UTS

PENGOLAHAN CITRA

NAMA : Fakhri Faros

NIM : 202331200

KELAS : F

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

NO.PC :

ASISTEN : 1. Sasikirana Ramadhanty Setiawan Putri

2. Rizqy Amanda

3. Ridho Chaerullah

4. Sakura Amastasya Salsabila Setiyanto

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2024/2025

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

Di zaman sekarang, teknologi pengolahan citra makin banyak dipakai di berbagai bidang, mulai dari kamera HP sampai sistem keamanan. Salah satu hal penting dalam pengolahan citra adalah mengenali warna dalam gambar, terutama warna dasar kayak merah, hijau, dan biru. Tapi masalahnya, nggak semua warna bisa langsung dikenali cuma dari tampilan visual aja. Kadang pencahayaan, kualitas kamera, atau latar belakang bisa bikin warna-warna itu jadi susah dibedakan. Nah, di praktikum UTS ini, kita mencoba menyelesaikan masalah tersebut dengan cara membuat program yang bisa mendeteksi warna merah, hijau, dan biru dari gambar yang kita ambil sendiri. Program ini juga akan menampilkan hasil deteksi warnanya, serta histogram untuk menganalisis penyebaran warnanya.

1.2 Tujuan Masalah

Tujuan utama dari praktikum ini adalah untuk membuat sebuah program dengan bahasa Python yang bisa mendeteksi warna merah, hijau, dan biru pada citra (gambar) yang diambil dari kamera pribadi. Selain itu, program ini juga akan menampilkan hasil citra dari masing-masing warna yang berhasil dideteksi, lalu bikin histogram dari gambar tersebut supaya kita bisa lihat sebaran intensitas warna pada gambar. Jadi dari hasilnya, kita bisa tahu seberapa banyak masing-masing warna muncul dan seberapa baik program bisa memisahkannya.

1.3 Manfaat Masalah

Dari kegiatan ini, kita bisa belajar langsung gimana konsep deteksi warna bekerja di dunia nyata. Mahasiswa juga jadi lebih paham tentang gimana cara ngolah citra digital, mulai dari baca gambar, ubah warna ke format HSV, sampai pakai threshold buat ngeblok warna tertentu. Ini juga melatih logika pemrograman kita karena harus ngerancang script yang bisa jalan otomatis. Selain itu, dengan melihat histogram, kita jadi bisa menganalisis warna gambar secara kuantitatif, bukan cuma visual aja. Ilmu ini bisa banget dipakai buat tugas akhir, penelitian, atau bahkan kerjaan di bidang computer vision nantinya.

BAB II

LANDASAN TEORI

2.1 Warna dalam Citra Digital

Dalam dunia digital, gambar tersusun dari piksel-piksel kecil yang masing-masing punya warna. Warna itu biasanya ditulis dalam bentuk RGB, singkatan dari Red, Green, dan Blue. Tiga warna ini bisa dicampur buat bikin warna lain. Tapi, buat mendeteksi warna dengan komputer, format RGB ini kadang kurang cocok karena warnanya tergantung pencahayaan dan intensitas. Karena itu, kita pakai format HSV (Hue, Saturation, Value). Di HSV, komponen "Hue" nunjukin jenis warnanya, jadi lebih gampang buat bedain warna kayak merah, hijau, atau biru.

2.2 Deteksi Warna Menggunakan HSV dan Threshold

Supaya komputer bisa "lihat" warna merah, hijau, atau biru, kita harus ngasih tahu dia rentang angka berapa yang termasuk warna itu. Ini dilakukan pakai threshold. Jadi kita set angka minimal dan maksimal dari warna yang kita cari di format HSV, lalu kita bikin masking. Masking ini kayak nyaring gambar: bagian yang termasuk warna yang kita cari akan ditampilkan, yang lain disembunyiin. Misalnya buat warna merah, kita tentuin hue dari 0–10 dan 160–180 (karena merah ada di dua sisi spektrum). Lalu bagian gambar yang masuk rentang itu akan dipisahin jadi gambar sendiri.

2.3 Histogram Warna

Histogram warna adalah grafik yang nunjukin seberapa banyak piksel punya intensitas warna tertentu. Misalnya, kalau histogram merah tinggi di angka 200, itu artinya banyak piksel merah terang. Dengan histogram, kita bisa lihat warna apa yang paling dominan di gambar dan apakah penyebarannya rata atau nggak. Ini penting banget buat analisis visual, terutama kalau kita mau tahu seberapa "kuat" kehadiran warna tertentu dalam gambar. Histogram juga bisa nunjukin kalau gambar terlalu gelap atau terlalu terang.

2.4 Library Python: OpenCV dan Matplotlib

Dalam praktikum ini, kita pakai OpenCV buat ngelola gambar, mulai dari baca gambar, ubah ke HSV, sampai masking dan deteksi warna. Sedangkan Matplotlib dipakai buat nampilin gambar dan bikin grafik histogram. Dua library ini udah umum banget dipakai di dunia komputer visi dan cocok buat pemula maupun profesional.

BAB III

HASIL

3.1 Soal Nomor 1 :

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

Penjelasan Kodingan :

- cv2: OpenCV, library utama untuk pengolahan citra.
- numpy: Digunakan untuk membuat dan mengolah array (misalnya rentang warna).
- matplotlib.pyplot: Untuk menampilkan gambar dan histogram secara visual.

```
# --- Load Gambar --- 202331200_Fakhri Faros
img = cv2.imread('gambar tugas pcd.png')  # Ganti dengan nama file Anda
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

- cv2.imread(...): Membaca gambar dari file.
- cv2.cvtColor(..., cv2.COLOR_BGR2RGB): OpenCV secara default membaca dalam format BGR, dikonversi ke RGB untuk matplotlib.

```
# --- Konversi ke HSV --- 202331200_Fakhri Faros
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

- HSV (Hue, Saturation, Value) digunakan karena lebih efektif untuk segmentasi warna dibanding RGB.

```
# --- Deteksi Warna --- 202331200_Fakhri Faros
# Rentang warna merah (dua rentang karena melintasi batas 180 derajat di 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])

# Rentang warna hijau 202331200_Fakhri Faros
lower_green = np.array([40, 40, 40])
upper_green = np.array([80, 255, 255])

# Rentang warna biru 202331200_Fakhri Faros
lower_blue = np.array([100, 100, 100])
upper_blue = np.array([140, 255, 255])
```

- Warna merah memiliki nilai Hue di dua sisi ekstrem spektrum:
  - Sekitar 0–10 derajat (bagian awal spektrum)
  - Dan 160–180 derajat (bagian akhir spektrum)

- Karena HSV pada OpenCV melingkar (circular), maka kita perlu dua rentang:
  - lower_red1 dan upper_red1 untuk area awal (0–10)
  - lower_red2 dan upper_red2 untuk area akhir (160–180)

- Saturation dan Value diatur dari 100–255 agar tidak mendeteksi warna merah yang terlalu pucat atau terlalu gelap.
- Hue hijau berada di kisaran 40–80 derajat.
- Range ini bisa disesuaikan tergantung tingkat kepekatan hijau di gambar.
- Saturation dan Value dimulai dari 40 agar mendeteksi cukup banyak variasi hijau, termasuk hijau muda dan tua.
- Hue untuk biru berada di rentang 100–140 derajat.
- Saturation dan Value diatur dari 100 ke atas untuk menghindari warna abu-abu kebiruan atau biru sangat terang yang mirip putih.

```
# Masking 202331200_Fakhri Faros
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)
```

- cv2.inRange(...): Membuat mask (hitam putih) berdasarkan rentang HSV.
- bitwise_or: Menggabungkan dua mask merah menjadi satu.

```
# Terapkan Mask 202331200_Fakhri Faros
res_red = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_red)
res_green = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_green)
res_blue = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_blue)
```

- bitwise_and: Menampilkan hanya bagian gambar yang sesuai dengan mask (warna terdeteksi), bagian lain jadi hitam.

```
# --- Tampilkan Hasil Deteksi Warna --- 202331200_Fakhri Faros
plt.figure(figsize=(12, 8))
```

- Membuat kanvas baru dengan ukuran 12x8 inch.
- Berguna agar gambar-gambar tampil lebih besar dan jelas.

```
plt.subplot(2, 2, 1)
plt.imshow(img_rgb)
plt.title('Gambar Asli')
plt.axis('off')

plt.subplot(2, 2, 2)
plt.imshow(res_red)
plt.title('Deteksi Merah')
plt.axis('off')

plt.subplot(2, 2, 3)
plt.imshow(res_green)
plt.title('Deteksi Hijau')
plt.axis('off')

plt.subplot(2, 2, 4)
plt.imshow(res_blue)
plt.title('Deteksi Biru')
plt.axis('off')
```

- plt.subplot(2, 2, 1) artinya: Buat tampilan dengan 2 baris × 2 kolom, dan ini adalah posisi ke-1.
- plt.imshow(img_rgb): Tampilkan citra asli (format RGB).
- plt.title(...): Menuliskan judul di atas gambar.
- plt.axis('off'): Menyembunyikan sumbu koordinat (tidak perlu dalam tampilan visual).
- Baris ini diulang dengan indeks (2, 2, 2), (2, 2, 3), dan (2, 2, 4) untuk:
- res_red → Deteksi warna merah
- res_green → Deteksi warna hijau
- res_blue → Deteksi warna biru

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

- tight_layout(): Menata ruang antar subplot agar tidak saling tumpang tindih.
- plt.show(): Menampilkan semua subplot di layar.

```
# --- Histogram RGB --- 202331200_Fakhri Faros
colors = ('b', 'g', 'r')
```

- Tuple ini menentukan urutan warna: biru, hijau, merah.

```
plt.figure(figsize=(10, 5))
```

- Membuat kanvas histogram dengan ukuran 10×5 inci.

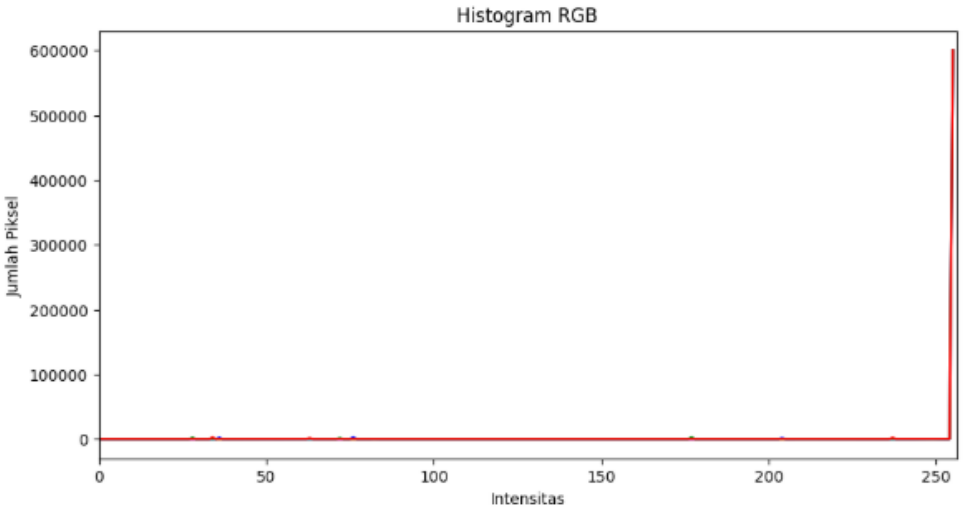
```
for i, col in enumerate(colors):
    hist = cv2.calcHist([img], [i], None, [256], [0, 256])
    plt.plot(hist, color=col)
    plt.xlim([0, 256])
```

- enumerate(colors): Digunakan untuk memproses channel warna biru (0), hijau (1), dan merah (2).
- cv2.calcHist(...):
- [img]: Gambar asli (dalam format BGR).
- [i]: Channel ke-i (0 = biru, 1 = hijau, 2 = merah).
- None: Tidak ada mask, berarti histogram untuk seluruh gambar.
- [256]: Histogram dengan 256 bin (0–255 intensitas).
- [0, 256]: Rentang intensitas piksel.
- plt.plot(hist, color=col): Gambar garis histogram warna sesuai channel.
- plt.xlim(...): Batasi sumbu X dari 0 sampai 256.

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

- plt.title(...): Judul histogram.
- xlabel('Intensitas'): Label sumbu X = nilai intensitas piksel (0–255).
- ylabel('Jumlah Piksel'): Label sumbu Y = jumlah piksel pada intensitas tertentu.
- plt.show(): Menampilkan histogram.

Hasil :



3.2 Soal Nomor 2 :

[4]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

- cv2: OpenCV, library utama untuk pengolahan citra.
- numpy: Digunakan untuk membuat dan mengolah array (misalnya rentang warna).
- matplotlib.pyplot: Untuk menampilkan gambar dan histogram secara visual.

```
# Load gambar
img = cv2.imread('citra_nama.jpg')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

- cv2.imread(...): Membaca gambar dari file.
- cv2.cvtColor(..., BGR2RGB): Konversi ke RGB agar cocok ditampilkan dengan matplotlib.
- cv2.cvtColor(..., BGR2HSV): Konversi ke HSV untuk segmentasi warna yang lebih akurat.

```
# --- Rentang Warna HSV ---
# 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])
```

- Warna merah terbagi dua rentang HUE karena melintasi batas spektrum HSV (0–10 dan 160–180).

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

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

- Hijau dan biru hanya butuh satu rentang karena posisinya tidak terpotong.

```
# --- 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)
```

- Membuat dua mask merah lalu digabung.
- cv2.inRange: Menghasilkan citra biner (putih = warna terdeteksi).

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

- Membuat mask hijau dan biru dari HSV.

```
# --- Gabungan Mask ---
mask_rb = cv2.bitwise_or(mask_red, mask_blue)
mask_rgb = cv2.bitwise_or(mask_rb, mask_green)
```

- Gabungkan merah + biru.
- Lalu gabungkan dengan hijau → merah + biru + hijau.

```
# --- Terapkan Mask ke Gambar ---
hasil_red = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_red)
hasil_rb = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_rb)
hasil_rgb = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_rgb)
```

- Menampilkan hasil deteksi dengan menyaring hanya area warna yang terdeteksi.
- Piksel di luar mask akan jadi hitam.

```python
# --- Terapkan Mask ke Gambar ---
hasil_red = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_red)
hasil_rb = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_rb)
hasil_rgb = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_rgb)

# --- Tampilkan Hasil Deteksi Warna ---
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
plt.imshow(hasil_red)
plt.title('Deteksi: Merah')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(hasil_rb)
plt.title('Deteksi: Merah + Biru')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(hasil_rgb)
plt.title('Deteksi: Merah + Biru + Hijau')
plt.axis('off')

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

- Menampilkan 3 gambar hasil deteksi secara berdampingan:
- Merah
- Merah + Biru
- Merah + Biru + Hijau
- axis('off'): Menyembunyikan sumbu koordinat.
- tight_layout(): Menghindari tampilan saling tumpang tindih.

```python
# --- Cetak Nilai Threshold ---
threshold_merah = [lower_red1[0], upper_red1[0], lower_red2[0], upper_red2[0]]
threshold_hijau = [lower_green[0], upper_green[0]]
threshold_biru = [lower_blue[0], upper_blue[0]]
```

- Kita hanya mengambil komponen Hue (H) dari rentang HSV karena Hue adalah dimensi yang menentukan jenis warna (misalnya merah ≈ 0, hijau ≈ 60, biru ≈ 120).
- [0] artinya kita ambil elemen pertama dari array HSV → komponen Hue.
- Masing-masing threshold disimpan dalam list Python agar bisa diproses atau digabungkan.

```python
# Gabungkan semua ambang batas dan urutkan
semua_threshold = sorted(set(threshold_merah + threshold_hijau + threshold_biru))

print("Threshold Merah :", threshold_merah)
print("Threshold Hijau :", threshold_hijau)
print("Threshold Biru  :", threshold_biru)
print("Threshold Terurut:", semua_threshold)
```

- threshold_merah + threshold_hijau + threshold_biru: Menggabungkan semua nilai ambang batas dari masing-masing warna.
- set(...): Menghapus nilai duplikat (jika ada), karena beberapa warna bisa punya nilai yang sama.
- sorted(...): Mengurutkan semua threshold dari nilai terkecil ke terbesar.
- Menampilkan masing-masing nilai ambang batas agar bisa dianalisis atau langsung dicatat ke dalam laporan.

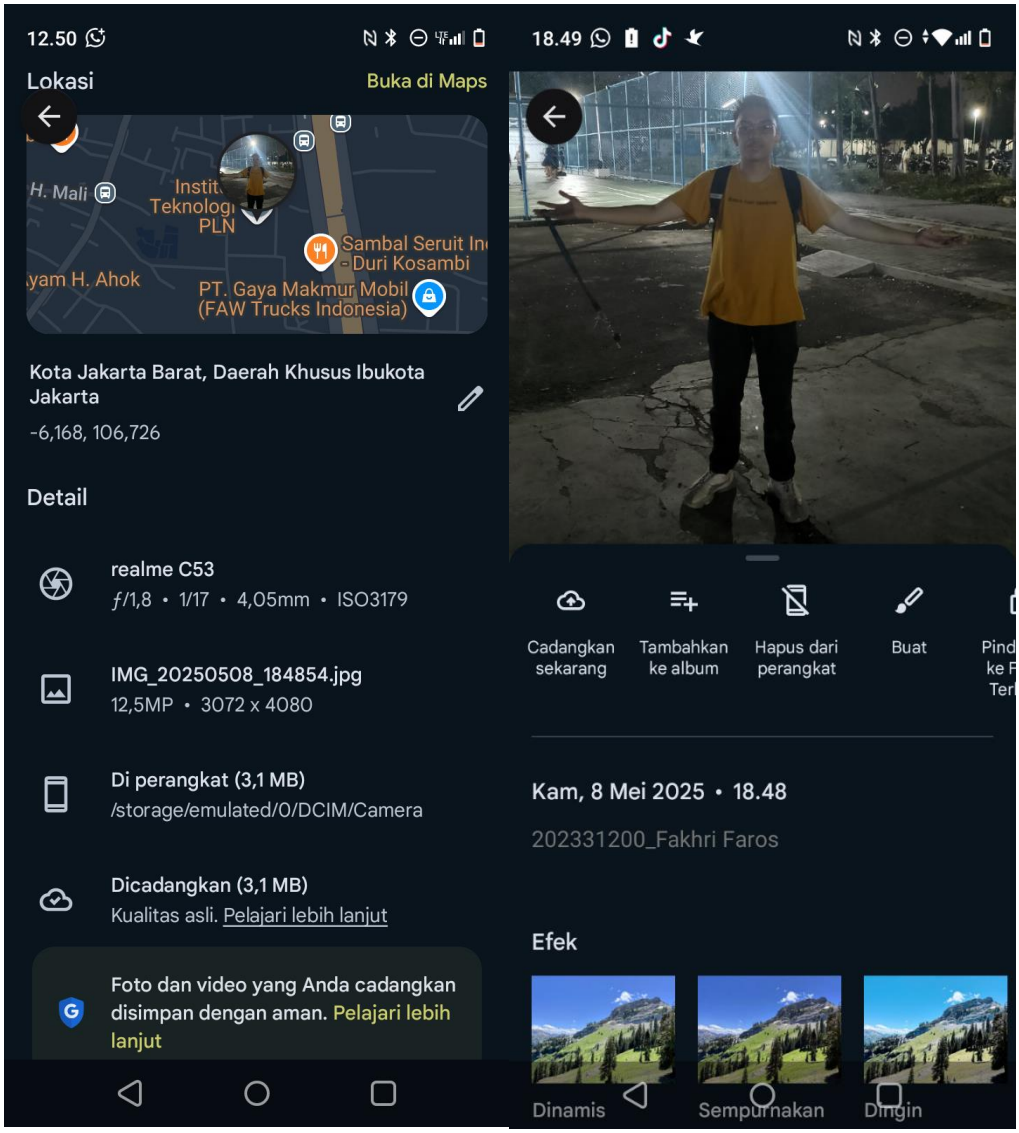
Hasil :


```
Threshold Merah : [np.int32(0), np.int32(10), np.int32(160), np.int32(180)]
Threshold Hijau : [np.int32(40), np.int32(80)]
Threshold Biru  : [np.int32(100), np.int32(140)]
Threshold Terurut: [np.int32(0), np.int32(10), np.int32(40), np.int32(80), np.int32(100), np.int32(140), np.int32(160), np.int32(180)]
```

Nilai tersebut diperoleh karena nilai ambang batas (threshold) HUE untuk setiap warna ditentukan berdasarkan karakteristik spektrum HSV. Warna merah memiliki dua rentang karena posisinya terletak di awal dan akhir spektrum (0–10 dan 160–180), sementara warna hijau (40–80) dan biru (100–140) cukup dengan satu rentang. Nilai-nilai ini dipilih agar segmentasi warna berjalan optimal terhadap warna tinta pena pada citra, tanpa menangkap area yang tidak relevan.

3.3 Soal Nomor 3 :

Citra :



Penjelasan Koding :

```
import cv2
import matplotlib.pyplot as plt
```

- cv2 (OpenCV): Untuk membaca gambar dan melakukan manipulasi.
- matplotlib.pyplot: Untuk menampilkan gambar hasil.

```
# Load gambar backlight
img = cv2.imread('foto_backlight.jpg')  #202331200_Fakhri Faros
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

- cv2.imread: Membaca gambar dari file (misalnya, foto diri Anda membelakangi cahaya).
- cv2.cvtColor(..., cv2.COLOR_BGR2GRAY): Mengubah gambar menjadi grayscale agar lebih mudah diproses (karena fokus pada intensitas, bukan warna).

```
# --- Perbaikan Kontras dan Kecerahan ---
# Tingkatkan kecerahan
bright = cv2.convertScaleAbs(gray, alpha=1, beta=50)  # beta = brightness
```

- convertScaleAbs: Fungsi OpenCV untuk mengubah intensitas piksel.
- alpha=1: Tidak mengubah kontras.
- beta=50: Menambah nilai 50 ke semua piksel → gambar menjadi lebih terang.
- Digunakan untuk mencerahkan area wajah/tubuh yang gelap akibat backlight.

```
# Tingkatkan kontras
contrast = cv2.convertScaleAbs(bright, alpha=2.0, beta=0)  # alpha = contrast
```

- alpha=2.0: Mengalikan semua nilai piksel → meningkatkan perbedaan terang-gelap → meningkatkan kontras.
- beta=0: Tidak menambah kecerahan.
- Digunakan untuk membuat area wajah/tubuh lebih tegas dan menonjol.

```
# Gabungkan: Kecerahan + Kontras
bright_contrast = cv2.convertScaleAbs(gray, alpha=2.0, beta=50)
```

- Dalam satu langkah:
  - alpha=2.0 → menaikkan kontras.
  - beta=50 → menaikkan kecerahan.

```
# Tampilkan hasil
plt.figure(figsize=(12, 6))
```

- Menentukan ukuran tampilan untuk keempat gambar (hasil proses citra).

```
plt.subplot(1, 4, 1)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Gambar Asli')
plt.axis('off')
```

- Gambar asli berwarna (RGB).
- Posisi 1 dari 4 kolom.

```
plt.subplot(1, 4, 2)
plt.imshow(gray, cmap='gray')
plt.title('Grayscale')
plt.axis('off')
```

- Gambar grayscale (abu-abu), hasil konversi awal.

```
plt.subplot(1, 4, 3)
plt.imshow(bright, cmap='gray')
plt.title('Dicerahkan')
plt.axis('off')
```

- Gambar grayscale yang sudah dicerahkan saja.

```
plt.subplot(1, 4, 4)
plt.imshow(bright_contrast, cmap='gray')
plt.title('Cerah + Kontras')
plt.axis('off')
```

- Gambar hasil pencerahan dan peningkatan kontras sekaligus → hasil terbaik.

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

- Menyusun tampilan agar tidak saling tumpang tindih.
- plt.show() untuk menampilkan semuanya.

Hasil :

BAB IV

PENUTUP

Dari hasil praktikum UTS Pengolahan Citra Digital ini, bisa disimpulkan bahwa teknik-teknik dasar dalam pengolahan citra seperti deteksi warna, analisis histogram, pencarian ambang batas, sampai perbaikan gambar backlight sangat berguna dalam memahami bagaimana komputer bisa "melihat" gambar.

Pada soal pertama, deteksi warna merah, hijau, dan biru berhasil dilakukan dengan cukup baik. Warna bisa dikenali sesuai area tulisan yang dibuat, dan histogram RGB membantu kita melihat distribusi warna dalam gambar. Soal kedua, kita menentukan ambang batas (threshold) warna di ruang HSV. Nilainya diambil sesuai dengan posisi warna dalam spektrum hue, dan hasil segmentasinya juga cukup akurat.

Lanjut ke soal ketiga, perbaikan gambar backlight dilakukan dengan cara meningkatkan kecerahan dan kontras gambar grayscale. Hasilnya, area wajah yang tadinya gelap jadi lebih kelihatan jelas dan fokus utama di gambar jadi nggak tenggelam sama cahaya di belakang.

Secara keseluruhan, praktikum ini menunjukkan kalau teknik-teknik sederhana bisa sangat berguna buat memproses dan memperbaiki gambar. Dan ternyata ngulik warna dan cahaya di gambar itu seru juga, apalagi pas lihat hasilnya berubah jadi lebih bagus.

.

DAFTAR PUSTAKA

1. Nisa Fachrunnisa, Ari Usman, Mufida Khairani. (2024). ” Implementasi Noise Removal Dan Image Enhancement Pada Citra Digital Menggunakan Metode Adaptive Median Filter”.
2. Fatwa, M., Rizki, R., Sriwinarty, P., & Supriyadi, E. (2022). Pengaplikasian Matlab pada Perhitungan Matriks. Papanda Journal of Mathematics and Science Research, 1(2), 81–93.
3. Jumadi, J., Yupianti, Y., & Sartika, D. (2021). Pengolahan Citra Digital Untuk Identifikasi Objek Menggunakan Metode Hierarchical Agglomerative Clustering. JST (Jurnal Sains Dan Teknologi), 10(2), 148–156.