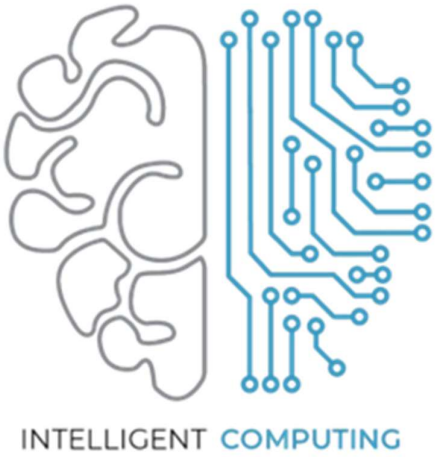
UTS

PENGOLAHAN CITRA



NAMA : Dimas Dwi Al Bukhori

NIM : 202331307

KELAS : B

DOSEN : Ir. Darma Rusjdi, M.Kom

NO.PC : 09

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

DAFTAR ISI 2
BAB I 3
PENDAHULUAN 3
1.1 Rumusan Masalah 3
1.2 Tujuan Masalah 3
1.3 Manfaat Masalah 3
BAB II 4
LANDASAN TEORI 4
BAB III 6
HASIL 6
3.1 Soal 1 Deteksi Warna Pada Citra 6
3.2 Soal 2 Memperbaiki gambar Backlight 9
BAB IV 12
PENUTUP 12
4.1 Kesimpulan 12
DAFTAR PUSTAKA 13

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

- Bagaimana mendeteksi dan menampilkan warna merah, hijau, dan biru secara terpisah pada citra digital hasil potret pribadi?
- Bagaimana cara menentukan nilai ambang batas terkecil hingga terbesar untuk memisahkan warna dalam citra?
- Bagaimana proses pembuatan histogram dari masing-masing warna dan bagaimana interpretasi hasilnya?
- Bagaimana cara mengatasi masalah pencahayaan pada citra backlight agar objek utama tetap terlihat jelas?\

1.2 Tujuan Masalah

- Mengembangkan program deteksi warna RGB pada citra menggunakan bahasa pemrograman Python.
- Menemukan nilai ambang batas optimal untuk klasifikasi warna dalam gambar.
- Menganalisis citra berdasarkan histogram intensitas warna untuk memahami distribusi warnanya.
- Meningkatkan kualitas citra backlight agar bagian wajah atau profil lebih terang dan fokus.

1.3 Manfaat Masalah

- Mahasiswa memahami penerapan teori deteksi warna dalam pengolahan citra secara langsung.
- Meningkatkan kemampuan teknis dalam pengolahan dan analisis citra digital menggunakan Python.
- Menumbuhkan keterampilan problem solving dalam menangani gambar dengan kondisi pencahayaan buruk.
- Meningkatkan kesiapan mahasiswa untuk menghadapi proyek pengolahan citra nyata di dunia kerja atau riset.

BAB II

LANDASAN TEORI

Arti histogram pada pengolahan citra digital. Histogram dalam pengolahan citra digital adalah grafik yang menunjukkan sebaran intensitas piksel dalam sebuah gambar. Sumbu horizontal pada histogram (x) mewakili nilai intensitas piksel, biasanya dari 0 (hitam) hingga 255 (putih) untuk gambar 8-bit, sedangkan sumbu vertikal (y) menunjukkan jumlah piksel yang memiliki intensitas tertentu tersebut. Dengan melihat bentuk histogram, kita bisa mengetahui apakah gambar cenderung gelap, terang, atau memiliki kontras yang rendah. Histogram sangat berguna untuk menganalisis kecerahan dan kontras gambar. Jika sebagian besar nilai piksel berada di sebelah kiri (nilai rendah), berarti gambar cenderung gelap. Sebaliknya, jika histogram lebih banyak berada di sebelah kanan (nilai tinggi), Selain itu, histogram juga digunakan untuk meningkatkan kualitas gambar histogram equalization, yang berfungsi menyebarkan intensitas piksel agar kontras gambar lebih seimbang.

Histogram citra adalah representasi grafis dari distribusi intensitas piksel dalam sebuah citra. Histogram menunjukkan jumlah piksel untuk setiap tingkat keabuan (dalam citra grayscale) atau intensitas warna (dalam citra berwarna). Dalam pengolahan citra digital, histogram berguna untuk analisis kontras, kecerahan, dan distribusi warna.dan Fungsi cv2.calcHist() dari OpenCV digunakan untuk menghitung histogram sebuah citra.

Fungsi cv2.equalizeHist() melakukan histogram equalization, yaitu menyamakan distribusi intensitas piksel agar kontras citra meningkat. Fungsi ini bekerja hanya pada citra grayscale. Dan Dampaknya, citra menjadi lebih terang atau lebih kontras, terutama pada citra yang tampak kusam atau gelap. Piksel yang semula terkonsentrasi di intensitas tertentu akan tersebar lebih merata.

Deteksi Warna RGB pada Citra Digital
Deteksi warna dalam citra digital melibatkan pemisahan komponen warna utama (merah, hijau, biru) dari gambar berwarna. Proses ini biasanya dilakukan dengan mengubah ruang warna dari RGB ke HSV atau HSL untuk memudahkan segmentasi warna berdasarkan hue (warna), saturation (kejenuhan), dan value (kecerahan). Metode ini memungkinkan identifikasi warna secara lebih akurat karena HSV lebih mendekati persepsi manusia terhadap warna.

Segmentasi Warna dengan Thresholding
Segmentasi citra menggunakan metode thresholding bertujuan untuk memisahkan objek berdasarkan intensitas piksel. Metode Otsu adalah salah satu teknik thresholding global yang menentukan nilai ambang optimal dengan meminimalkan varians intra-kelas dan memaksimalkan varians antar-kelas. Teknik ini efektif untuk citra dengan histogram bimodal, di mana dua puncak mewakili objek dan latar belakang.

Histogram Citra dan Analisis Intensitas
Histogram citra merepresentasikan distribusi intensitas piksel dalam gambar. Analisis histogram digunakan untuk memahami kontras dan kecerahan citra. Teknik seperti histogram equalization dapat meningkatkan kontras dengan meratakan distribusi intensitas, sementara adaptive histogram

equalization (AHE) dan contrast limited adaptive histogram equalization (CLAHE) digunakan untuk meningkatkan kontras lokal dan menghindari amplifikasi noise berlebih.

Perbaikan Citra Backlight

Citra dengan kondisi backlight sering kali menghasilkan objek utama yang gelap karena sumber cahaya berada di belakang objek. Untuk mengatasi hal ini, teknik peningkatan kontras dan kecerahan diterapkan, seperti histogram equalization dan CLAHE, yang dapat menyesuaikan intensitas piksel secara lokal untuk menonjolkan detail pada area gelap tanpa mengorbankan kualitas keseluruhan gambar.

Implementasi dalam Bahasa Pemrograman Python

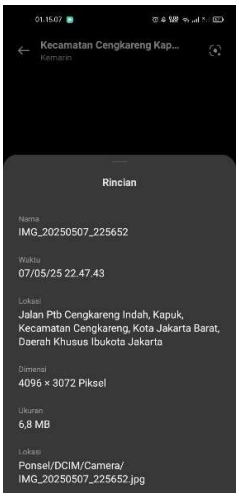
Python merupakan bahasa pemrograman yang umum digunakan dalam pengolahan citra digital karena ketersediaan pustaka seperti OpenCV dan scikit-image. Pustaka ini menyediakan fungsi-fungsi untuk konversi ruang warna, segmentasi, analisis histogram, dan peningkatan citra, memungkinkan implementasi metode-metode di atas secara efisien dan efektif.

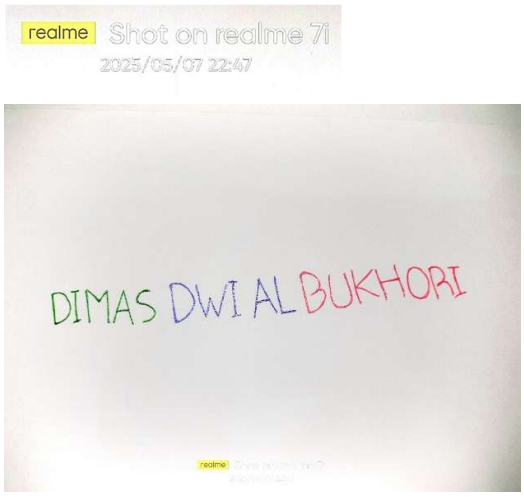
BAB III

HASIL

3.1 Soal 1 Deteksi Warna Pada Citra

- Bukti Gambar Terlampir:





1. Pertama-tama Mengimport Library berikut ini

Import Library

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

2. Fingsi ini untuk Membaca citra dari file bernama nama.jpg. lalu Mengubah format warna dari BGR ke RGB.

Membaca Gambar

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

3. Selanjutnya fungsi ini untuk Mengubah citra dari format BGR ke HSV, Karena Format HSV (Hue, Saturation, Value) lebih mudah untuk mendeteksi warna berdasarkan spektrum warna.

Konversi ke HSV

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

4. Printah ini digunakan untuk deteksi warna Hijau, Birru, Maupun Merah.

- Fungsi lower dan upper ini digunakan untuk Menentukan rentang HSV untuk warna Hijau, Birru, Maupun Merah.
- Kalau mask dan res digunakan untuk Membuat masker biner untuk deteksi warna Hijau, Birru, Maupun Merah. dan menerapkannya ke gambar asli.
- Kenapa merah terlihat banyak, Karena warna merah berada di dua sisi spektrum hue HSV (0–10 dan 160–180), perlu dua rentang. Fungsinya untuk Menggabungkan dua masker merah untuk mendeteksi seluruh range merah.

Deteksi warna ¶

Hijau

[7]:
```
lower_green = np.array([40, 40, 40])
upper_green = np.array([80, 255, 255])
mask_green = cv2.inRange(hsv, lower_green, upper_green)
res_green = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_green
```

Biru

[9]:
```
lower_blue = np.array([100, 100, 50])
upper_blue = np.array([140, 255, 255])
mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)
res_blue = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_blue)
```

Merah

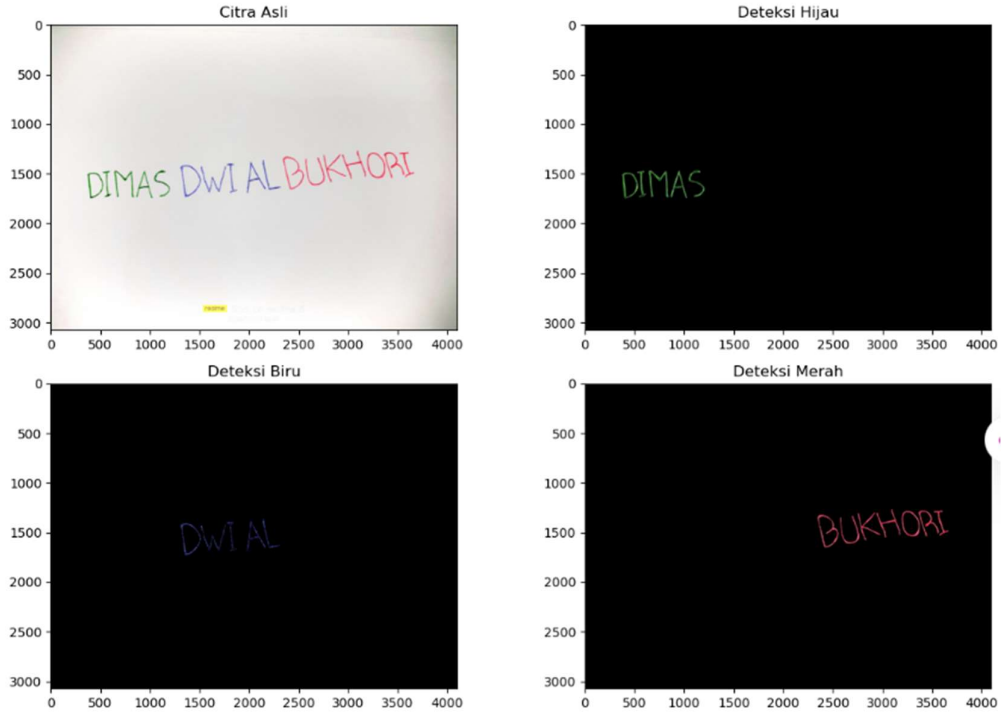
[11]:
```
lower_red1 = np.array([0, 100, 100])
upper_red1 = np.array([10, 255, 255])
lower_red2 = np.array([160, 100, 100])
upper_red2 = np.array([180, 255, 255])
mask_red1 = cv2.inRange(hsv, lower_red1, upper_red1)
mask_red2 = cv2.inRange(hsv, lower_red2, upper_red2)
mask_red = cv2.bitwise_or(mask_red1, mask_red2)
res_red = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_red)
```

5. Selanjutnya ini untuk Menampilkan semua hasil (citra asli dan hasil deteksi warna) dalam satu grid (2x2). plt.imshow() digunakan untuk menampilkan gambar RGB.

Tampilkan hasil deteksi warna

[46]:
```
titles = ['Citra Asli', 'Deteksi Hijau', 'Deteksi Biru', 'Deteksi Merah']
images = [img_rgb, res_green, res_blue, res_red]

plt.figure(figsize=(12, 8))
for i in range(4):
    plt.subplot(2, 2, i+1)
    plt.imshow(images[i])
    plt.title(titles[i])
plt.tight_layout()
plt.show()
```

Hasil Deteksi warna



6. Fungsi ini untuk Menyimpan data ambang batas hue dari setiap warna sebagai tuple (nama, ambang_bawah, ambang_atas).

Nilai Ambang Batas Hijau, Biru, dan Merah ¶

[21]:
```
thresholds = [
    ('Merah 1', lower_red1[0], upper_red1[0]),
    ('Hijau', lower_green[0], upper_green[0]),
    ('Biru', lower_blue[0], upper_blue[0]),
    ('Merah 2', lower_red2[0], upper_red2[0])
]
```

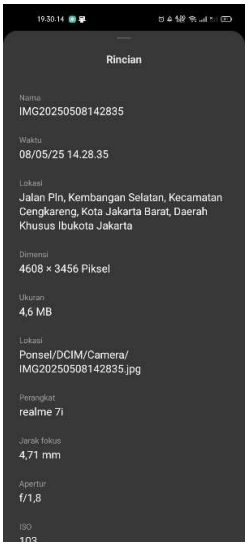
7. Fungsi ini digunakan untuk Mengurutkan warna berdasarkan ambang batas bawah (Hue) dari kecil ke besar. Lalu akan dicetak hasil urutan ambang batas ke terminal.

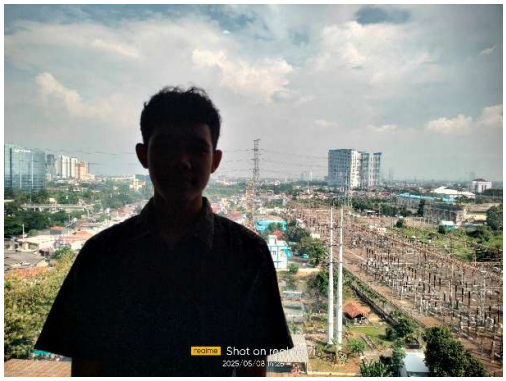
```
sorted_thresholds = sorted(thresholds, key=lambda x: x[1])

print("Urutan Ambang Batas Berdasarkan Hue:")
for name, low, high in sorted_thresholds:
    print(f"{name}: Hue {low} - {high}")
```

8. Hasil Ambang Batas
   Urutan Ambang Batas Berdasarkan Hue:
   Merah 1: Hue 0 - 10
   Hijau: Hue 40 - 80
   Biru: Hue 100 - 140
   Merah 2: Hue 160 – 180

3.2 Soal 2 Memperbaiki gambar Backlight

- Bukti Gambar Terlampir:





1. Pertama-tama Mengimport Library berikut ini

Import Library

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

2. Ini fungsinya untuk Membaca gambar dari file. backlight.jpg harus berada di direktori yang sama atau path-nya harus disesuaikan. Lalu cv2.resize() ini Mengubah ukuran gambar menjadi 600x400 piksel.

Membaca gambar

```
4]: img = cv2.imread('backlight.jpg')
    img = cv2.resize(img, (600, 400))
```

3. Selanjutnya fungsi ini digunakan untuk Mengubah urutan warna dari BGR (standar OpenCV) ke RGB (standar matplotlib). Lalu gambar akan di tampilkan dan di tambahkan judul lalu di tampilkan gambar aslinya.

Menampilkan Gambar Asli

```
[122]: img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
       plt.imshow(img_rgb)
       plt.title("Gambar Asli")
       plt.show()
```

4. Fungsi ini mengkonversi atau Mengubah gambar berwarna menjadi grayscale (hitam-putih).

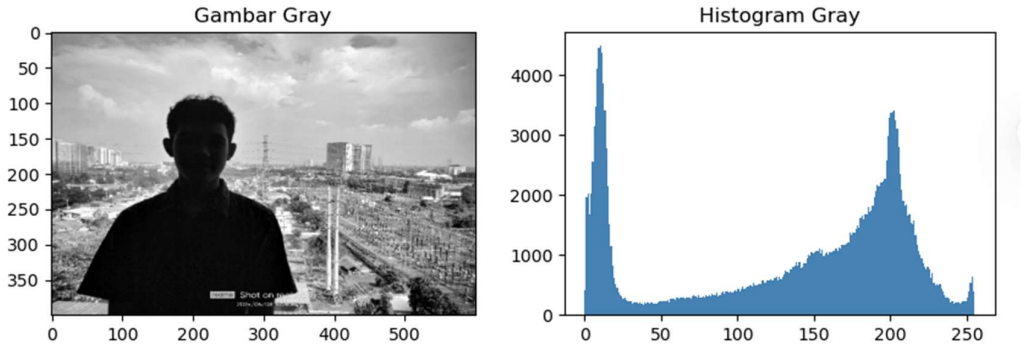
BGR to Grayscale ¶

```
[36]: gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

5. Selanjutnya Fungsi ini Menampilkan gambar grayscale dan Menampilkan histogram tingkat keabuan dari gambar grayscale. Histogram menunjukkan jumlah piksel untuk tiap intensitas (0–255).

Menampilkan gambar Grayscale dan Histogram

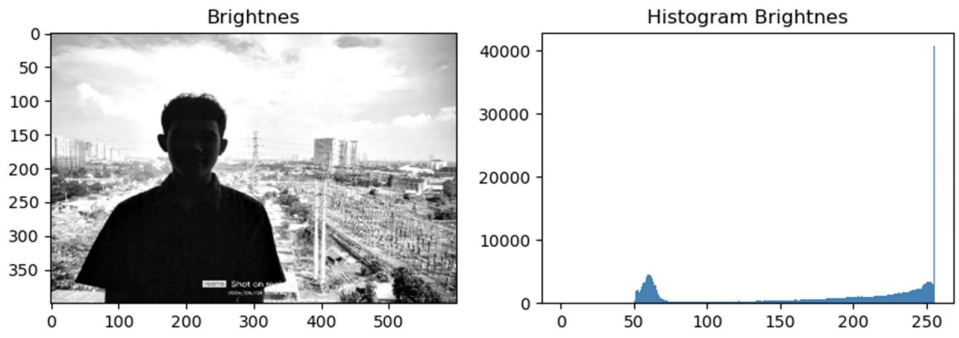
```
92]: fig, axs = plt.subplots(1, 2, figsize=(10, 3))

     # Tampilkan gambar
     axs[0].imshow(gray, cmap='gray')
     axs[0].set_title("Gambar Gray")

     # Tampilkan histogram
     axs[1].hist(gray.ravel(), bins=256, range=[0, 256], color='steelblue')
     axs[1].set_title("Histogram Gray")

92]: Text(0.5, 1.0, 'Histogram Gray')
```



6. Fungsi cv2.convertScaleAbs() Melakukan transformasi linier pada piksel. alpha=1 mempertahankan kontras, dan beta=50 menambah kecerahan sebesar 50 unit. Lalu Menampilkan gambar yang telah ditingkatkan kecerahannya dan histogram barunya.

Menampilkan gambar Brightness dan Histogram

```
[112]: bright = cv2.convertScaleAbs(gray, alpha=1, beta=50)
       fig, axs = plt.subplots(1, 2, figsize=(10, 3))

       # Tampilkan gambar
       axs[0].imshow(bright, cmap='gray')
       axs[0].set_title("Brightnes")

       # Tampilkan histogram
       axs[1].hist(bright.ravel(), bins=256, range=[0, 256], color='steelblue')
       axs[1].set_title("Histogram Brightnes")

[112]: Text(0.5, 1.0, 'Histogram Brightnes')
```

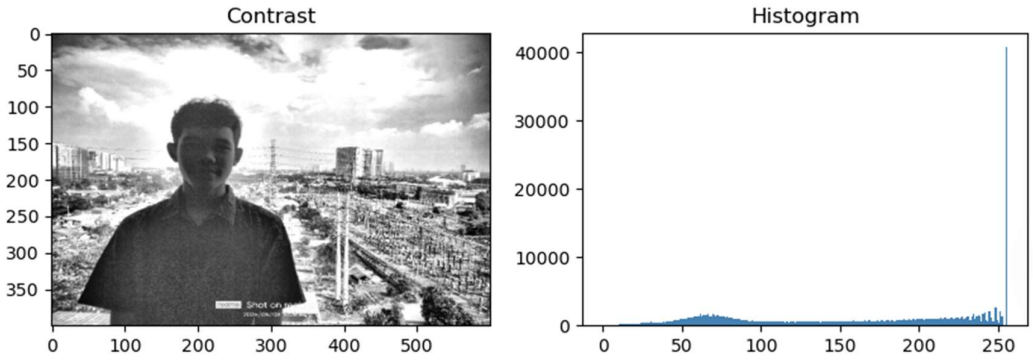


7. Berikutnya fungsi cv2.createCLAHE() fungsi ini Membuat objek CLAHE (Contrast Limited Adaptive Histogram Equalization), teknik untuk meningkatkan kontras gambar secara lokal. Selanjutnya perintah clipLimit=3.0: ini fungsinya Batas maksimum untuk kontras agar tidak terlalu berlebihan. Dan fungsi tileGridSize=(8, 8) ini berfungsi supaya Gambar dibagi menjadi blok 8x8, dan histogram equalization dilakukan di masing-masing blok. Lalu fungsi apply(bright) untuk Menerapkan CLAHE pada gambar yang telah ditingkatkan kecerahannya. lalu Menampilkan gambar hasil peningkatan kontras dan histogram hasilnya.

Menampilkan gambar Contrast dan Histogram

```
[130]: clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))

contrast = clahe.apply(bright)
fig, axs = plt.subplots(1, 2, figsize=(10, 3))
axs[0].imshow(contrast, cmap='gray')
axs[0].set_title("Contrast")

# Tampilkan histogram
axs[1].hist(contrast.ravel(), bins=256, range=[0, 256], color='steelblue')
axs[1].set_title("Histogram")
```

```
[130]: Text(0.5, 1.0, 'Histogram')
```



BAB IV
PENUTUP

4.1 Kesimpulan

1. Proses deteksi warna merah, hijau, dan biru pada citra berhasil dilakukan dengan pendekatan thresholding menggunakan rentang HSV, dan hasilnya dapat ditampilkan secara terpisah.
2. Penentuan nilai ambang batas dilakukan dengan memilih rentang HSV yang tepat, sehingga warna pada gambar dapat diklasifikasikan dan ditampilkan secara selektif.
3. Proyek ini juga berhasil menerapkan teknik peningkatan kualitas citra backlight dengan metode grayscale, brightness adjustment, dan CLAHE (Contrast Limited Adaptive Histogram Equalization), yang mampu memperjelas objek utama (profil wajah) tanpa mengorbankan kualitas latar belakang.
4. Histogram yang dihasilkan dari setiap tahap pengolahan membantu dalam menganalisis distribusi intensitas piksel dan menunjukkan keberhasilan perbaikan visual pada citra.
5. Praktikum ini menunjukkan pentingnya pemahaman dasar teori pengolahan citra dalam menerapkan algoritma yang efektif untuk berbagai kondisi pencahayaan dan kebutuhan deteksi warna.

DAFTAR PUSTAKA

hou, Y., & Wang, H. (2021). "Color image segmentation using HSV color space and region-growing method." Journal of Visual Communication and Image Representation, 73, 102949.

Dibya, J. B., & Thakur, R. S. (2022). "An Efficient Technique for Medical Image Enhancement Based on Interval Type-2 Fuzzy Set Logic." Progress in Computational Analysis and Network, 710, 667–678.

Toet, A. (2020). "Methods in quantitative image analysis." Histochemistry and Cell Biology, 154(2), 123–135.

Zhou, Y., & Wang, H. (2021). Color image segmentation using HSV color space and region-growing method. Journal of Visual Communication and Image Representation, 73, 102949.

Singh, K., & Kapoor, R. (2020). Image enhancement using exposure fusion technique. Procedia Computer Science, 167, 2462–2469.

Kumari, V., & Patel, D. (2021). Comparative study of different techniques used for image enhancement. Materials Today: Proceedings, 44, 3957–3962.