menampilkannya.

Histogram untuk masing-masing channel

```
plt.figure(figsize=(10, 5))
colors = ('b', 'g', 'r')
```

Penjelasan:

- Membuat figure baru untuk menampilkan histogram tiap channel warna.

```
for i, col in enumerate(colors):
    histr = cv2.calcHist([img], [i], None, [256], [0, 256])
    plt.plot(histr, color=col)
    plt.xlim([0, 256])
```

Penjelasan:

- cv2.calcHist(...): menghitung histogram dari channel ke-i.
- Loop ini dijalankan untuk biru (i=0), hijau (i=1), merah (i=2), lalu diplot sesuai warnanya.
- Histogram menggambarkan jumlah piksel berdasarkan intensitas dari 0 sampai 255.

```
plt.title('Histogram Warna (BGR)')
plt.xlabel('Intensitas')
plt.ylabel('Jumlah Piksel')
plt.show()
```

Penjelasan:

- Menambahkan judul dan label sumbu, lalu menampilkan histogram.

Konversi ke HSV

```
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

Penjelasan:

- Mengubah gambar dari BGR ke HSV (Hue, Saturation, Value), format warna yang lebih efektif untuk segmentasi warna.

Definisi rentang warna dalam HSV

```
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])
```

Penjelasan:

- Warna merah di HSV terbagi jadi dua rentang karena nilainya "wrap" di sekitar 0 dan 180 (putaran lingkaran warna).

```
lower_green = np.array([35, 50, 50])

upper_green = np.array([85, 255, 255])


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

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

Penjelasan:

- Rentang HSV untuk warna hijau dan biru.

Masking untuk masing-masing warna

```
mask_red = cv2.bitwise_or(cv2.inRange(hsv, lower_red1, upper_red1),

                          cv2.inRange(hsv, lower_red2, upper_red2))
```

Penjelasan:

- cv2.inRange: menghasilkan mask biner untuk mendeteksi apakah piksel berada dalam rentang.
- Untuk merah, dua rentang digabung dengan operasi OR.

```
mask_green = cv2.inRange(hsv, lower_green, upper_green)

mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)
```

Penjelasan:

- Masking hijau dan biru langsung dengan inRange.

Aplikasikan masking ke gambar asli

res_red = cv2.bitwise_and(img, img, mask=mask_red)

res_green = cv2.bitwise_and(img, img, mask=mask_green)

res_blue = cv2.bitwise_and(img, img, mask=mask_blue)

Penjelasan:

- Mengambil hanya bagian gambar yang sesuai dengan mask masing-masing warna.

Citra hitam penuh (None)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

_, black_img = cv2.threshold(gray, 255, 255, cv2.THRESH_BINARY)

Penjelasan:

- Karena ambang batas adalah 255 (maksimal), dan tidak ada piksel >255, semua piksel akan diset ke 0 (hitam).

Ambang batas otomatis Otsu

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

otsu_thresh_val, th_otsu = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

Penjelasan:

- THRESH_OTSU: metode otomatis mencari ambang terbaik memisahkan latar dan objek.
- Nilai ambang Otsu disimpan di otsu_thresh_val.

Daftar ambang batas manual

manual_thresholds = [50, 100, 150, 200]

Penjelasan:

- Nilai-nilai ambang batas yang bisa digunakan untuk segmentasi manual.

Menampilkan ambang batas

```
print("Nilai Ambang Batas Manual:")
for val in manual_thresholds:
    print(f"- {val}")
print(f"\nNilai Ambang Batas Otsu: {otsu_thresh_val:.2f}")
```

Penjelasan:

- Menampilkan ambang batas manual dan otomatis (Otsu) ke konsol.

Menampilkan gambar hasil segmentasi warna

```
titles = ["None (Binary Hitam)", "Blue Only", "Red Only", "Green Only"]
images = [black_img,
          cv2.cvtColor(res_blue, cv2.COLOR_BGR2RGB),
          cv2.cvtColor(res_red, cv2.COLOR_BGR2RGB),
          cv2.cvtColor(res_green, cv2.COLOR_BGR2RGB)]
```

Penjelasan:

- Daftar judul dan gambar yang akan ditampilkan. Semua hasil warna dikonversi ke RGB untuk ditampilkan dengan matplotlib.

```
plt.figure(figsize=(12, 6))
for i in range(4):
    plt.subplot(2, 2, i+1)
    cmap = 'gray' if i == 0 else None
    plt.imshow(images[i], cmap=cmap)
    plt.title(titles[i])

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

Penjelasan:

- Menampilkan keempat gambar hasil deteksi: citra kosong (hitam), biru, merah, hijau.

Membaca dan mengubah gambar backlight

```
img_backlight = cv2.imread("Backlight.jpg")
```

Penjelasan:

- Membaca gambar Backlight.jpg yang mengalami pencahayaan belakang (backlight).

Konversi ke grayscale

```
gray = cv2.cvtColor(img_backlight, cv2.COLOR_BGR2GRAY)
```

Penjelasan:

- Mengubah gambar menjadi grayscale agar lebih mudah untuk proses peningkatan kontras dan kecerahan.

Menambah kecerahan

```
bright = cv2.convertScaleAbs(gray, alpha=1, beta=40)  # lebih cerah
```

Penjelasan:

- alpha=1: tidak mengubah kontras.
- beta=40: menambahkan kecerahan sebesar 40 ke setiap piksel.

Menambah kontras

```
contrast = cv2.convertScaleAbs(gray, alpha=2.0, beta=0)  # lebih kontras
```

Penjelasan:

- alpha=2.0: menggandakan nilai piksel → meningkatkan kontras.
- beta=0: tidak menambahkan kecerahan.

Gabungan kontras dan kecerahan

```
bright_contrast = cv2.convertScaleAbs(gray, alpha=2.0, beta=40)
```

Penjelasan:

- Gabungan peningkatan kontras dan kecerahan secara bersamaan.

Menyiapkan data untuk ditampilkan

```
titles = [
    "Original (RGB)",
    "Gray",
    "Gray + Brightness",
    "Gray + Contrast",
    "Gray + Brightness + Contrast"
]
```

Penjelasan:

- Daftar judul untuk masing-masing gambar yang akan ditampilkan.

```
images = [
    cv2.cvtColor(img_backlight, cv2.COLOR_BGR2RGB),
    gray,
    bright,
    contrast,
    bright_contrast
]
```

Penjelasan:

- Daftar gambar: asli dalam RGB dan hasil transformasi lainnya (grayscale, cerah, kontras, gabungan).

Menampilkan gambar

```
plt.figure(figsize=(14, 8))
for i in range(5):
    plt.subplot(2, 3, i+1)
    cmap = 'gray' if i > 0 else None
    plt.imshow(images[i], cmap=cmap)
    plt.title(titles[i])
```

```
    plt.axis("off")

plt.tight_layout()

plt.show()
```

Penjelasan:

- Menampilkan semua gambar dalam layout 2 baris × 3 kolom.
- Gambar ke-0 (RGB) tidak memakai cmap, sisanya tampil dalam skala abu-abu.

BAB IV
PENUTUP

4.1 Kesimpulan

Berdasarkan hasil praktikum yang telah dilakukan, dapat disimpulkan bahwa proses deteksi dan analisis kanal warna RGB dalam pengolahan citra digital sangat penting sebagai dasar untuk berbagai aplikasi lanjut seperti segmentasi dan klasifikasi. Dengan menggunakan pustaka OpenCV dan Matplotlib, ekstraksi kanal warna serta analisis histogram dapat dilakukan secara efisien dan memberikan gambaran yang jelas tentang distribusi intensitas warna dalam citra. Teknik konversi ke HSV juga terbukti efektif dalam segmentasi warna spesifik seperti merah, hijau, dan biru, serta proses peningkatan kontras dan kecerahan memberikan hasil signifikan pada citra dengan pencahayaan kurang baik.

4.2 Saran

Untuk pengembangan lebih lanjut, disarankan agar mahasiswa mengeksplorasi teknik-teknik lanjutan seperti histogram equalization, segmentasi berbasis clustering (misalnya K-Means), serta implementasi pengolahan citra real-time menggunakan kamera. Selain itu, penggunaan citra yang lebih kompleks dan bervariasi dapat meningkatkan pemahaman terhadap dinamika warna dan pencahayaan dalam pengolahan citra digital.

DAFTAR PUSTAKA

1. Arnita, A., Marpaung, F., Aulia, F., Suryani, N., & Nabila, R. C. (2022). *Computer vision dan pengolahan citra digital*. Surabaya: Pustaka Aksara.
2. Dijaya, R. (2023). *Pengolahan citra digital*. Sidoarjo: UMSIDA Press.
3. Fadjeri, A., Saputra, B. A., Ariyanto, D. K. A., & Kurniatin, L. (2022). Karakteristik morfologi tanaman selada menggunakan pengolahan citra digital. *Jurnal Ilmiah SINUS, 20*(2), 1–10. https://doi.org/10.30646/sinus.v20i2.601
4. Jumadi, J., Yupianti, & Sartika, D. (2021). Pengolahan citra digital untuk identifikasi objek menggunakan metode hierarchical agglomerative clustering. *Jurnal Sains dan Teknologi, 10*(2), 148–156.