UTS

PENGOLAHAN CITRA

NAMA : Nurul Dea Gamal

NIM : 202331154

KELAS : B

DOSEN : Ir. Darma Rusjdi, M.kom

NO.PC : 22

ASISTEN : 1. Davina Najwa Ermawan

2. Fakhrul Fauzi Nugraha Tarigan

3. Viana Salsabila Fairuz Syahla

4. Muhammad Hanief Febriansyah

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2024/2025

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

- Bagaimana cara mengenali warna-warna utama dalam sebuah citra digital?
- Apa saja teknik yang dapat digunakan untuk menentukan ambang batas warna dalam gambar?
- Bagaimana memperbaiki gambar yang memiliki pencahayaan latar belakang yang terlalu terang (backlight)?

1.2 Tujuan Masalah

- Mempelajari dan menerapkan metode deteksi warna menggunakan citra hasil tangkapan kamera sendiri.
- Mengetahui proses penentuan ambang batas dalam segmentasi warna.
- Melakukan perbaikan gambar backlight agar objek utama terlihat jelas menggunakan teknik pengolahan citra.

1.3 Manfaat Masalah

- Menambah pengalaman langsung dalam menerapkan teori pengolahan citra.
- Memberikan pemahaman dasar mengenai cara kerja deteksi warna dan pengolahan gambar.
- Membantu mengatasi permasalahan dalam foto seperti pencahayaan yang tidak merata.

BAB II
LANDASAN TEORI

Di era sekarang yang serba digital, gambar atau foto digital bukan cuma untuk sekadar dilihat atau disimpan sebagai kenangan. Citra digital ternyata juga punya banyak fungsi penting, mulai dari komunikasi visual, pengolahan data, sampai jadi bagian dari sistem kecerdasan buatan seperti pengenalan wajah atau kendaraan. Pengolahan citra digital sendiri bisa diartikan sebagai proses mengolah gambar menggunakan komputer agar hasilnya bisa lebih jelas, lebih informatif, atau lebih mudah dianalisis sesuai kebutuhan.

Kalau dibedah lebih dalam, gambar digital sebenarnya terdiri dari jutaan titik kecil yang disebut piksel. Setiap piksel menyimpan informasi tentang warna dan tingkat kecerahannya. Warna dalam gambar digital biasanya menggunakan sistem RGB, yaitu Red (merah), Green (hijau), dan Blue (biru). Ketiga warna dasar ini bisa digabung dengan berbagai kombinasi untuk menghasilkan warna-warna lainnya. Nah, dengan memahami dan memisahkan warna-warna ini, kita bisa mengenali bagian-bagian tertentu dalam gambar, misalnya tulisan warna merah, daun yang berwarna hijau, atau langit yang dominan biru.

Salah satu teknik dasar dalam pengolahan citra adalah thresholding, yaitu proses menentukan batas nilai tertentu untuk memisahkan bagian gambar. Contohnya, kalau kita ingin mengambil bagian gambar yang berwarna merah, maka kita bisa atur ambang batas nilai merah, dan hanya bagian piksel yang melebihi nilai itu yang akan dianggap sebagai bagian "merah". Teknik ini sangat berguna untuk mendeteksi objek atau mengelompokkan warna secara otomatis dalam suatu gambar.

Masalah umum dalam pengambilan gambar adalah pencahayaan yang tidak merata. Salah satunya adalah kondisi backlight, yaitu saat cahaya datang dari belakang objek utama. Akibatnya, objek malah jadi gelap dan kurang terlihat. Untuk memperbaiki gambar seperti ini, kita bisa menggunakan teknik penyesuaian kecerahan (brightness) dan kontras, supaya bagian yang penting (seperti wajah) bisa terlihat lebih jelas meskipun latar belakangnya terang.

Selain itu, untuk melihat seberapa baik pencahayaan atau sebaran warna dalam gambar, kita bisa menggunakan histogram. Histogram adalah grafik yang menunjukkan distribusi warna atau kecerahan piksel dalam gambar. Dari histogram, kita bisa menilai apakah gambar terlalu gelap, terlalu terang, atau punya kontras yang kurang. Ini sangat membantu saat kita ingin memperbaiki kualitas gambar secara keseluruhan.

Secara umum, dasar-dasar seperti pengenalan piksel, warna RGB, thresholding, perbaikan gambar backlight, dan histogram adalah pondasi yang penting dalam pengolahan citra. Walaupun terlihat teknis, semua ini bisa dipelajari dan diterapkan secara bertahap. Apalagi sekarang banyak tools atau library seperti OpenCV dan Matplotlib di Python yang mempermudah kita dalam mengolah gambar. Kalau kita sudah memahami dasar-dasar ini, akan sangat bermanfaat untuk mengembangkan teknologi yang melibatkan gambar di masa depan.

BAB III

HASIL

Soal 1: Mendeteksi warna pada CITRA

Library yang Digunakan

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
```

- ***cv2:*** library OpenCV, digunakan untuk membaca dan memproses gambar.
- ***numpy:*** untuk manipulasi array (struktur data gambar digital).
- ***matplotlib.pyplot:*** untuk menampilkan gambar dan grafik (histogram).
- ***matplotlib.patches:*** untuk menambahkan bentuk seperti **kotak (rectangle)** pada tampilan gambar.

Membaca dan Menyiapkan Gambar

```
# Baca gambar
image_path = "ndea rgb.jpg"
image = cv2.imread(image_path)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

- Gambar dibaca dari file ***"ndea rgb.jpg"***.
- Karena OpenCV menggunakan format warna BGR, gambar diubah ke RGB agar warna *tampil benar saat ditampilkan dengan matplotlib.*

Ekstrak Channel Warna (R, G, B)

```
# Ekstrak channel warna
r = image[:, :, 0]
g = image[:, :, 1]
b = image[:, :, 2]
```

- Gambar RGB terdiri dari 3 kanal: merah (R), hijau (G), dan biru (B).
- Bagian ini memisahkan masing-masing kanal untuk dianalisis atau ditampilkan secara terpisah.

Tampilan Gambar dan Histogram dalam Grid 4x2

```
# Tampilkan hasil dalam grid 4x2
fig, axes = plt.subplots(4, 2, figsize=(14, 16))
```

- Membuat ***grid 4 baris* × *2 kolom*** untuk menampilkan 4 gambar (RGB + 3 channel warna) dan 4 histogram-nya.
- ***figsize=(14, 16)*** memastikan tampilannya besar dan rapi.

Menampilkan citra

```
# CITRA KONTRAS
axes[0, 0].set_title('CITRA KONTRAS')
axes[0, 0].imshow(image)
axes[0, 0].axis('off')
```

- Menampilkan gambar dengan judul "CITRA KONTRAS".
- ***imshow(image):*** menampilkan citra berwarna (image) pada sumbu axes[0, 0].
- ***axis('off'):*** menghilangkan sumbu/tick dari tampilan gambar agar tampilan lebih bersih.

Menambahkan batas persegi

```
height, width, _ = image.shape
rect = patches.Rectangle((0, 0), width, height, linewidth=3, edgecolor='black', facecolor='none')
axes[0, 0].add_patch(rect)
```

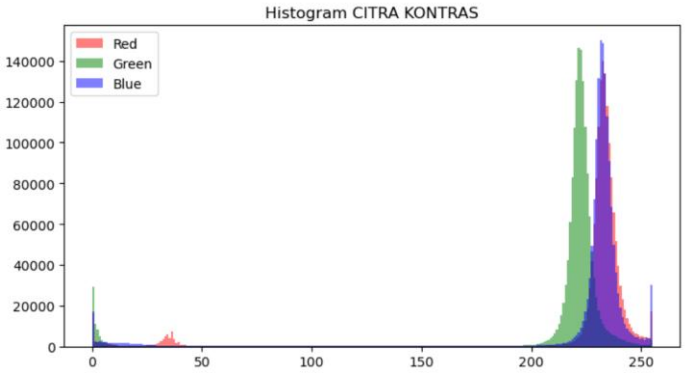
- Mengambil ukuran tinggi dan lebar gambar.
- Membuat persegi panjang (Rectangle) sebagai batas tepi gambar.
- Diberi garis tepi hitam (***edgecolor='black'***) dan tanpa warna isi (***facecolor='none'***).
- Patch ini ditambahkan ke tampilan gambar di axes[0, 0].

Menampilkan histogram RGB

```
axes[0, 1].hist(r.ravel(), bins=256, color='red', alpha=0.5, label='Red')
axes[0, 1].hist(g.ravel(), bins=256, color='green', alpha=0.5, label='Green')
axes[0, 1].hist(b.ravel(), bins=256, color='blue', alpha=0.5, label='Blue')
axes[0, 1].set_title("Histogram CITRA KONTRAS")
axes[0, 1].legend()
```

- r, g, dan b adalah channel warna merah, hijau, dan biru dari citra.
- ravel() mengubah array 2D menjadi 1D untuk perhitungan histogram.
- Histogram menunjukkan distribusi intensitas warna dari 0 hingga 255 untuk masing-masing channel warna.
- Histogram ditampilkan di axes[0, 1].
- ***alpha=0.5:*** transparansi agar warna histogram bisa saling terlihat.
- ***legend():*** menampilkan keterangan warna.

GAMBAR ASLI

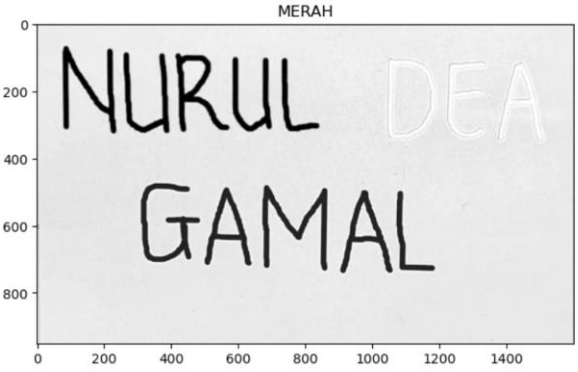


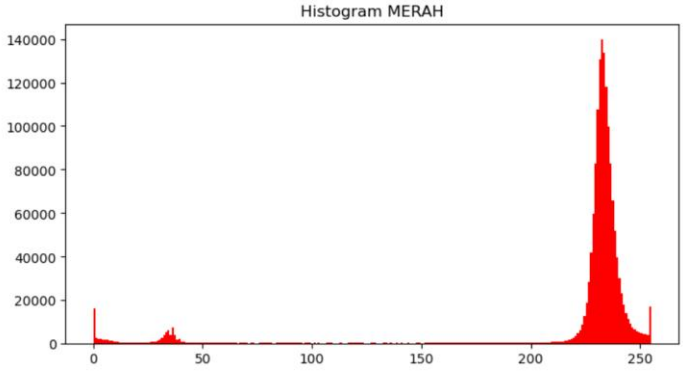
WARNA MERAH

```
# MERAH
axes[1, 0].set_title('MERAH')
axes[1, 0].imshow(r, cmap='gray')
axes[1, 0].axis('on')

axes[1, 1].hist(r.ravel(), bins=256, color='red')
axes[1, 1].set_title("Histogram MERAH")
```

Kode tersebut menampilkan komponen merah dari suatu gambar dalam bentuk gambar grayscale, dan histogram distribusi intensitas warna merahnya untuk analisis visual.



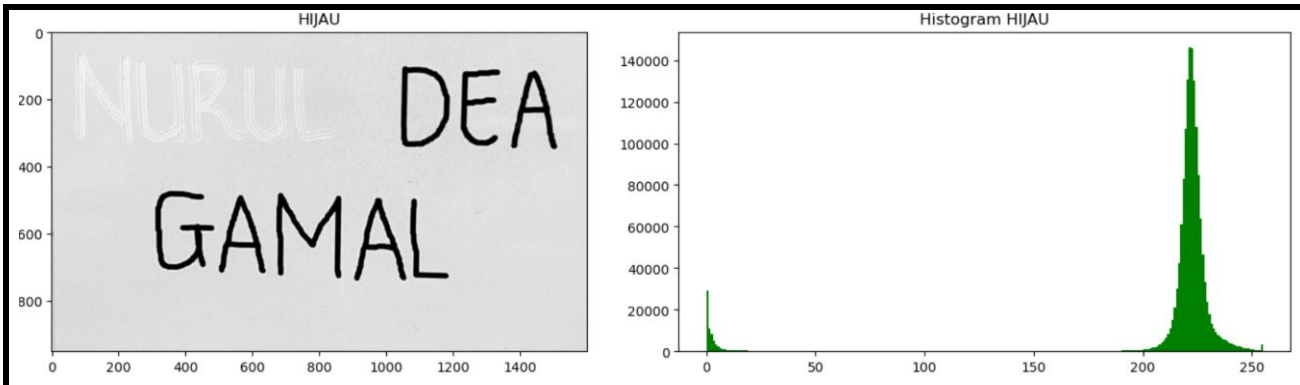


WARNA HIJAU

```
# HIJAU
axes[2, 0].set_title('HIJAU')
axes[2, 0].imshow(g, cmap='gray')
axes[2, 0].axis('on')

axes[2, 1].hist(g.ravel(), bins=256, color='green')
axes[2, 1].set_title("Histogram HIJAU")
```

Kode tersebut menampilkan komponen hijau dari gambar dalam bentuk grayscale, serta histogram intensitas warna hijau untuk analisis distribusi pikselnya.

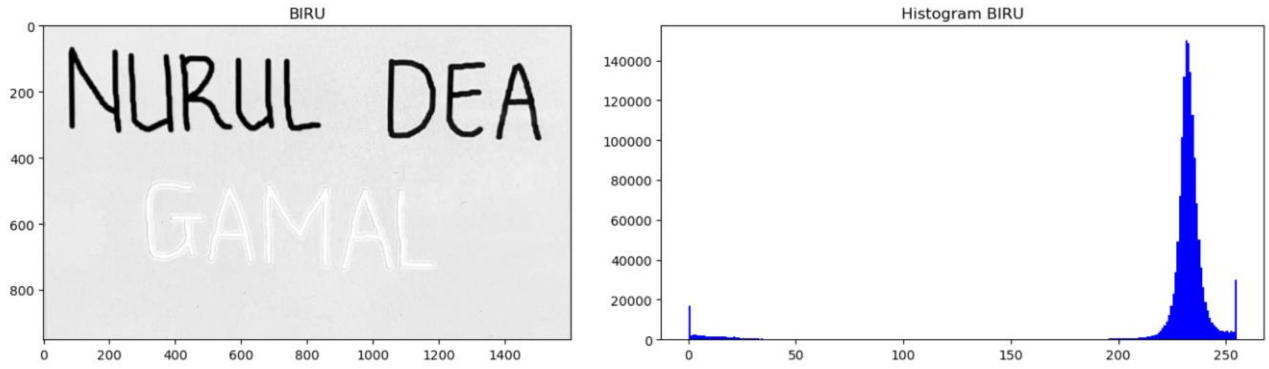


WARNA BIRU

```
# BIRU
axes[3, 0].set_title('BIRU')
axes[3, 0].imshow(b, cmap='gray')
axes[3, 0].axis('on')

axes[3, 1].hist(b.ravel(), bins=256, color='blue')
axes[3, 1].set_title("Histogram BIRU")
```

Kode tersebut menampilkan komponen biru dari gambar dalam bentuk grayscale, serta histogram intensitas warna biru untuk analisis distribusi pikselnya.



mengatur dan menampilkan plot

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

kedua perintah ini memastikan hasil visualisasi tampil dengan baik dan jelas.

Soal 2: Urutkan ambang batas terkecil sampai terbesar

Library yang di gunakan

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

- ***cv2:*** untuk pemrosesan citra (OpenCV).
- ***numpy:*** untuk operasi array.
- ***matplotlib.pyplot:*** untuk menampilkan gambar dan grafik.

Membaca dan Mengonversi Gambar

```
# Baca gambar dan konversi ke RGB
image_path = "ndea rgb.jpg"  # Ganti jika perlu
image = cv2.imread(image_path)
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
```

- Membaca gambar dengan OpenCV (***format default: BGR***).
- Mengonversi ke RGB untuk visualisasi yang benar di Matplotlib.
- Juga mengonversi ke HSV, yang lebih efektif untuk deteksi warna.

Menentukan Rentang Warna (HSV)

```
# Rentang deteksi warna
lower_blue = np.array([100, 100, 50])
upper_blue = np.array([140, 255, 255])
lower_red1 = np.array([0, 100, 50])
upper_red1 = np.array([10, 255, 255])
lower_red2 = np.array([160, 100, 50])
upper_red2 = np.array([180, 255, 255])
lower_green = np.array([40, 100, 50])
upper_green = np.array([80, 255, 255])
```

- Menentukan batas bawah dan atas warna biru, merah, dan hijau dalam ruang warna HSV. Warna merah butuh dua rentang karena berada di awal dan akhir spektrum HSV.

Membuat Masker Warna

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

- Mendeteksi piksel yang berada dalam rentang warna tertentu dan menghasilkan ***masker biner*** (***putih** = **terdeteksi, hitam** = **tidak***).

Gabungkan Masker

```
# Gabungan mask
none_mask = np.zeros_like(mask_blue)
red_blue_mask = cv2.bitwise_or(mask_red, mask_blue)
rgb_mask = cv2.bitwise_or(cv2.bitwise_or(mask_red, mask_blue), mask_green)
```

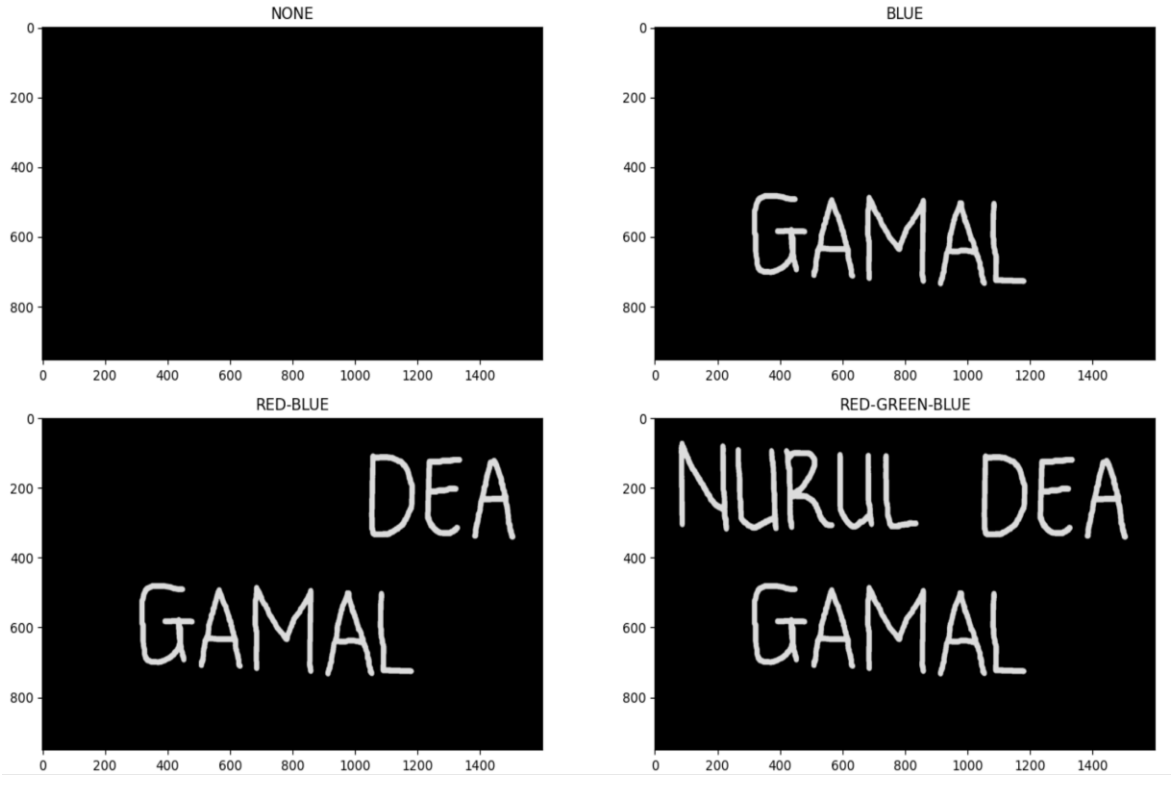
- ***none_mask***: masker kosong.
- ***red_blue_mask:*** gabungan warna merah dan biru.
- ***rgb_mask***: gabungan semua (merah, hijau, biru).

Fungsi Pewarnaan Masker

```
# Fungsi untuk menampilkan hasil dengan warna putih silver
def apply_mask_silver(mask):
    output = np.zeros_like(image_rgb)
    output[mask > 0] = [220, 220, 220]
    return output
```

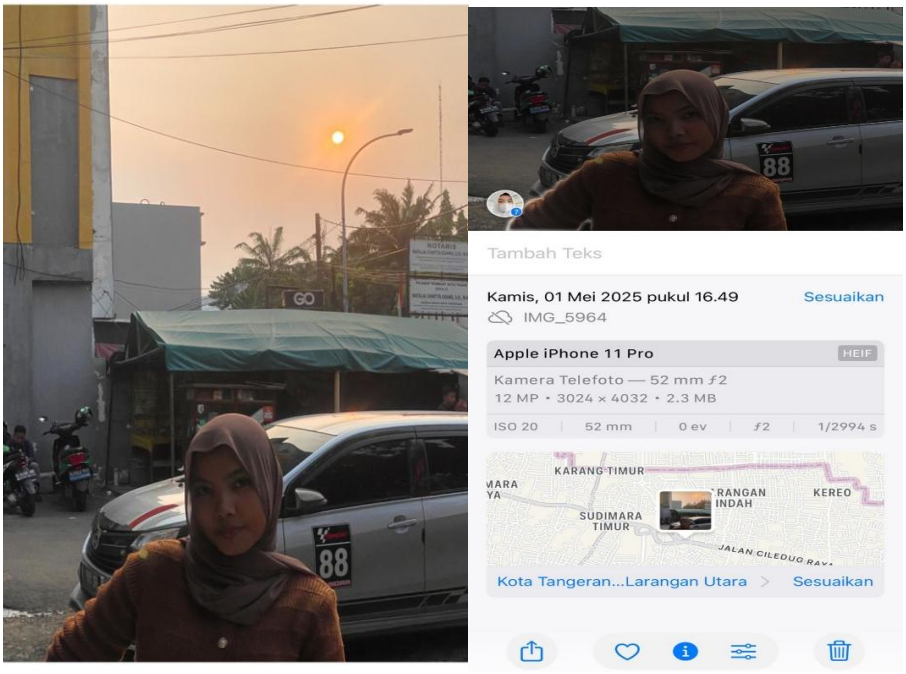
Visualisasi Hasil

Tulisan terdeteksi jika nilai ambang batas diterapkan pada saluran warna yang mengandung informasi cukup. Semakin banyak saluran (RGB) yang digunakan, semakin lengkap informasi yang diperoleh, sehingga lebih banyak tulisan yang bisa dikenali. Sebaliknya, jika tidak ada saluran atau informasinya minim, tulisan tidak terdeteksi.



Soal 3: perbaiki gambar backlight

Gambar yang di gunakan :



Library yang di gunakan

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

- ***cv2:*** Library OpenCV untuk pemrosesan citra.
- ***numpy:*** Library numerik (digunakan dalam proses internal OpenCV).
- ***matplotlib.pyplot:*** Digunakan untuk menampilkan gambar secara visual dalam bentuk grafik.

Membaca dan Mengonversi Gambar

```
# Baca gambar
img = cv2.imread('nuruldeag rgb.jpg')
```

- Membaca gambar berwarna dari file bernama ***'nuruldeag rgb.jpg'***.

```
# Ubah ke grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

- Mengubah gambar berwarna menjadi ***grayscale*** agar lebih fokus pada intensitas cahaya, bukan warna.
- Penting saat mengolah kontras dan pencahayaan, terutama untuk wajah.

Peningkatan Kecerahan (Brightness)

```
# Tingkatkan kecerahan
bright = cv2.convertScaleAbs(gray, alpha=1, beta=50)
```

- ***alpha = 1:*** Tidak mengubah kontras.
- ***beta = 50:*** Menambahkan 50 ke setiap nilai piksel (*meningkatkan* ***kecerahan***).
- Berguna untuk membuat wajah lebih terang jika terkena bayangan backlight.

Peningkatan Kontras

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

- ***alpha = 2.0:*** Menggandakan kontras.
- ***beta = 0:*** Tidak mengubah kecerahan.
- Kontras meningkatkan perbedaan antara area terang dan gelap — membantu wajah terlihat lebih jelas.

Gabungan Brightness + Kontras

```
# Gabungan peningkatan brightness dan kontras
bright_contrast = cv2.convertScaleAbs(gray, alpha=2.0, beta=50)
```

- Gabungan dari keduanya:
  - ✓ ***alpha = 2.0*** → kontras meningkat.
  - ✓ ***beta = 50*** → brightness meningkat.
- Biasanya hasil terbaik didapat di sini untuk gambar backlight.

Menampilkan Hasil Gambar

```
# Tampilkan semua hasil
plt.figure(figsize=(12, 8))
```

- Membuat kanvas gambar ukuran besar agar tampilan visual lebih jelas.

```
plt.subplot(2, 3, 1)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title("Gambar Asli")
plt.axis("off")
```

- Menampilkan gambar asli (diubah dari BGR ke RGB agar warna terlihat normal).


```
plt.subplot(2, 3, 2)
plt.imshow(gray, cmap='gray')
plt.title("Grayscale")
plt.axis("off")
```

- Menampilkan gambar dalam format grayscale.

```
plt.subplot(2, 3, 3)
plt.imshow(bright, cmap='gray')
plt.title("Dipercerah")
plt.axis("off")
```


```
plt.subplot(2, 3, 4)
plt.imshow(contrast, cmap='gray')
plt.title("Diperkontras")
plt.axis("off")
```

```
plt.subplot(2, 3, 5)
plt.imshow(bright_contrast, cmap='gray')
plt.title("Cerah + Kontras")
plt.axis("off")
```

- Menampilkan hasil peningkatan brightness, kontras, dan kombinasi keduanya.

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

- Menata semua subplot agar rapi dan menampilkannya ke layar.

BAB IV

PENUTUP

Dari hasil pengerjaan tugas ini, saya bisa menyimpulkan bahwa pengolahan citra digital bukan hanya teori semata, tapi juga sangat bisa diterapkan langsung ke hal-hal praktis. Dengan memanfaatkan bahasa pemrograman Python dan beberapa pustaka seperti OpenCV, saya jadi bisa memproses gambar sendiri untuk mengenali warna, memperbaiki kualitas gambar, dan menganalisis tampilan visualnya. Walaupun awalnya terlihat rumit, ternyata setelah dijalani, proses pengolahan gambar seperti mendeteksi warna merah, hijau, dan biru bisa dilakukan secara bertahap dan logis.

Proses pencarian nilai ambang batas juga cukup menarik, karena saya jadi tahu bagaimana sebuah gambar bisa dipisahkan berdasarkan nilai-nilai warnanya. Saya belajar bahwa thresholding itu bukan sekadar angka, tapi sangat berpengaruh terhadap hasil akhir dari pengolahan citra. Kalau nilai ambangnya tidak tepat, maka hasilnya juga tidak akan sesuai harapan.

Hal lain yang cukup menantang adalah memperbaiki gambar backlight. Biasanya saat memotret di luar ruangan dan membelakangi matahari, wajah kita jadi gelap. Lewat tugas ini saya mencoba mengolah gambar seperti itu, mengubahnya ke grayscale lalu mempercerah dan meningkatkan kontras supaya wajah atau tubuh bisa lebih jelas kelihatan. Dari sini saya sadar, ternyata dengan teknik yang tepat, gambar yang awalnya terlihat buruk bisa jadi jauh lebih baik.

Saya juga jadi paham pentingnya histogram. Grafik ini membantu saya melihat sebaran warna dalam gambar. Dengan histogram, saya bisa tahu apakah gambar saya terlalu gelap, terlalu terang, atau sudah pas pencahayaannya. Ini sangat berguna sebagai bahan evaluasi sebelum melakukan pengolahan lebih lanjut.

Secara keseluruhan, tugas ini membuat saya lebih memahami dasar-dasar pengolahan citra. Bukan cuma soal koding atau teori, tapi juga tentang cara melihat dan menganalisis gambar dengan lebih detail. Pengetahuan ini menurut saya akan sangat bermanfaat ke depannya, apalagi jika ingin mendalami bidang teknologi visual atau kecerdasan buatan. Saya juga jadi lebih terbiasa untuk menganalisis masalah visual dan mencari solusi secara logis dan terstruktur.

DAFTAR PUSTAKA

Jain, A. K. (2021). Fundamentals of Digital Image Processing. Pearson.

Zhang, Y., & Wang, S. (2020). "Color-Based Image Segmentation Techniques," Journal of Visual Communication and Image Representation.

Liu, X., & Yang, Q. (2023). "Modern Applications of Image Processing in Computer Vision," Image and Vision Computing Journal.