

UTS
PENGOLAHAN CITRA



NAMA : Putri Azzahra

NIM : 202331080

KELAS : B

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

NO.PC : 18

ASISTEN : 1. Davina Najwa Ermawan
2. Fakhrol Fauzi Nugraha Tarigan
3. Viana Salsabila Fairuz Syahla
4. Muhammad Hanief Febriansyah

INSTITUT TEKNOLOGI PLN
TEKNIK INFORMATIKA
2024/2025

DAFTAR ISI

Contents

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
2.1 Deteksi Warna pada Citra.....	4
2.2 Histogram Citra dan Analisisnya.....	4
2.3 Thresholding untuk Deteksi Warna	4
2.4 Perbaikan Gambar Backlight.....	5
BAB III.....	6
HASIL	6
3.1 Foto Pertama	6
3.2 Soal no 1 (Deteksi Warna)	7
3.3 Soal no 2 (Ambang Batas).....	9
3.4 soal no 3 (Backlight).....	12
BAB IV.....	16
PENUTUP	16
DAFTAR PUSTAKA	17

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

1. Bagaimana cara mendeteksi warna merah, hijau, dan biru pada citra digital menggunakan Python?
2. Bagaimana cara menentukan nilai ambang batas (threshold) yang tepat untuk memisahkan tiap warna pada citra?
3. Bagaimana karakteristik histogram dari citra hasil segmentasi warna dan apa maknanya terhadap distribusi warna dalam citra tersebut?
4. Bagaimana cara meningkatkan kualitas citra yang mengalami efek backlight agar objek utama lebih jelas terlihat?

1.2 Tujuan Masalah

1. Untuk memahami dan mengimplementasikan proses deteksi warna dasar (merah, hijau, biru) pada citra menggunakan pemrograman Python.
2. Untuk menentukan nilai ambang batas yang sesuai dalam segmentasi warna menggunakan teknik thresholding.
3. Untuk menganalisis histogram hasil segmentasi warna agar dapat memahami distribusi intensitas warna pada citra.
4. Untuk mengembangkan teknik pengolahan citra dalam memperbaiki gambar backlight, sehingga objek utama menjadi lebih menonjol.

1.3 Manfaat Masalah

1. Memahami prinsip dasar deteksi warna menggunakan model RGB dan thresholding.
2. Meningkatkan keterampilan analisis visual terhadap hasil histogram citra.
3. Meningkatkan pemahaman teknis dalam menerapkan teori pengolahan citra ke dalam praktik berbasis kode.
4. Mampu mengatasi permasalahan umum dalam fotografi seperti backlight melalui pendekatan pemrosesan citra.

BAB II

LANDASAN TEORI

2.1 Deteksi Warna pada Citra

Ruang warna RGB merupakan representasi warna yang paling umum dalam citra digital, yang terdiri dari tiga kanal warna: Red (Merah), Green (Hijau), dan Blue (Biru). Setiap piksel di dalam citra digital memiliki nilai intensitas pada ketiga kanal tersebut, dengan rentang nilai 0–255.

Dalam praktik deteksi warna, kita menggunakan teknik thresholding pada nilai R, G, dan B untuk memisahkan warna yang diinginkan. Misalnya:

- Untuk mendeteksi merah, nilai kanal R harus tinggi, sementara G dan B rendah.
- Untuk mendeteksi hijau, nilai G dominan dan R serta B rendah.
- Untuk mendeteksi biru, nilai B tinggi, dan R serta G rendah.

2.2 Histogram Citra dan Analisisnya

Histogram pada citra adalah representasi grafis dari distribusi frekuensi nilai intensitas piksel dalam sebuah gambar. Untuk gambar berwarna, histogram dapat dibuat untuk masing-masing kanal (R, G, B), sedangkan untuk gambar grayscale, hanya satu histogram diperlukan.

Histogram berfungsi untuk:

- Mengetahui distribusi terang–gelap dalam citra.
- Menganalisis kontras.
- Mengetahui dominasi warna dalam suatu objek.

Analisis Visual Histogram

- Jika histogram menyebar luas (dari nilai rendah ke tinggi), maka gambar memiliki kontras tinggi.
- Jika histogram terkonsentrasi di satu sisi, maka gambar cenderung terlalu gelap (kiri) atau terlalu terang (kanan).

2.3 Thresholding untuk Deteksi Warna

a. Pengertian Thresholding

Thresholding adalah teknik segmentasi citra yang memisahkan piksel berdasarkan nilai intensitas. Dalam konteks warna, kita menetapkan rentang nilai pada kanal R, G, dan B untuk mengisolasi warna tertentu dari citra.

b. Pemilihan Threshold

Nilai ambang batas biasanya ditentukan secara empiris dengan:

- Mencoba berbagai nilai hingga warna target terlihat jelas.
- Menggunakan bantuan histogram kanal RGB.

Metode thresholding warna RGB berbeda dari thresholding grayscale. Pada citra grayscale, metode seperti Otsu's Method dapat digunakan untuk menentukan nilai ambang secara otomatis, tetapi untuk citra berwarna, thresholding dilakukan secara manual untuk masing-masing kanal.

2.4 Perbaikan Gambar Backlight

a. Masalah Backlight

Efek backlight terjadi ketika sumber cahaya lebih terang berada di belakang objek, sehingga wajah atau objek utama tampak gelap. Untuk memperbaiki citra ini, kita perlu menerapkan:

- Konversi ke grayscale
- Penyesuaian brightness (kecerahan)
- Penyesuaian contrast (kontras)

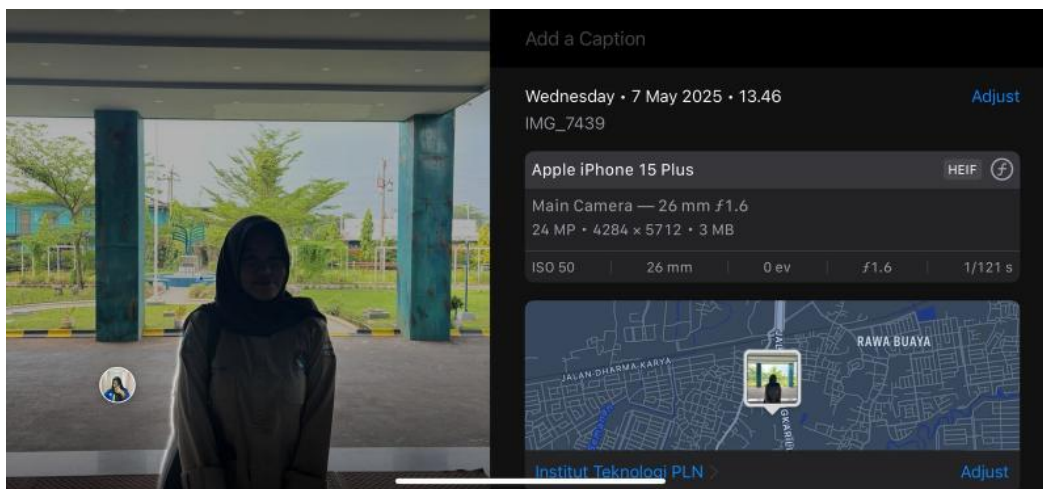
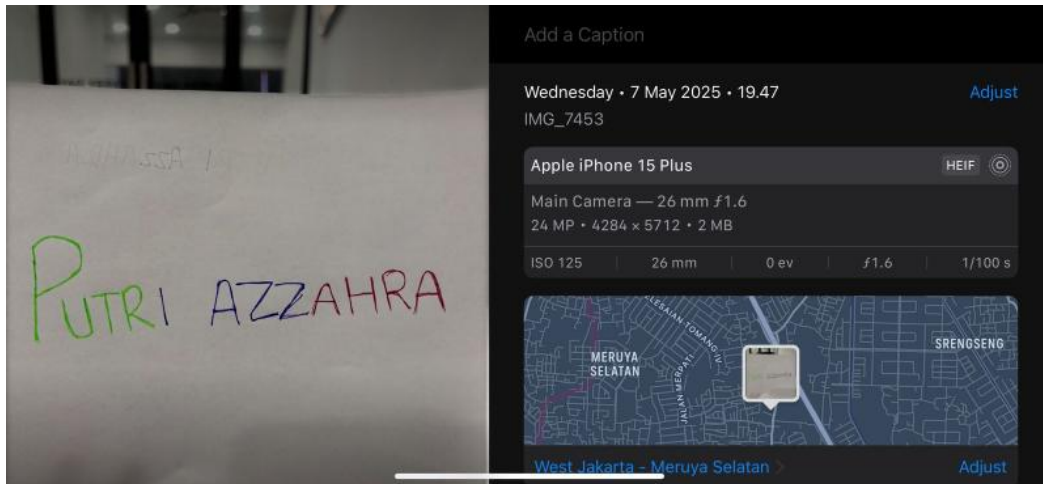
b. Teknik Perbaikan

- Grayscale: Mengubah citra warna menjadi intensitas tunggal untuk memudahkan proses peningkatan.
- Brightness Adjustment: Menambahkan nilai konstan ke setiap piksel untuk mencerahkan citra.
- Contrast Stretching: Mengubah rentang intensitas agar distribusi piksel menyebar lebih luas.
- Histogram Equalization: Teknik yang meratakan histogram untuk meningkatkan visibilitas area gelap.

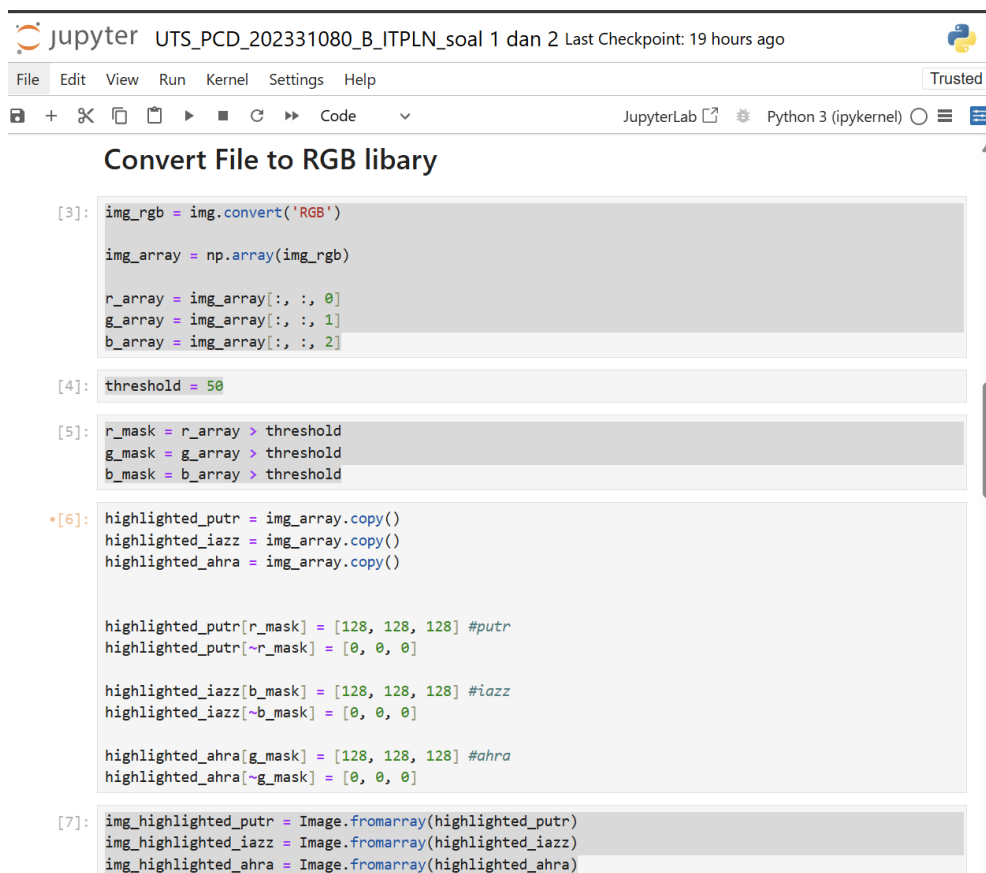
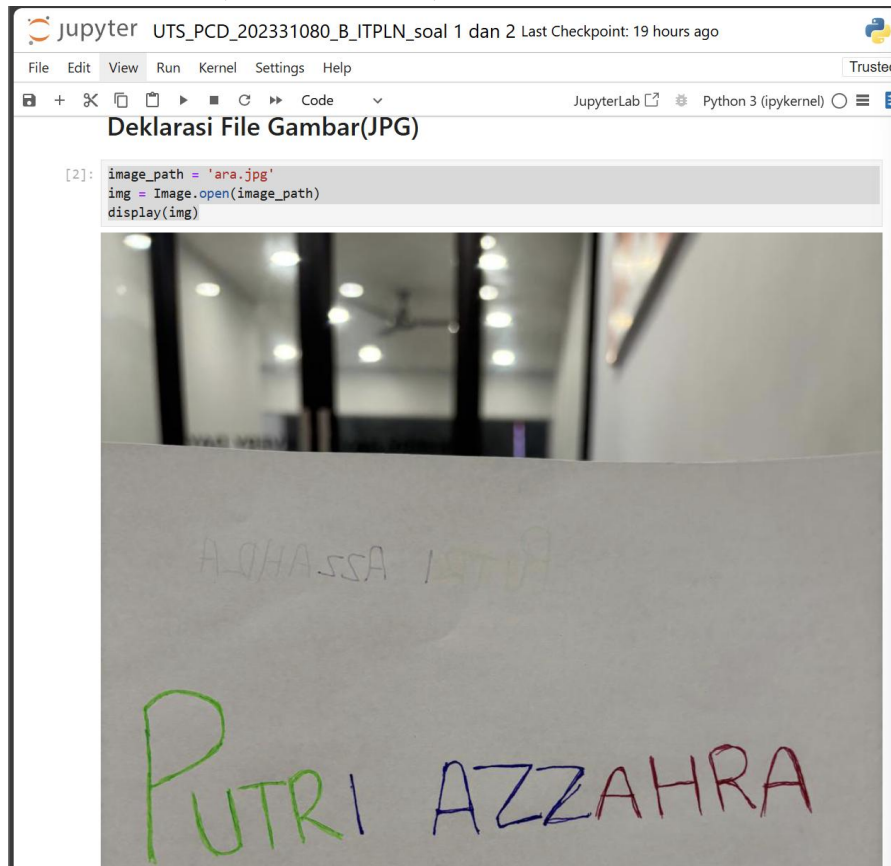
BAB III

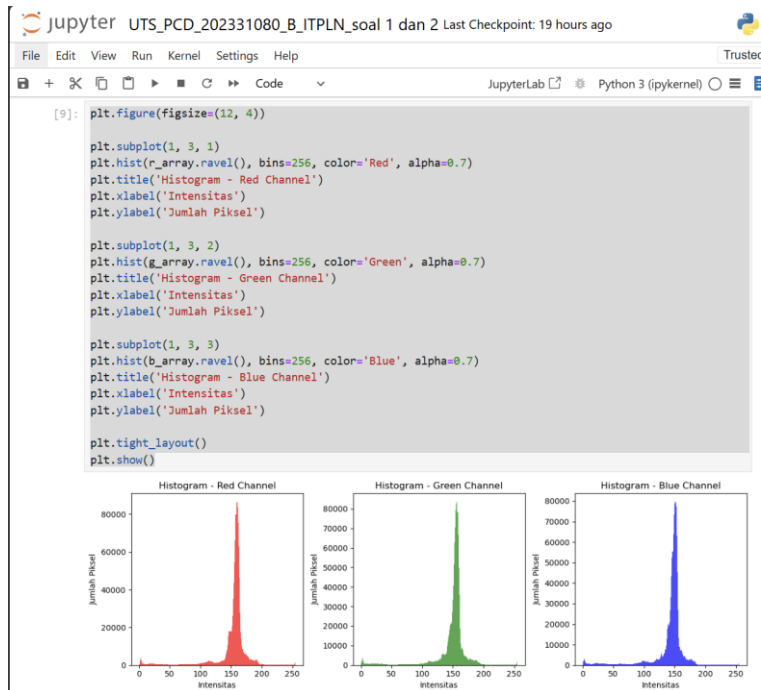
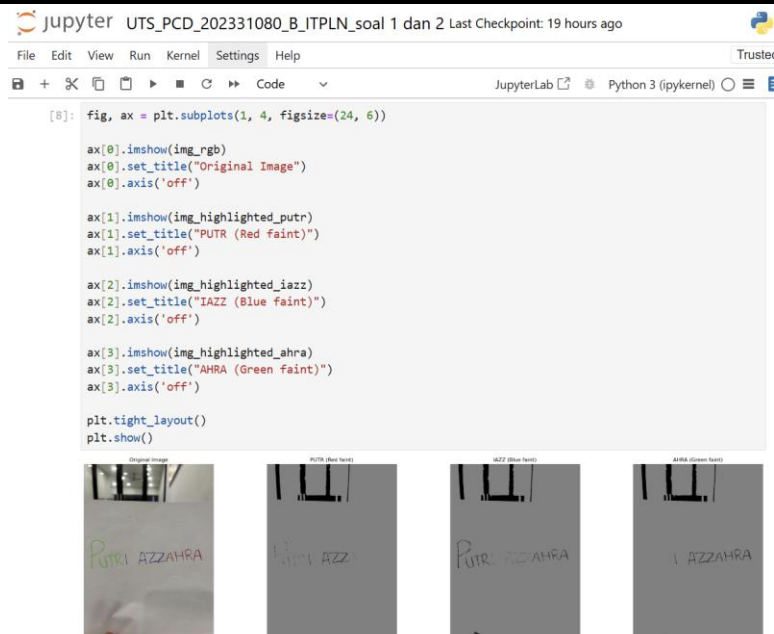
HASIL

3.1 Foto Pertama



3.2 Soal no 1 (Deteksi Warna)





Jupyter UTS_PCD_202331080_B_ITPLN_soal 1 dan 2 Last Checkpoint: 19 hours ago

```

[9]: none_mask = ~(r_mask | g_mask | b_mask)
blue_mask = b_mask & ~(r_mask | g_mask)
red_blue_mask = r_mask & b_mask & ~g_mask
red_green_blue_mask = r_mask & g_mask & b_mask

[10]: highlighted_none = img_array.copy()
highlighted_none[~none_mask] = [0, 0, 0]

highlighted_blue = img_array.copy()
highlighted_blue[~blue_mask] = [0, 2, 2]

highlighted_red_blue = img_array.copy()
highlighted_red_blue[~red_blue_mask] = [0, 2, 2]

highlighted_red_green_blue = img_array.copy()
highlighted_red_green_blue[~red_green_blue_mask] = [0, 0, 0]

[11]: img_highlighted_none = Image.fromarray(highlighted_none)
img_highlighted_blue = Image.fromarray(highlighted_blue)
img_highlighted_red_blue = Image.fromarray(highlighted_red_blue)
img_highlighted_red_green_blue = Image.fromarray(highlighted_red_green_blue)

```


Penjelasan:

- Membuka dan Mengonversi Gambar dari file 'ara.jpg' dibuka menggunakan PIL, kemudian dikonversi menjadi format RGB dengan `img.convert('RGB')` untuk memisahkan saluran warna.
- Ekstraksi Saluran Warna dari gambar yang telah diubah menjadi array NumPy, memisahkan saluran merah, hijau, dan biru (R, G, B) dalam array terpisah (`r_array`, `g_array`, `b_array`), memungkinkan pemrosesan intensitas warna secara independen.
- Penerapan Threshold dan Pembuatan Mask untuk setiap saluran warna, kode ini menerapkan threshold (nilai ambang batas 50) untuk menghasilkan mask. Mask ini berfungsi menandai piksel dengan intensitas lebih tinggi dari threshold untuk masing-masing warna (merah, hijau, biru).
- Highlighting Piksel dengan Masking berdasarkan mask yang dibuat, kode ini memodifikasi gambar dengan menyorot area tertentu. Piksel yang memenuhi syarat (intensitas warna lebih besar dari threshold) diberi warna abu-abu ([128, 128, 128]), sementara piksel lainnya diubah menjadi hitam ([0, 0, 0]). Ini menghasilkan tiga gambar yang menyoroti area dengan warna yang berbeda berdasarkan saluran warna (merah, hijau, biru).
- Menampilkan Hasil Gambar asli dan gambar hasil highlight (untuk masing-masing saluran warna: merah, hijau, biru) ditampilkan dalam satu plot dengan empat subplots untuk membandingkan perbedaan antara gambar asli dan gambar yang telah diproses.
- Histogram Saluran Warna dibuat untuk setiap saluran warna (merah, hijau, biru) untuk memvisualisasikan distribusi intensitas warna di dalam gambar. Ini membantu untuk memahami bagaimana intensitas warna tersebar di seluruh gambar.
- Penerapan Masking Lanjutan selain mask untuk masing-masing saluran, kode ini juga membuat mask gabungan untuk area yang tidak memenuhi syarat warna apapun, hanya biru, kombinasi merah dan biru, serta kombinasi ketiga saluran. Gambar hasil kombinasi mask ini juga ditampilkan.

3.3 Soal no 2 (Ambang Batas)



```

jupyter UTS_PCD_202331080_B_ITPLN_soal 1 dan 2 Last Checkpoint: 20 hours ago
File Edit View Run Kernel Settings Help Trusted
+ + + + + Code Python 3 (spykernel)

2. Ambang batas dari yang terkecil sampai terbesar

[13]: import cv2
img = cv2.imread('ara.jpg') # Gambar dibaca dalam format BGR
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

[14]: highlight_blue = np.zeros_like(img_rgb)
highlight_green = np.zeros_like(img_rgb)
highlight_red = np.zeros_like(img_rgb)

# Ambil channel
r = img_rgb[:, :, 0]
g = img_rgb[:, :, 1]
b = img_rgb[:, :, 2]

# Buat masker berdasarkan dominasi warna
mask_blue = (b > 100) & (b > r + 5) & (b > g + 5)
mask_green = (g > 100) & (g > r + 30) & (g > b + 30)
mask_red = (r > 100) & (r > g + 30) & (r > b + 30)

# Terapkan ke masing-masing channel
highlight_blue[mask_blue] = [0, 0, 255] # Hanya biru
highlight_green[mask_green] = [0, 255, 0] # Hanya hijau
highlight_red[mask_red] = [255, 0, 0] # Hanya merah

# Gabungan sesuai perintah
highlight_none = img_rgb.copy()
highlight_red_blue = cv2.add(highlight_red, highlight_blue)
highlight_red_green_blue = cv2.add(highlight_red, cv2.add(highlight_green, highlight_blue))
  
```

```

jupyter UTS_PCD_202331080_B_ITPLN_soal 1 dan 2 Last Checkpoint: 20 hours ago
File Edit View Run Kernel Settings Help Trusted
JupyterLab Python 3 (ipykernel)

# Plotting
fig, ax = plt.subplots(2, 2, figsize=(12, 10))

ax[0, 0].imshow(highlight_none)
ax[0, 0].set_title("None (Original)")

ax[0, 1].imshow(highlight_blue)
ax[0, 1].set_title("Blue (IAZZ)")

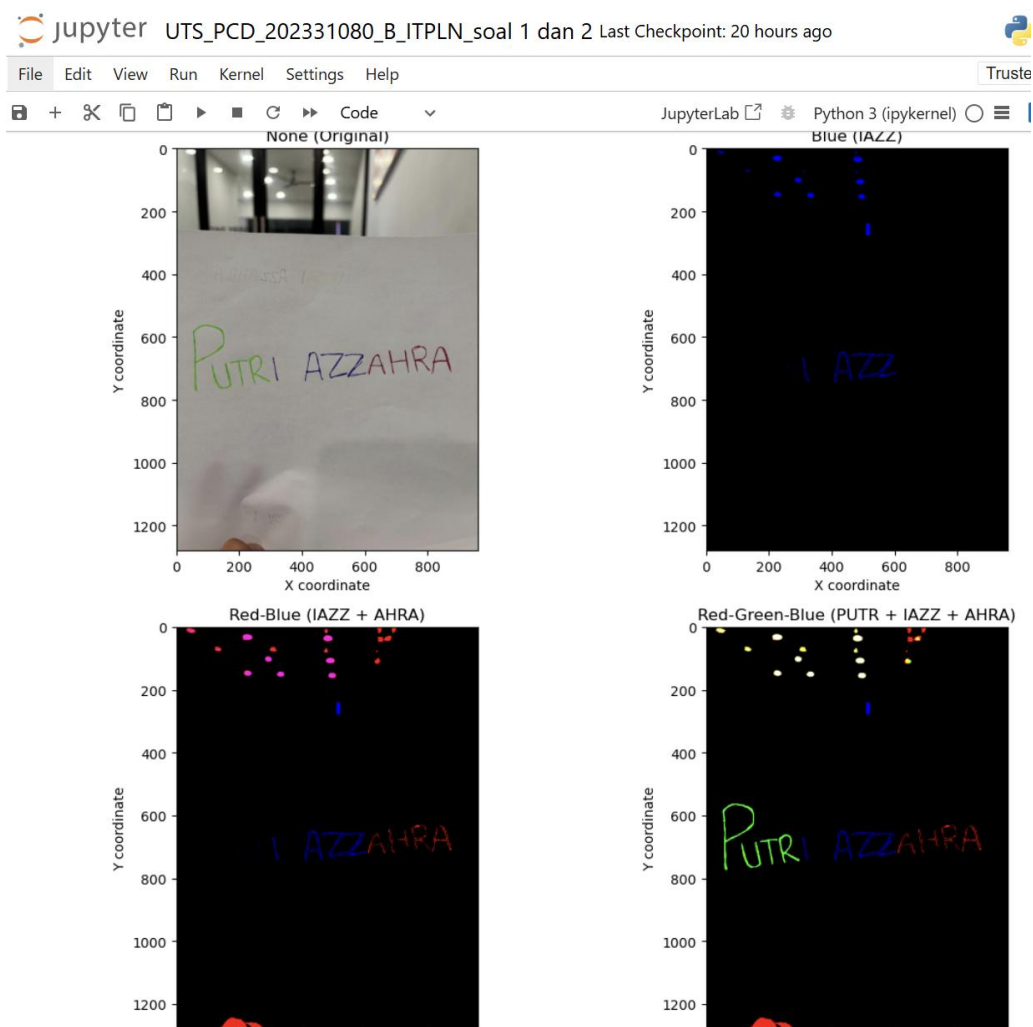
ax[1, 0].imshow(highlight_red_blue)
ax[1, 0].set_title("Red-Blue (IAZZ + AHRA)")

ax[1, 1].imshow(highlight_red_green_blue)
ax[1, 1].set_title("Red-Green-Blue (PUTR + IAZZ + AHRA)")

# Set koordinat
for a in ax.flatten():
    a.set_xlabel('X coordinate')
    a.set_ylabel('Y coordinate')
    a.set_xlim(0, img_rgb.shape[1])
    a.set_ylim(img_rgb.shape[0], 0)

plt.tight_layout()
plt.show()

```



Penjelasannya:

- **Membaca dan Mengonversi Gambar**
Gambar 'ara.jpg' dibaca menggunakan OpenCV dalam format BGR.
Gambar kemudian dikonversi ke format RGB untuk memudahkan analisis warna.
- **Inisialisasi Array Kosong untuk Highlighting:**
Tiga array kosong dibuat untuk menampung hasil penyorotan masing-masing warna: biru, hijau, dan merah.
- **Ekstraksi Saluran Warna:**
Saluran merah (r), hijau (g), dan biru (b) dipisahkan dari gambar RGB untuk analisis intensitas masing-masing warna.
- **Pembuatan Masker Berdasarkan Dominasi Warna:**
Mask Biru: Pixel dengan nilai biru > 100 dan lebih besar dari nilai merah dan hijau (dengan selisih minimal 5).
Mask Hijau: Pixel dengan nilai hijau > 100 dan lebih besar dari nilai merah dan biru (dengan selisih minimal 30).
Mask Merah: Pixel dengan nilai merah > 100 dan lebih besar dari nilai hijau dan biru (dengan selisih minimal 30).
- **Penerapan Masker untuk Menyorot Warna Dominan:**
Pixel yang memenuhi kondisi mask biru diwarnai biru murni $[0, 0, 255]$.
Pixel yang memenuhi kondisi mask hijau diwarnai hijau murni $[0, 255, 0]$.
Pixel yang memenuhi kondisi mask merah diwarnai merah murni $[255, 0, 0]$.
- **Pembuatan Gambar Gabungan:**
highlight_red_blue: Gabungan area yang disorot merah dan biru.
highlight_red_green_blue: Gabungan area yang disorot merah, hijau, dan biru.
- **Visualisasi Hasil:**
Gambar asli dan hasil penyorotan ditampilkan dalam subplot 2x2 menggunakan Matplotlib:
Gambar asli.
Area dominan biru.
Gabungan area dominan merah dan biru.
Gabungan area dominan merah, hijau, dan biru.
Setiap subplot diberi label koordinat X dan Y untuk referensi posisi piksel.

3.4 soal no 3 (Backlight)

jupyter UTS_PCD_202331080_B_ITPLN_Soal no 3 Last Checkpoint: 19 hours ago

File Edit View Run Kernel Settings Help Trusted

Python 3 (ipykernel)

Putri Azzahra (202331080)

3. Memperbaiki Gambar Backlight

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

[3]: def show_image(title, image, cmap='gray'):
    plt.figure(figsize=(6,6))
    plt.title(title)
    plt.imshow(image, cmap=cmap)
    plt.axis('off')
    plt.show()


[4]: img = cv2.imread('putri.jpg') # Gambar asli
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
show_image('Gambar Asli', img_rgb, cmap='viridis') # warna asli
```

jupyter UTS_PCD_202331080_B_ITPLN_Soal no 3 Last Checkpoint: 19 hours ago

File Edit View Run Kernel Settings Help Trusted

Python 3 (ipykernel)

Gambar Asli




```
[5]: gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
show_image('Gambar Gray', gray)
```

jupyter UTS_PCD_202331080_B_ITPLN_Soal no 3 Last Checkpoint: 19 hours ago

File Edit View Run Kernel Settings Help Trusted

JupyterLab Python 3 (ipykernel)

Gambar Gray



```
[6]: bright = cv2.convertScaleAbs(gray, alpha=1, beta=60) # beta = tingkat kecerahan
show_image('Gambar Gray yang Dicerahkan', bright)
```

Gambar Gray yang Dicerahkan

Collapse Output



```
: contrast = cv2.convertScaleAbs(gray, alpha=2.0, beta=0) # alpha = tingkat kontras
show_image('Gambar Gray yang Diperkontras', contrast)
```

jupyter UTS_PCD_202331080_B_ITPLN_Soal no 3 Last Checkpoint: 19 hours ago

File Edit View Run Kernel Settings Help Truste

Markdown

JupyterLab Python 3 (ipykernel)

Gambar Gray yang Diperkontras



```
[9]: combined = cv2.convertScaleAbs(gray, alpha=1.5, beta=50)
show_image('Gambar Gray yang Dicerahkan & Diperkontras', combined)
```


jupyter UTS_PCD_202331080_B_ITPLN_Soal no 3 Last Checkpoint: 19 hours ago

File Edit View Run Kernel Settings Help Truste

Markdown

JupyterLab Python 3 (ipykernel)

Gambar Gray yang Dicerahkan & Diperkontras



```
[10]: cv2.imwrite('hasil_gray.jpg', gray)
cv2.imwrite('hasil_bright.jpg', bright)
cv2.imwrite('hasil_contrast.jpg', contrast)
cv2.imwrite('hasil_combined.jpg', combined)

[10]: True
```

Penjelasannya:

- Pertama, gambar dibaca menggunakan OpenCV dalam format BGR dan dikonversi ke RGB agar tampilannya sesuai saat ditampilkan dengan Matplotlib. Setelah itu, gambar dikonversi ke grayscale untuk menghilangkan informasi warna dan hanya menyisakan intensitas cahaya. Proses ini penting untuk mempermudah analisis citra berdasarkan pencahayaan saja.
- Selanjutnya, dilakukan tiga transformasi pada gambar grayscale:
- Peningkatan kecerahan dilakukan dengan menambahkan nilai pada setiap piksel menggunakan parameter beta pada fungsi `cv2.convertScaleAbs()`, yang membuat keseluruhan gambar tampak lebih terang.
- Peningkatan kontras dilakukan dengan mengalikan nilai intensitas piksel menggunakan parameter alpha, sehingga perbedaan antara piksel gelap dan terang menjadi lebih mencolok, tanpa menambah terang secara keseluruhan.
- Gabungan kecerahan dan kontras dilakukan dengan mengatur alpha dan beta secara bersamaan, sehingga gambar menjadi lebih jelas dan terang sekaligus.
- Setiap hasil dari transformasi ini kemudian ditampilkan satu per satu dengan fungsi `show_image()` menggunakan Matplotlib, dan disimpan ke file dengan `cv2.imwrite()` untuk keperluan dokumentasi atau analisis lebih lanjut.

BAB IV

PENUTUP

Disimpulkan bahwa proses deteksi warna dasar (merah, hijau, dan biru) pada citra digital dapat dilakukan secara efektif menggunakan metode thresholding pada ruang warna RGB. Dengan menetapkan nilai ambang batas tertentu (misalnya $\text{threshold} = 50$), citra dapat disegmentasi berdasarkan kanal warna untuk mengekstrak masing-masing komponen warna secara terpisah. Teknik ini terbukti sederhana namun efisien untuk mendeteksi warna dominan pada gambar.

Selanjutnya, analisis histogram terhadap masing-masing kanal warna memberikan gambaran menyeluruh tentang distribusi intensitas piksel dalam citra. Histogram menunjukkan seberapa banyak piksel yang memiliki intensitas tertentu, serta membantu dalam memahami dominasi dan penyebaran warna. Kanal warna dengan distribusi luas menandakan adanya variasi warna yang tinggi, sedangkan kanal dengan distribusi sempit menunjukkan warna yang lebih homogen.

Melalui implementasi menggunakan pustaka Python seperti PIL untuk manipulasi gambar, NumPy untuk pemrosesan numerik, dan Matplotlib untuk visualisasi, proses pengolahan citra berjalan dengan baik. Proyek ini tidak hanya memberikan hasil visual yang informatif, tetapi juga meningkatkan pemahaman mahasiswa terhadap dasar-dasar pengolahan citra digital.

Secara keseluruhan, praktikum ini memberikan manfaat besar dalam mengasah kemampuan teknis serta memperkuat pemahaman konseptual dalam bidang segmentasi warna dan analisis visual citra. Hal ini menjadi fondasi penting untuk memahami teknik pengolahan citra yang lebih kompleks di masa mendatang.

DAFTAR PUSTAKA

- [1] Nugroho, A. S., & Setiawan, R. A. (2022). *Color-Based Object Detection for Tomato Maturity Classification Using Image Processing Techniques*. International Journal of Advanced Computer Science and Applications, 13(4), 71–77.
- [2] Khaoula, B., Abdelaziz, A., & El Habib, B. (2021). *Image Enhancement Based on Histogram Processing Techniques*. Procedia Computer Science, 184, 476–481.
- [3] Mohammed, R. A., & Marhoon, H. H. (2020). *Color Image Segmentation Using RGB Color Space Based on Threshold Values*. Materials Today: Proceedings, 42, 2143–2148.
- [4] Dey, N., & Ashour, A. S. (2019). *Contrast Enhancement Techniques for Low Contrast Images*. Journal of Intelligent & Fuzzy Systems, 36(1), 597–606.
- [5] Rachmawati, A., Sari, R. M., & Irawan, R. (2021). *Implementasi Segmentasi Warna Menggunakan Ruang Warna RGB dan HSV untuk Deteksi Objek*. Jurnal Teknologi dan Sistem Komputer, 9(3), 239–246.
- [6] Ardiansyah, F., & Widodo, A. (2022). *Peningkatan Kualitas Citra Menggunakan Teknik Penyesuaian Kontras dan Brightness Berbasis Python*. Jurnal INFOKOM, 11(1), 12–18.
- [7] Gunawan, A., & Prasetyo, H. (2023). *Visualisasi Histogram RGB untuk Analisis Warna Citra Digital pada Aplikasi Pemrosesan Gambar*. Jurnal Teknologi dan Informasi, 6(4), 25–33.