UTS

PENGOLAHAN CITRA

NAMA : Antonio Gerald Renjann

NIM : 20233289

KELAS : D

DOSEN : Ir. Darma Rusjdi, M.Kom

NO.PC : 28

ASISTEN : 1. Clarenca Sweetdiva Pereira

2. Fakhrul Fauzi Nugraha Tarigan

3. Sakura Amastasya Salsabila Setiyanto

4. Muhammad Hanief Febriansyah

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2024/2025

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

- Bagaimana cara mendeteksi warna tertentu pada citra digital menggunakan teknik pengolahan citra?
- Bagaimana metode untuk mencari dan mengurutkan ambang batas pada citra dari nilai terkecil hingga terbesar?
- Bagaimana teknik yang dapat digunakan untuk memperbaiki kualitas gambar yang mengalami pencahayaan backlight?

1.2 Tujuan Masalah

- Untuk menerapkan metode deteksi warna dalam citra digital guna mengenali objek atau bagian tertentu berdasarkan warna.
- Untuk memahami proses pencarian dan pengurutan ambang batas dalam citra guna keperluan segmentasi atau analisis lebih lanjut.
- Untuk meningkatkan kualitas visual dari gambar backlight dengan menerapkan teknik perbaikan pencahayaan.

1.3 Manfaat Masalah

- Memberikan pemahaman praktis kepada mahasiswa tentang cara kerja deteksi warna pada citra digital.
- Menambah wawasan mengenai pentingnya proses thresholding (pengambangan) dalam analisis citra.
- Membantu mahasiswa memahami dan mengaplikasikan metode perbaikan pencahayaan gambar sehingga gambar yang kurang optimal dapat ditingkatkan kualitasnya.

BAB II

LANDASAN TEORI

1. Deteksi Warna pada Citra

Deteksi warna adalah salah satu teknik dasar dalam pengolahan citra digital yang berfungsi untuk mengenali dan mengekstraksi bagian citra berdasarkan informasi warna. Warna merupakan atribut visual penting yang digunakan dalam banyak aplikasi, seperti pengenalan objek, pelacakan objek bergerak (object tracking), pengklasifikasian buah atau sayuran berdasarkan tingkat kematangan, hingga sistem pengawasan otomatis seperti deteksi lampu lalu lintas atau plat nomor.

Citra digital biasanya tersimpan dalam model warna **RGB (Red, Green, Blue)**. Meskipun umum digunakan dalam tampilan layar, model RGB tidak ideal untuk proses segmentasi warna karena perubahan intensitas cahaya sangat memengaruhi hasilnya. Oleh karena itu, dalam deteksi warna, citra RGB sering dikonversi ke model warna **HSV (Hue, Saturation, Value)**. Model HSV memisahkan informasi warna (hue) dari intensitas cahaya (value), sehingga lebih stabil terhadap perubahan pencahayaan dan lebih menyerupai cara manusia memandang warna.

Langkah-langkah dalam Deteksi Warna:

1. **Konversi Model Warna:** Citra dikonversi dari format RGB ke HSV atau format lain yang sesuai.
2. **Penentuan Rentang Warna:** Menentukan nilai minimal dan maksimal dari hue, saturation, dan value yang sesuai dengan warna yang ingin dideteksi.
3. **Pembuatan Masker Warna:** Dengan menggunakan fungsi logika (seperti inRange di OpenCV), masker warna dibuat untuk menyaring bagian citra yang sesuai dengan rentang warna.
4. **Ekstraksi Objek:** Menggunakan masker, bagian citra yang memiliki warna sesuai dapat dipisahkan dan dianalisis lebih lanjut.

Aplikasi praktis dari deteksi warna sangat luas, terutama dalam sistem visi komputer dan otomatisasi industri. Selain itu, deteksi warna juga digunakan sebagai dasar dalam proses segmentasi lanjutan dan pelacakan objek dalam video.

2. Mencari dan Mengurutkan Ambang Batas (Thresholding)

Thresholding atau pengambangan adalah metode segmentasi citra yang memisahkan objek dari latar belakang berdasarkan tingkat intensitas piksel. Tujuan utama thresholding adalah untuk menghasilkan citra biner, yaitu citra yang hanya memiliki dua warna: putih (representasi objek) dan hitam (representasi latar belakang).

Thresholding sangat berguna dalam proses identifikasi bentuk, perhitungan objek, dan ekstraksi fitur pada citra. Dengan thresholding, proses pengolahan citra menjadi lebih efisien karena data yang diolah lebih sederhana.

Jenis Thresholding:

1. **Thresholding Sederhana (Global Thresholding):**

- Menggunakan satu nilai ambang (threshold) untuk seluruh citra.
- Contoh: Jika nilai ambang adalah 127, maka semua piksel dengan nilai >127 menjadi putih, dan sisanya menjadi hitam.

2. **Otsu Thresholding:**
   - Metode otomatis untuk menentukan ambang batas optimal dengan cara meminimalkan variansi dalam kelas antar piksel.
   - Sangat efektif untuk citra dengan histogram bimodal (dua puncak intensitas).
3. **Adaptive Thresholding:**
   - Menggunakan ambang batas lokal berdasarkan nilai piksel di sekitar titik tertentu.
   - Cocok untuk citra dengan pencahayaan yang tidak merata.
4. **Multi-Level Thresholding:**
   - Digunakan ketika citra memiliki lebih dari dua kelas objek.
   - Memerlukan beberapa ambang batas, yang nantinya akan diurutkan dari yang terkecil hingga terbesar.

Proses Pencarian dan Pengurutan Ambang Batas:

- Setelah nilai-nilai threshold ditemukan (baik secara manual maupun otomatis), langkah selanjutnya adalah mengurutkannya.
- Pengurutan ini penting, terutama untuk citra yang membutuhkan klasifikasi bertingkat seperti segmentasi multi-objek atau pewarnaan ulang berdasarkan kelas.

Thresholding merupakan dasar dalam banyak algoritma pengolahan citra lainnya seperti deteksi tepi, segmentasi bentuk, dan klasifikasi objek.

3. Perbaikan Gambar Backlight

Gambar backlight adalah gambar yang memiliki pencahayaan dominan dari arah belakang objek, sehingga objek utama tampak gelap atau bahkan tidak terlihat dengan jelas. Kondisi backlight menyebabkan ketidakseimbangan pencahayaan antara latar belakang dan objek utama, yang dapat mempersulit analisis atau klasifikasi citra.

Perbaikan citra backlight termasuk dalam proses **image enhancement** atau peningkatan kualitas citra. Tujuannya adalah memperjelas detail pada objek utama dan menyeimbangkan distribusi cahaya agar informasi visual pada citra lebih mudah dianalisis.

Teknik-teknik Perbaikan Gambar Backlight:

1. **Histogram Equalization:**
   - Meratakan histogram citra grayscale agar semua nilai intensitas memiliki distribusi yang lebih merata.

- Membantu meningkatkan kontras pada citra gelap atau terlalu terang.

2. **CLAHE (Contrast Limited Adaptive Histogram Equalization):**
    - Teknik penyempurnaan dari histogram equalization.
    - Bekerja secara lokal dengan membagi citra menjadi blok-blok kecil (tiles) dan meningkatkan kontras masing-masing blok.
    - Membatasi kontras maksimal untuk menghindari noise berlebih.
3. **Gamma Correction:**
    - Mengubah intensitas citra menggunakan fungsi pangkat (eksponensial).
    - Cocok untuk menyesuaikan pencahayaan pada citra gelap atau terang secara halus.
4. **Logarithmic Transformation:**
    - Meningkatkan intensitas piksel gelap dengan cepat tanpa membuat area terang menjadi terlalu terang.
    - Menggunakan fungsi logaritma sebagai dasar transformasi.
5. **Unsharp Masking dan Filtering:**
    - Teknik yang digunakan untuk mempertajam detail pada objek yang tadinya kabur atau gelap akibat backlight.

BAB III

HASIL

Deteksi Warna Pada Citra

[362]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

[363]:
```
img = cv2.imread('nama .jpg')
```

```
The history saving thread hit an unexpected error (OperationalError('database or disk is full')).History will not be written to the database.
```

[364]:
```
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

[365]:
```
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

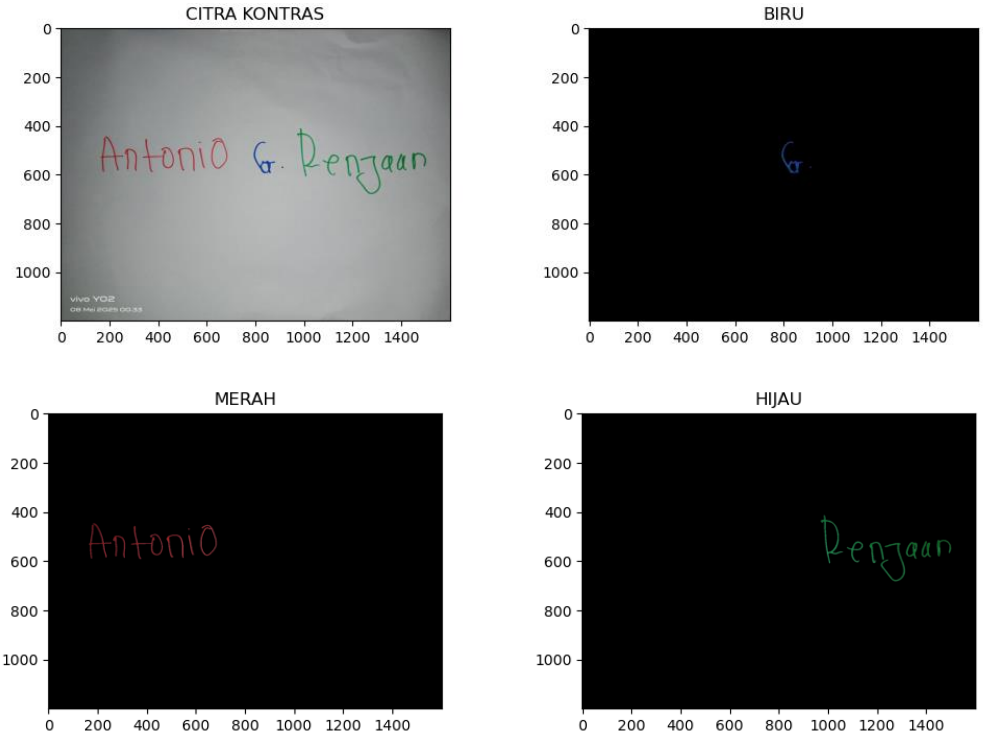
[366]:
```
lower_blue = np.array([100, 100, 50])
upper_blue = np.array([140, 255, 255])
mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)
result_blue = cv2.bitwise_and(img, img, mask=mask_blue)
```

[367]:
```
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])
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)
result_red = cv2.bitwise_and(img, img, mask=mask_red)
```


[368]:
```
lower_green = np.array([40, 100, 50])
upper_green = np.array([80, 255, 255])
mask_green = cv2.inRange(hsv, lower_green, upper_green)
result_green = cv2.bitwise_and(img, img, mask=mask_green)
```

[369]:
```
blue_rgb = cv2.cvtColor(result_blue, cv2.COLOR_BGR2RGB)
red_rgb = cv2.cvtColor(result_red, cv2.COLOR_BGR2RGB)
green_rgb = cv2.cvtColor(result_green, cv2.COLOR_BGR2RGB)
```

[370]:
```
fig, axs = plt.subplots(2, 2, figsize=(12, 8))
axs[0, 0].imshow(img_rgb); axs[0, 0].set_title('CITRA KONTRAS')
axs[0, 1].imshow(blue_rgb); axs[0, 1].set_title('BIRU')
axs[1, 0].imshow(red_rgb); axs[1, 1].set_title('MERAH')
axs[1, 1].imshow(green_rgb); axs[1, 1].set_title('HIJAU')
```



```
[371]: for ax in axs.flat:
           ax.axis('off')

[372]: plt.tight_layout()
       plt.show()
```

<Figure size 640x480 with 0 Axes>

Diatas adalah codingan untuk mendeteksi warna pada citra gambar yang dimana diawali dengan:

1. Import library untuk pemrosesan citra (cv2), perhitungan numerik (numpy), dan visualisasi grafik (matplotlib).
2. Membaca gambar dari file bernama 'nama.jpg'.
3. □ img_rgb: Konversi gambar dari BGR ke RGB untuk ditampilkan dengan matplotlib.
   □ hsv: Konversi gambar dari BGR ke HSV untuk memudahkan deteksi warna.
4. □ Mendefinisikan rentang HSV untuk warna biru.
   □ Membuat *mask* (filter) yang hanya melewatkan piksel dalam rentang biru.
   □ Menampilkan hanya bagian biru dari gambar.
5. □ Karena warna merah terletak di ujung rentang hue (0–10 dan 160–180), maka diperlukan dua rentang.
   □ Kedua mask digabung dengan bitwise_or.
6. Mendeteksi area berwarna hijau dalam gambar.
7. □ Membuat subplot 2x2.
   □ Menampilkan gambar asli dan hasil deteksi warna secara terpisah.

- CITRA KONTRAS: Gambar asli yang berisi tulisan berwarna.
- BIRU: Hanya menampilkan huruf "G." yang berwarna biru.
- MERAH: Menampilkan tulisan "Antonio" yang berwarna merah.
- HIJAU: Menampilkan tulisan "Rengaan" yang berwarna hijau.

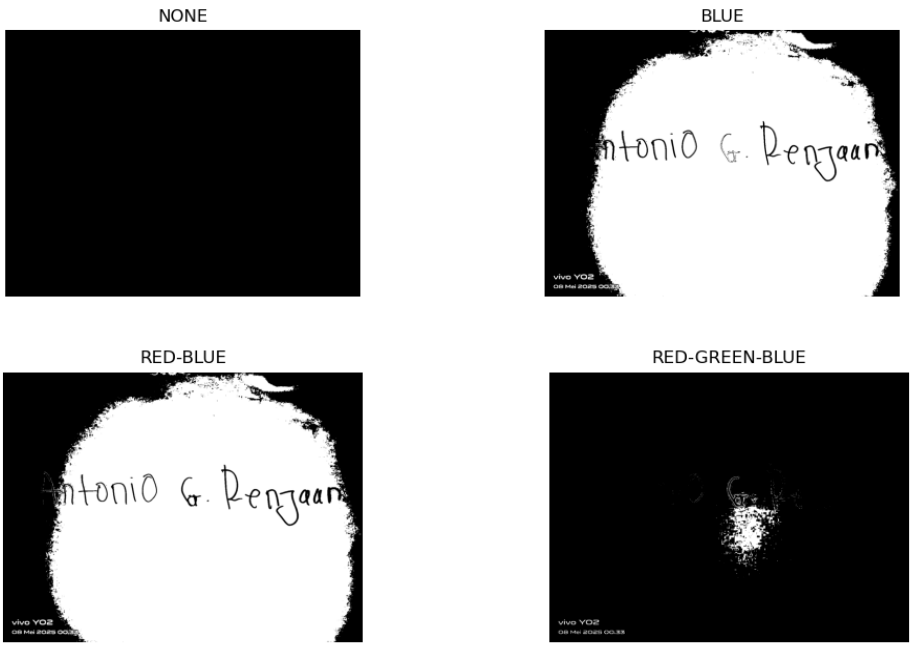
Mencari Dan Mengurutkan Ambang Batas Terkecil Sampai Dengan Terbesar

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

[374]: img_path = 'nama .jpg'
       img = cv2.imread(img_path)

[375]: if img is None:
           print("Gambar tidak ditemukan. Periksa kembali path file.")
       else:
           # 2. Konversi BGR ke RGB
           img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

           # 3. Pisahkan channel RGB
           r, g, b = img_rgb[:, :, 0], img_rgb[:, :, 1], img_rgb[:, :, 2]

           # 4. Hitung threshold Otsu dari masing-masing channel
           r_thresh_val, _ = cv2.threshold(r, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
           g_thresh_val, _ = cv2.threshold(g, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
           b_thresh_val, _ = cv2.threshold(b, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

           print("Threshold Merah  :", r_thresh_val)
           print("Threshold Hijau  :", g_thresh_val)
           print("Threshold Biru   :", b_thresh_val)

           # 5. NONE (kosong, semua hitam)
```

```
none_mask = np.zeros_like(r)

# 6. BLUE saja
_, blue_mask = cv2.threshold(b, b_thresh_val, 255, cv2.THRESH_BINARY)

# 7. RED-BLUE gabungan
rb = cv2.addWeighted(r.astype(np.uint8), 0.5, b.astype(np.uint8), 0.5, 0)
_, rb_mask = cv2.threshold(rb, (r_thresh_val + b_thresh_val) / 2, 255, cv2.THRESH_BINARY)

# 8. RED-GREEN-BLUE gabungan
rgb = cv2.addWeighted(r.astype(np.uint8), 1/3, g.astype(np.uint8), 1/3, 0)
rgb = cv2.addWeighted(rgb, 2/3, b.astype(np.uint8), 1/3, 0)
_, rgb_mask = cv2.threshold(rgb, (r_thresh_val + g_thresh_val + b_thresh_val) / 3, 255, cv2.THRESH_BINARY)

# 9. Tampilkan keempat gambar
fig, axs = plt.subplots(2, 2, figsize=(12, 6))

axs[0, 0].imshow(none_mask, cmap='gray')
axs[0, 0].set_title("NONE")

axs[0, 1].imshow(blue_mask, cmap='gray')
axs[0, 1].set_title("BLUE")

axs[1, 0].imshow(rb_mask, cmap='gray')
axs[1, 0].set_title("RED-BLUE")

axs[1, 1].imshow(rgb_mask, cmap='gray')
axs[1, 1].set_title("RED-GREEN-BLUE")
```



```
for ax in axs.flat:
    ax.axis('off')

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



Berikutnya adalah codingan untuk Mencari Dan Mengurutkan Ambang Batas Terkecil Sampai Dengan Terbesar yang dimana diawali dengan:

1. Membaca dan mengonversi gambar ke format RGB.
2. Memisahkan channel merah (R), hijau (G), dan biru (B).
3. Menghitung nilai ambang (threshold) untuk setiap channel dengan metode Otsu.
4. Membuat masker biner berdasarkan:
    - Hanya channel biru.
    - Gabungan merah-biru.

- Gabungan merah-hijau-biru.
- Masker kosong (semua hitam).

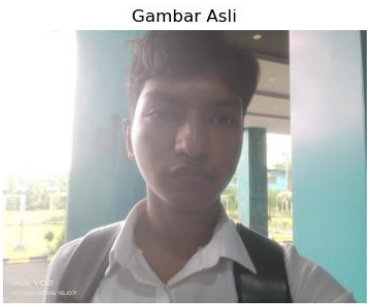
5 Menampilkan hasil dalam 4 subplot.

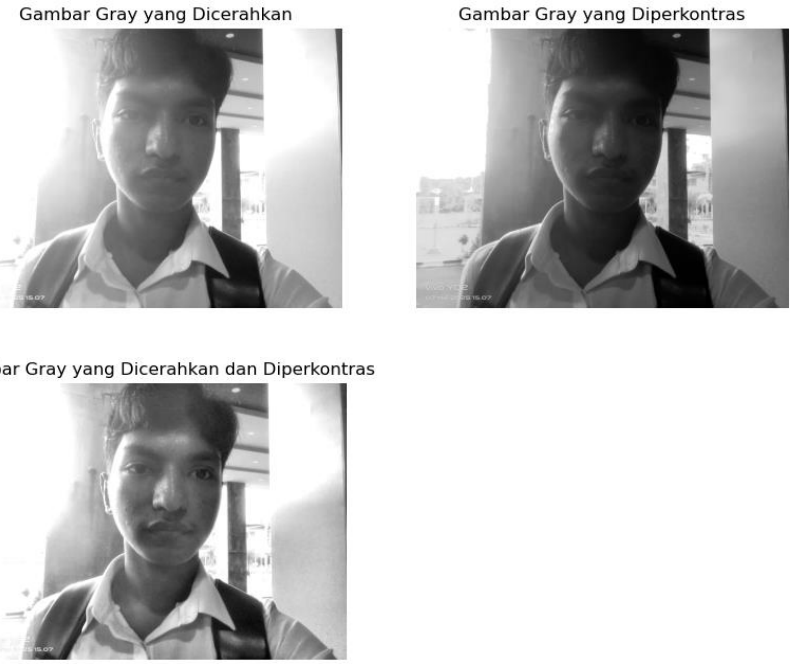
- NONE: Semua hitam (sesuai np.zeros_like()).
- BLUE: Menampilkan objek tulisan secara cukup jelas. Artinya, channel biru cukup dominan dalam membedakan latar belakang dan tulisan.
- RED-BLUE: Kombinasi ini tetap mempertahankan tulisan, namun sedikit lebih terang.
- RED-GREEN-BLUE: Gabungan ketiga channel menghasilkan segmentasi lebih selektif, tapi tulisan tampak lebih kabur, menunjukkan threshold ketat dari gabungan 3 channel.

Memperbaiki Gambar Backlight

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

[377]: img = cv2.imread("gerald.jpg")
       img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

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

[379]: bright = cv2.convertScaleAbs(gray, alpha=1, beta=50)

[380]: contrast = cv2.equalizeHist(gray)

[381]: clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
       bright_contrast = clahe.apply(bright)
```

```
[382]: plt.figure(figsize=(10, 12))

       plt.subplot(3, 2, 1)
       plt.imshow(img_rgb)
       plt.title("Gambar Asli")
       plt.axis("off")

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

       plt.subplot(3, 2, 3)
       plt.imshow(bright, cmap='gray')
       plt.title("Gambar Gray yang Dicerahkan")
       plt.axis("off")

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

       plt.subplot(3, 2, 5)
       plt.imshow(bright_contrast, cmap='gray')
       plt.title("Gambar Gray yang Dicerahkan dan Diperkontras")
       plt.axis("off")
```




```
[383]: plt.tight_layout()
       plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```

Yang ketiga ini adalah codingan untuk Memperbaiki Gambar Backlight yang diawali dengan:

1. Import Library

Mengimpor pustaka yang dibutuhkan: OpenCV (cv2), NumPy untuk operasi array, dan matplotlib untuk menampilkan gambar.

2. Baca dan Konversi Gambar

- Membaca gambar asli.
- OpenCV menggunakan format BGR secara default, jadi dikonversi ke RGB agar cocok saat ditampilkan dengan matplotlib.

3. Ubah ke Grayscale

Mengubah gambar berwarna menjadi grayscale (abu-abu).

4. Terangkan Gambar

Meningkatkan kecerahan gambar grayscale dengan menambahkan beta=50.

5. Tingkatkan Kontras

Menggunakan histogram equalization untuk meningkatkan kontras gambar grayscale.

6. Gabungan Kecerahan + Kontras (CLAHE)

Menggunakan CLAHE (Contrast Limited Adaptive Histogram Equalization):

- Teknik ini membagi gambar menjadi blok kecil lalu meningkatkan kontras lokalnya tanpa menyebabkan over-eksposur (cocok untuk gambar dengan backlight).

7. Menampilkan Semua Gambar

Tiap plt.subplot digunakan untuk menampilkan hasil dari setiap tahap pemrosesan.

- Gambar Asli: Terlihat gelap di bagian wajah karena backlight.
- Gambar Gray: Konversi ke abu-abu tanpa perbaikan apapun.
- Gambar Gray yang Dicerahkan: Lebih terang, tapi masih kurang detail.
- Gambar Gray yang Diperkontras: Kontras lebih tinggi, wajah terlihat lebih tajam.
- Gambar Gray yang Dicerahkan dan Diperkontras: Hasil terbaik, wajah terlihat jelas dan latar tidak terlalu mengganggu.

BAB IV

PENUTUP

Melalui praktikum ini, diperoleh pemahaman bahwa deteksi warna dapat digunakan untuk mengenali objek berdasarkan karakteristik warnanya, thresholding bermanfaat untuk memisahkan objek dari latar belakang menggunakan nilai ambang, dan perbaikan gambar backlight mampu meningkatkan kualitas citra yang pencahayaannya tidak merata. Ketiga teknik ini merupakan dasar penting dalam pengolahan citra digital yang dapat diaplikasikan dalam berbagai bidang seperti pengenalan objek, segmentasi, dan peningkatan kualitas visual.

DAFTAR PUSTAKA

1. Rochim, F. N., Sompie, G. D., Mu'ammar, M., Saputra, R. I., & Rosyani, P. (2024). *Perancangan Sistem Deteksi Warna Real-Time Menggunakan Metode Gaussian Blur dan Ruang Warna HSV*. Biner: Jurnal Ilmu Komputer, Teknik dan Multimedia, 2(2), 178–183.
2. Maulana, A. (2024). *Implementasi OpenCV dengan Metode Image Thresholding pada Segmentasi Gambar Digital*. Jurnal AI dan SPK: Jurnal Artificial Intelligent dan Sistem Penunjang Keputusan, 2(1), 27–32.
3. Ridho'i, A. (2022). *Penerapan Pengolahan Citra untuk Perbaikan Gambar 2 Dimensi dengan Menggunakan MATLAB*. Jurnal Informatika, 8(2), 45–52.
4. Setiawan, W. D. H. A. S. (2019). *Pengolah Citra dengan Metode Thresholding dengan MATLAB R2014A*. Jurnal Media Infotama, 15(2), 123–130.
5. Masril, M. A., & Noviardi, R. (2020). *Analisa Morfologi Dilasi untuk Perbaikan Kualitas Citra Deteksi Tepi pada Pola Batik Menggunakan Operator Prewitt dan Laplacian of Gaussian*. Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), 4(6), 1052–1059.