UTS

PENGOLAHAN CITRA

NAMA : Alya Az-Zahra Maulana

NIM : 202331300

KELAS : D

DOSEN : Darma Rusjdi., Ir., M.Kom

NO.PC : -

ASISTEN : 1. Fakhrul Fauzi Nugraha Tarigan

2. Muhammad Hanief Febriansyah

3. Clarenca Sweetdiva Pereira

4. Sakura Amastasya Salsabila Setiyanto

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2024/2025

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

Bagaimana cara mendeteksi objek berdasarkan warna merah, hijau, dan biru menggunakan ruang warna HSV?.

1.2 Tujuan Masalah

Bagaimana histogram HSV dapat digunakan untuk menganalisis sebaran warna dalam citra digital?.

1.3 Manfaat Masalah

Bagaimana cara menampilkan hasil deteksi warna dalam bentuk visual yang informatif?

BAB II
LANDASAN TEORI

Dalam bidang pengolahan citra digital dan tampilan visual, pencahayaan menjadi komponen penting, terutama pada teknologi layar. Salah satu teknologi utama yang digunakan untuk mendukung tampilan visual adalah backlight. Backlight atau pencahayaan belakang merupakan sumber cahaya yang diposisikan di belakang panel tampilan, seperti layar LCD, yang berfungsi untuk menerangi piksel karena LCD tidak dapat menghasilkan cahaya sendiri. Tanpa adanya backlight, tampilan LCD akan tampak gelap dan tidak dapat memperlihatkan gambar secara jelas. Backlight biasanya menggunakan teknologi LED (Light Emitting Diode), baik dalam bentuk White LED (WLED) maupun RGB LED, untuk menghasilkan spektrum warna yang lebih luas dan akurat.

Dalam konteks pengolahan citra, salah satu teknik dasar yang umum digunakan untuk mengekstraksi informasi dari gambar adalah teknik thresholding atau teknik ambang batas. Teknik ini digunakan untuk memisahkan objek dari latar belakang pada citra skala abu-abu dengan mengubahnya menjadi citra biner. Piksel yang memiliki nilai intensitas di atas nilai ambang akan diklasifikasikan sebagai objek (biasanya diberi warna putih), sementara piksel yang berada di bawah ambang akan diklasifikasikan sebagai latar belakang (biasanya hitam). Thresholding dapat dibagi menjadi dua jenis utama, yaitu global thresholding yang menggunakan satu nilai ambang untuk seluruh citra, serta adaptive thresholding yang menyesuaikan nilai ambang berdasarkan wilayah lokal dalam citra. Salah satu metode thresholding yang terkenal adalah metode Otsu, yaitu metode otomatis yang menentukan nilai ambang optimal dengan memaksimalkan varians antar kelas dalam histogram citra.

Selanjutnya, dalam berbagai aplikasi pengolahan citra seperti pelacakan objek, system pengenalan warna, dan klasifikasi objek industri, digunakan teknik deteksi warna. Deteksi warna merupakan proses identifikasi piksel dalam citra berdasarkan nilai warnanya. Umumnya, citra dikonversi dari ruang warna RGB ke ruang warna lain yang lebih stabil terhadap pencahayaan, seperti HSV (Hue, Saturation, Value). Proses deteksi warna melibatkan penetapan rentang nilai warna tertentu untuk mengekstrak objek dengan warna spesifik dari latar belakang. Ruang warna HSV sangat populer karena lebih dekat dengan

persepsi manusia terhadap warna dan memisahkan informasi warna dari kecerahan, sehingga lebih efektif dalam kondisi pencahayaan yang berubah-ubah.

BAB III
HASIL

BACKLIGHT

[14]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

Kode tersebut untuk membaca gambar, mengolahnya, lalu menampilkan hasil pengolahan tersebut secara visual dalam program Python.

[15]:
```
img = cv2.imread('Foto.jpg')
if img is None:
    print("Gambar tidak ditemukan.")
else:
    print("Gambar berhasil dimuat.")
```

```
Gambar berhasil dimuat.
```

Membaca gambar dari file yang bernama Foto.jpg untuk siap diproses.
Kalau ternyata file gambar tidak ditemukan, program akan memberi peringatan lewat tulisan "Gambar tidak ditemukan". Tapi kalau berhasil dibuka, akan muncul tulisan "Gambar berhasil dimuat".

[18]:
```
# Konversi ke grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

mengubah gambar berwarna jadi grayscale (abu-abu saja).

```
# Manual brightness adjustment: gray * alpha + beta
alpha = 1.2
beta = 50
```

Mengatur kecerahann gambar dengan menggunakan alpha dan beta.

```
bright = np.zeros_like(gray)
rows, cols = gray.shape
for i in range(rows):
    for j in range(cols):
        new_val = int(alpha * gray[i, j] + beta)
        bright[i, j] = np.clip(new_val, 0, 255)
```

meningkatkan kecerahan dan kontras gambar grayscale dengan cara memodifikasi nilai setiap piksel, lalu menyimpannya di gambar baru (bright).

```
hist = np.zeros(256, dtype=int)

# Hitung histogram
for i in range(rows):
    for j in range(cols):
        hist[bright[i, j]] += 1
```

Menggunakan histogram untuk mengetahui berapa banyak piksel yang punya nilai keabuan tertentu. Jadi, kita hitung berapa kali tiap nilai (0–255) muncul dalam gambar yang sudah dicerahkan.

```
cdf = hist.cumsum()
cdf_normalized = cdf * 255 / cdf[-1]  # Normalisasi ke rentang 0-255
```

CDF (Cumulative Distribution Function) untuk menghitung jumlah kumulatif dari histogram. Kemudian hasilnya disesuaikan agar nilai-nilainya tetap dalam rentang 0–255.

```
enhanced = np.zeros_like(bright)
for i in range(rows):
    for j in range(cols):
        enhanced[i, j] = int(cdf_normalized[bright[i, j]])
```

meningkatkan kualitas kontras gambar menggunakan histogram equalization secara manual.

AMBANG BATAS

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

Kode tersebut untuk membaca gambar, mengolahnya, lalu menampilkan hasil pengolahan tersebut secara visual dalam program Python.

[9]:
```
img = cv2.imread('Nama.jpg')  # Ganti dengan path gambar kamu
if img is None:
    print("Citra tidak dapat dimuat. Pastikan path gambar benar.")
else:
    print("Citra berhasil dimuat.")
```

```
Citra berhasil dimuat.
```

Membaca gambar dari file yang bernama Foto.jpg untuk siap diproses.
Kalau ternyata file gambar tidak ditemukan, program akan memberi peringatan lewat tulisan "Gambar tidak ditemukan". Tapi kalau berhasil dibuka, akan muncul tulisan "Gambar berhasil dimuat".

[10]:
```
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

Mengubah format warna dari BGR (standar OpenCV) ke HSV.

[11]:
```
# Format: [H_min, S_min, V_min], [H_max, S_max, V_max]
batas_merah = ([0, 100, 100], [10, 255, 255])         # Rentang pertama merah
batas_merah2 = ([160, 100, 100], [179, 255, 255])     # Rentang kedua merah (karena merah wrap di HSV)
batas_hijau = ([40, 50, 50], [80, 255, 255])
batas_biru = ([100, 100, 100], [130, 255, 255])
```

Menetapkan nilai ambang bawah dan atas untuk masing-masing warna.

```
# Konversi ke np.array
lower_red1 = np.array(batas_merah[0])
upper_red1 = np.array(batas_merah[1])
lower_red2 = np.array(batas_merah2[0])
upper_red2 = np.array(batas_merah2[1])
lower_green = np.array(batas_hijau[0])
upper_green = np.array(batas_hijau[1])
lower_blue = np.array(batas_biru[0])
upper_blue = np.array(batas_biru[1])

# Masking warna
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)

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

Menandai piksel yang masuk dalam rentang warna (jadi citra hitam-putih).

```
[12]: # Visualisasi dengan plt
      fig, axs = plt.subplots(1, 4, figsize=(16, 5))
```

Membuat 1 baris dengan 4 kolom plot (subplot) untuk menampilkan 4 gambar.

```
axs[1].imshow(mask_red, cmap='gray')
axs[1].set_title('Deteksi Merah')
axs[1].axis('off')

axs[2].imshow(mask_green, cmap='gray')
axs[2].set_title('Deteksi Hijau')
axs[2].axis('off')

axs[3].imshow(mask_blue, cmap='gray')
axs[3].set_title('Deteksi Biru')
axs[3].axis('off')

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

Menampilkan hasil deteksi warna.

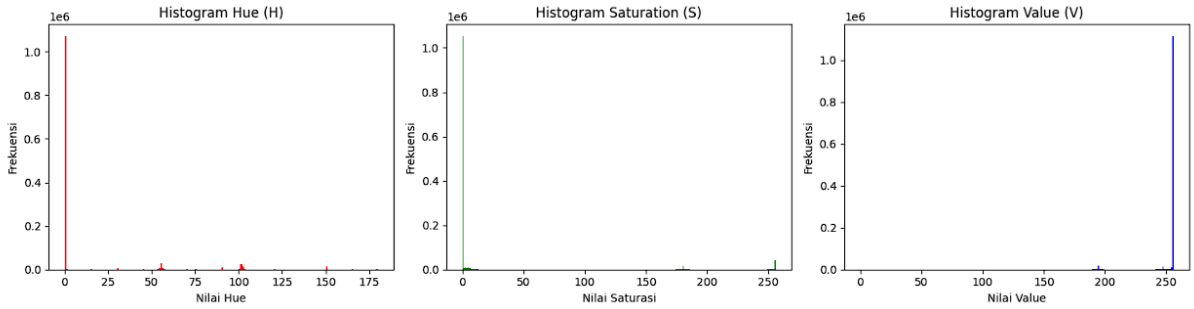
[13]:
```
print("Ambang batas warna (HSV):")
ambang_list = [
    ("Merah 1", lower_red1.tolist(), upper_red1.tolist()),
    ("Merah 2", lower_red2.tolist(), upper_red2.tolist()),
    ("Hijau", lower_green.tolist(), upper_green.tolist()),
    ("Biru", lower_blue.tolist(), upper_blue.tolist())
]

ambang_sorted = sorted(ambang_list, key=lambda x: x[1][0])

for nama, low, up in ambang_sorted:
    print(f"{nama}: Lower = {low}, Upper = {up}")
```

```
Ambang batas warna (HSV):
Merah 1: Lower = [0, 100, 100], Upper = [10, 255, 255]
Hijau: Lower = [40, 50, 50], Upper = [80, 255, 255]
Biru: Lower = [100, 100, 100], Upper = [130, 255, 255]
Merah 2: Lower = [160, 100, 100], Upper = [179, 255, 255]
```

Menampilkan nilai ambang batas.

[14]:
```
hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

hue, sat, val = cv2.split(hsv_img)

plt.figure(figsize=(15, 4))

plt.subplot(1, 3, 1)
plt.hist(hue.ravel(), bins=180, range=(0, 180), color='red')
plt.title('Histogram Hue (H)')
plt.xlabel('Nilai Hue')
plt.ylabel('Frekuensi')

plt.subplot(1, 3, 2)
plt.hist(sat.ravel(), bins=256, range=(0, 256), color='green')
plt.title('Histogram Saturation (S)')
plt.xlabel('Nilai Saturasi')
plt.ylabel('Frekuensi')

plt.subplot(1, 3, 3)
plt.hist(val.ravel(), bins=256, range=(0, 256), color='blue')
plt.title('Histogram Value (V)')
plt.xlabel('Nilai Value')
plt.ylabel('Frekuensi')

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

Menampilkan histogram untuk masing-masing channel HSV.



DETEKSI WARNA

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

Kode tersebut untuk membaca gambar, mengolahnya, lalu menampilkan hasil pengolahan tersebut secara visual dalam program Python.

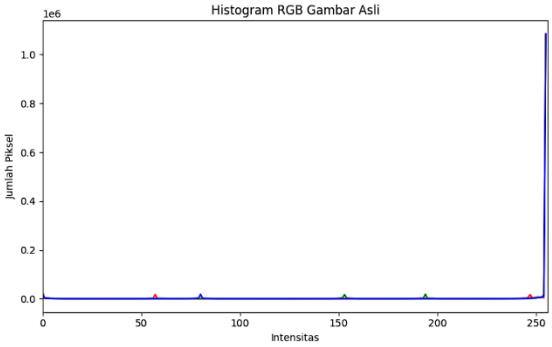
[2]:
```
img = cv2.imread('Nama.jpg')
if img is None:
    print("Citra tidak dapat dimuat. Pastikan path gambar benar.")
else:
    print("Citra berhasil dimuat.")
```

```
Citra berhasil dimuat.
```

Membaca gambar dari file yang bernama Foto.jpg untuk siap diproses.
Kalau ternyata file gambar tidak ditemukan, program akan memberi peringatan lewat tulisan "Gambar tidak ditemukan". Tapi kalau berhasil dibuka, akan muncul tulisan "Gambar berhasil dimuat".

[3]:
```
output_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

fig, axis = plt.subplots(1, 2, figsize=(15, 5))

axis[0].imshow(output_rgb)
axis[0].set_title("Gambar Asli")
axis[0].axis('off')

colors = ('r', 'g', 'b')
for i, col in enumerate(colors):
    hist = cv2.calcHist([img], [i], None, [256], [0, 256])
    axis[1].plot(hist, color=col)
axis[1].set_xlim([0, 256])
axis[1].set_title("Histogram RGB Gambar Asli")
axis[1].set_xlabel("Intensitas")
axis[1].set_ylabel("Jumlah Piksel")

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

Menampilkan gambar asli dan histogram RGB.



[4]:
```
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

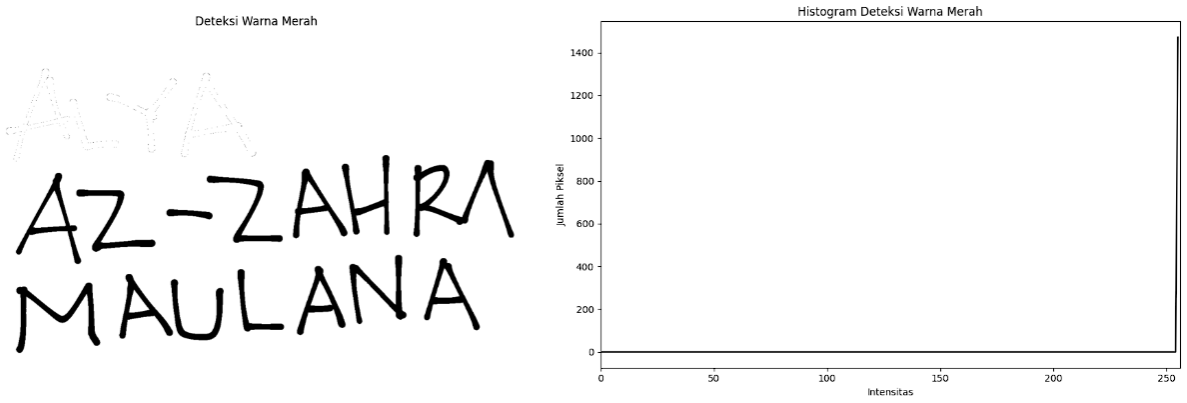
Mengubah format warna dari BGR (standar OpenCV) ke HSV.

[6]:
```
# Mask warna merah
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.inRange(hsv, lower_red1, upper_red1) | cv2.inRange(hsv, lower_red2, upper_red2)

# Mask warna hijau
lower_green = np.array([35, 100, 100])
upper_green = np.array([85, 255, 255])
mask_green = cv2.inRange(hsv, lower_green, upper_green)

# Mask warna biru
lower_blue = np.array([100, 100, 100])
upper_blue = np.array([140, 255, 255])
mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)

# Mask background putih (biar tetap putih)
lower_white = np.array([0, 0, 200])
upper_white = np.array([180, 40, 255])
mask_white = cv2.inRange(hsv, lower_white, upper_white)

# Menggabungkan mask warna target + background putih
red_result = cv2.bitwise_or(mask_red, mask_white)
green_result = cv2.bitwise_or(mask_green, mask_white)
blue_result = cv2.bitwise_or(mask_blue, mask_white)
```

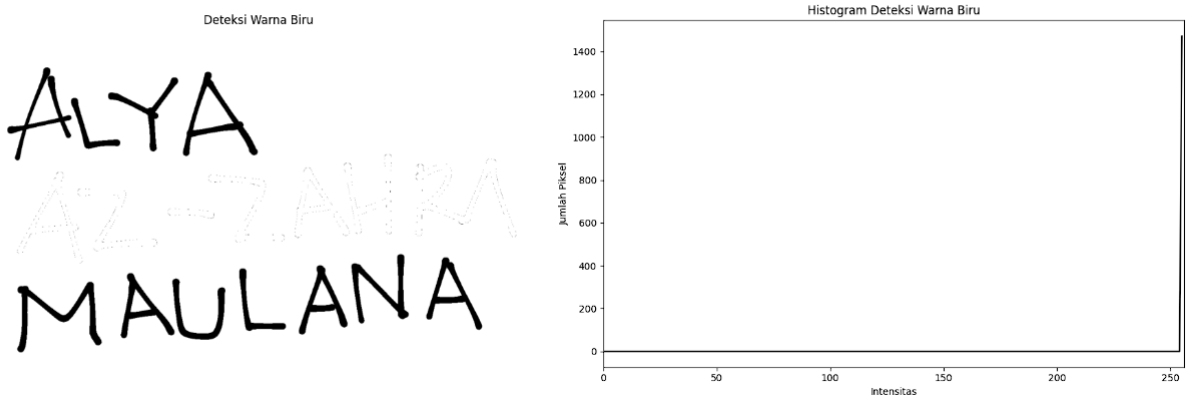
Menandai piksel yang masuk dalam rentang warna.

```
7]: def show_result_and_hist(image_mask, title):
        fig, axes = plt.subplots(1, 2, figsize=(18, 6))

        axes[0].imshow(image_mask, cmap='gray')
        axes[0].set_title(f"{title}")
        axes[0].axis('off')

        hist = cv2.calcHist(image_mask, [0], None, [256], [0, 256])
        axes[1].plot(hist, color='black')
        axes[1].set_title(f"Histogram {title}")
        axes[1].set_xlabel("Intensitas")
        axes[1].set_ylabel("Jumlah Piksel")
        axes[1].set_xlim([0, 256])

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

```
[8]: show_result_and_hist(red_result, "Deteksi Warna Merah")
```




```
[9]: show_result_and_hist(blue_result, "Deteksi Warna Biru")
```



Menampilkan hasil deteksi warna.

BAB IV
PENUTUP

Disimpulkan bahwa konversi citra dari format BGR ke HSV sangat membantu dalam proses deteksi warna, karena pada ruang warna HSV informasi tentang warna (Hue) dipisahkan dari pencahayaan (Value). Dengan penetapan nilai ambang HSV yang tepat, objek berwarna merah, hijau, dan biru dapat dideteksi dengan baik menggunakan fungsi cv2.inRange pada OpenCV. Visualisasi hasil deteksi dalam bentuk citra dan histogram memberikan gambaran yang jelas mengenai area objek yang teridentifikasi serta distribusi warna dalam gambar. Histogram HSV juga sangat berguna untuk mengetahui penyebaran warna dan intensitas pada sebuah gambar, sehingga bisa dijadikan dasar untuk pengolahan lanjutan seperti segmentasi, pelacakan objek, atau klasifikasi berbasis warna. Praktikum ini membuktikan bahwa pendekatan berbasis warna dalam ruang HSV sangat efektif dan mudah diterapkan dalam sistem pengolahan citra digital.

DAFTAR PUSTAKA

1. Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4th ed.). Pearson.
2. Szeliski, R. (2022). *Computer Vision: Algorithms and Applications* (2nd ed.). Springer.
3. Shapiro, L. G., & Stockman, G. C. (2021). *Computer Vision*. Prentice Hall.
4. OpenCV Documentation. (2024).
5. Python Software Foundation. (2024). *Python 3.x Documentation*.