UTS

PENGOLAHAN CITRA

NAMA : Nazif Alfathir Siregar

NIM : 202331103

KELAS : C

DOSEN : Ir. Darma Rusjdi, M.Kom

NO.PC : 23

ASISTEN : 1. Abdur Rasyid Ridho

2. Rizqy Amanda

3. Kashrina Masyid Azka

4. Izzat Islami Kagapi

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2024/2025

DAFTAR ISI

Contents

BAB I
PENDAHULUAN

1.1 Rumusan Masalah

Pada praktik pengolahan citra digital, terdapat beberapa tantangan umum yang sering ditemui. Salah satunya adalah bagaimana cara mengenali warna-warna tertentu seperti merah, hijau, dan biru dalam sebuah gambar digital. Selain itu, diperlukan pemahaman tentang cara menampilkan distribusi intensitas warna menggunakan histogram, serta bagaimana menentukan nilai ambang batas (threshold) untuk menyeleksi warna yang diinginkan. Permasalahan lain muncul saat gambar yang diambil memiliki pencahayaan yang kurang baik, misalnya ketika objek membelakangi cahaya (backlight), sehingga bagian penting dari gambar tampak gelap.

1.2 Tujuan Masalah

- Memahami cara mendeteksi warna primer (merah, hijau, dan biru) pada gambar digital.
- Menampilkan dan menganalisis histogram dari masing-masing channel warna.
- Mempelajari teknik thresholding untuk menyaring bagian gambar berdasarkan warna tertentu.
- Mengaplikasikan metode perbaikan gambar backlight menggunakan peningkatan kecerahan dan kontras.

1.3 Manfaat Masalah

- Menambah wawasan mahasiswa mengenai dasar-dasar pengolahan citra digital.
- Meningkatkan kemampuan analisis visual menggunakan histogram warna.
- Memberikan pengalaman langsung dalam penerapan threshold untuk ekstraksi warna pada gambar.
- Memberikan pemahaman cara menangani gambar dengan pencahayaan tidak ideal agar objek utama tetap terlihat jelas.

.

BAB II

LANDASAN TEORI

1. Pengolahan Citra Digital

Pengolahan citra digital merupakan cabang dari ilmu komputer yang memanfaatkan algoritma untuk memanipulasi citra dengan tujuan meningkatkan kualitas visual atau mengekstrak informasi dari gambar tersebut. Proses ini melibatkan berbagai teknik matematis yang memungkinkan pengolahan data gambar menjadi format yang lebih berguna untuk analisis lebih lanjut. Dalam praktikum ini, penggunaan pustaka seperti OpenCV dan matplotlib sangat penting untuk melakukan konversi citra dari satu format warna ke format lainnya, misalnya dari BGR ke RGB, yang lebih umum digunakan dalam aplikasi visualisasi dan analisis data. Salah satu teknik dasar dalam pengolahan citra adalah pemisahan citra menjadi beberapa saluran warna yang terpisah, seperti merah, hijau, dan biru, yang memungkinkan kita untuk menganalisis komponen warna secara lebih mendalam. Oleh karena itu, pemahaman dasar tentang bagaimana citra dikodekan dalam format digital dan bagaimana proses konversi serta pemisahan warna dilakukan sangat penting dalam pengolahan citra.

2. Histogram dan Analisis Warna

Histogram dalam pengolahan citra merupakan representasi grafis dari distribusi intensitas piksel dalam citra tersebut. Setiap citra terdiri dari berbagai tingkat intensitas yang dapat dianalisis menggunakan histogram untuk mendapatkan gambaran yang lebih jelas mengenai dominasi warna dalam citra. Salah satu langkah penting dalam pengolahan citra adalah analisis saluran warna, di mana citra dibagi menjadi beberapa saluran terpisah (misalnya, merah, hijau, biru) untuk melihat distribusi intensitas masing-masing warna. Teknik ini penting karena memungkinkan kita untuk mengidentifikasi warna mana yang dominan dalam citra serta bagaimana distribusinya. Sebagai contoh, pada gambar yang mengandung teks atau objek tertentu, saluran merah bisa saja menunjukkan puncak distribusi yang lebih tinggi dibandingkan dengan saluran hijau atau biru, yang menandakan dominasi warna merah. Oleh karena itu, histogram tidak hanya berguna untuk menganalisis intensitas, tetapi juga membantu dalam memahami struktur warna dalam citra.

3. Peningkatan Kontras dengan CLAHE

Contrast Limited Adaptive Histogram Equalization (CLAHE) adalah teknik yang digunakan untuk meningkatkan kontras citra, terutama pada citra yang memiliki pencahayaan yang tidak merata atau kontras yang rendah. Teknik ini bekerja dengan cara membagi citra menjadi beberapa blok kecil dan menerapkan histogram equalization pada masing-masing blok tersebut, namun dengan batasan tertentu untuk menghindari noise yang berlebihan. Dalam pengolahan citra ini, CLAHE sering digunakan untuk mengatasi masalah seperti backlight, di mana objek utama dalam gambar tampak lebih gelap karena sumber cahaya berada di belakang objek. Dengan meningkatkan kontras citra menggunakan CLAHE, kita dapat menonjolkan detail yang sebelumnya tersembunyi dalam bayangan atau area dengan kecerahan rendah. Teknik ini sangat efektif dalam aplikasi yang membutuhkan pencahayaan yang lebih merata, seperti pada pengolahan citra medis, pengenalan wajah, atau analisis citra satelit.

4. Deteksi Warna dalam Citra menggunakan Masking

Deteksi warna adalah salah satu teknik yang sering digunakan dalam pengolahan citra untuk mengidentifikasi dan memisahkan objek berdasarkan warna tertentu. Teknik ini memanfaatkan ruang warna HSV (Hue, Saturation, Value) yang lebih sesuai untuk pemisahan warna dibandingkan dengan ruang warna RGB. Dalam implementasinya, masking dilakukan dengan mendefinisikan rentang nilai HSV yang sesuai untuk warna yang ingin dideteksi. Misalnya, untuk mendeteksi warna biru, kita menentukan rentang nilai HSV yang mencakup berbagai variasi warna biru dalam citra. Begitu mask diterapkan, hanya piksel yang berada dalam rentang nilai tersebut yang akan dipilih, sementara piksel lainnya akan diabaikan. Dengan menggunakan teknik ini, kita dapat mengisolasi warna tertentu dalam citra, seperti deteksi objek berdasarkan warna atau pelacakan warna dalam video. Penggunaan dua rentang untuk warna merah yang terletak pada dua ujung spektrum warna juga merupakan bagian penting dari deteksi warna yang akurat dalam citra.

5. Konversi Citra ke Grayscale dan Efeknya

Konversi citra dari format warna ke grayscale adalah langkah awal yang penting dalam banyak aplikasi pengolahan citra, karena mengurangi kompleksitas gambar dan memungkinkan analisis lebih lanjut yang lebih mudah. Pada dasarnya, citra grayscale hanya mengandung informasi tentang kecerahan piksel, sementara komponen warna (merah, hijau, biru) diabaikan. Teknik ini penting karena banyak algoritma pengolahan citra, seperti deteksi tepi atau pengenalan objek, bekerja lebih baik dengan citra grayscale. Proses konversi ini dilakukan dengan mengubah citra berwarna (RGB atau BGR) menjadi satu saluran yang merepresentasikan intensitas piksel. Salah satu manfaat utama konversi ke grayscale adalah penyederhanaan citra, yang memudahkan dalam meningkatkan kecerahan atau kontras citra, serta mengurangi beban komputasi dalam analisis citra.

.

BAB III

HASIL

1. Citra dan Histogram

1.1 Membaca dan Mengonversi Citra

```
#202331103_Nazif Alfathir Siregar
import cv2
import matplotlib.pyplot as plt

img = cv2.imread("citra1.jpg")
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

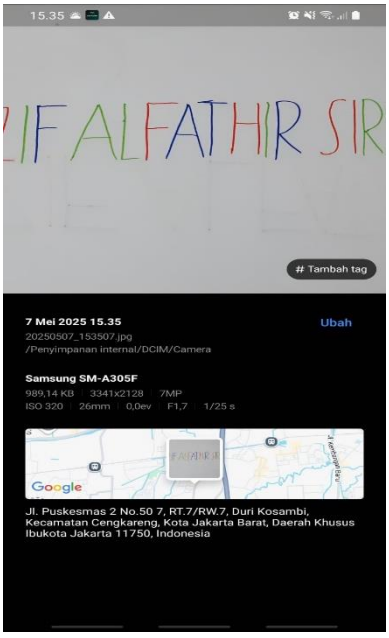
Langkah pertama yang dilakukan adalah memuat citra menggunakan pustaka OpenCV dengan fungsi cv2.imread(). Secara default, OpenCV membaca citra dalam format BGR (Blue, Green, Red), yang merupakan urutan warna standar pada OpenCV. Agar lebih mudah dianalisis dengan pustaka visualisasi lain seperti matplotlib, citra tersebut kemudian dikonversi ke format RGB (Red, Green, Blue). Proses konversi ini dilakukan dengan menggunakan fungsi cv2.cvtColor(), karena format RGB lebih umum digunakan dalam pengolahan gambar dan visualisasi.

1.2 Memisahkan Channel Warna

```
channel_merah = img_rgb[:, :, 0]
channel_hijau = img_rgb[:, :, 1]
channel_biru = img_rgb[:, :, 2]
```

Setelah citra berhasil dikonversi ke format RGB, langkah selanjutnya adalah memisahkan citra ke dalam tiga saluran warna terpisah: Merah, Hijau, dan Biru. Hal ini dilakukan dengan menggunakan teknik pengindeksan array pada citra yang telah diubah ke format RGB. Saluran Merah diambil dengan img_rgb[:, :, 0], saluran Hijau dengan img_rgb[:, :, 1], dan saluran Biru dengan img_rgb[:, :, 2].

1.3 Menampilkan Citra Asli



Dengan menggunakan fungsi plt.imshow(), citra dapat divisualisasikan dalam bentuk gambar yang jelas. Untuk meningkatkan tampilan, ukuran gambar diatur dengan plt.figure(figsize=(6, 4)) dan diberi judul "CITRA KONTRAS" menggunakan plt.title(). Sumbu gambar dihilangkan dengan plt.axis("off") agar hanya citra yang tampil tanpa gangguan. Terakhir, fungsi plt.show() digunakan untuk menampilkan gambar

tersebut, memberi gambaran umum tentang citra sebelum analisis saluran warna lebih lanjut dilakukan.

1.4 Mengatur data channel warna (channel_data)

```
channel_data = [
    ('MERAH', channel_merah, 'red'),
    ('HIJAU', channel_hijau, 'green'),
    ('BIRU', channel_biru, 'blue')
]
```

channel_data adalah sebuah daftar yang berisi tiga pasangan data, masing-masing mewakili satu channel warna: merah, hijau, dan biru. Setiap pasangan data terdiri dari tiga komponen: nama warna channel, data channel yang telah dipisahkan, dan warna yang akan digunakan untuk histogram channel tersebut. Misalnya, untuk channel merah, terdapat pasangan data ('MERAH', channel_merah, 'red'), yang berarti nama channel tersebut adalah 'MERAH', data channelnya berasal dari variabel channel_merah, dan histogram untuk channel merah akan diberi warna merah. Untuk channel hijau, terdapat pasangan data ('HIJAU', channel_hijau, 'green'), yang berarti nama channelnya adalah 'HIJAU', data channel diambil dari variabel channel_hijau, dan histogram akan diberi warna hijau. Begitu juga untuk channel biru, pasangan data ('BIRU', channel_biru, 'blue') menunjukkan bahwa nama channelnya adalah 'BIRU', data channel berasal dari variabel channel_biru, dan histogramnya akan diberi warna biru.

1.5 Looping untuk menampilkan gambar dan histogram

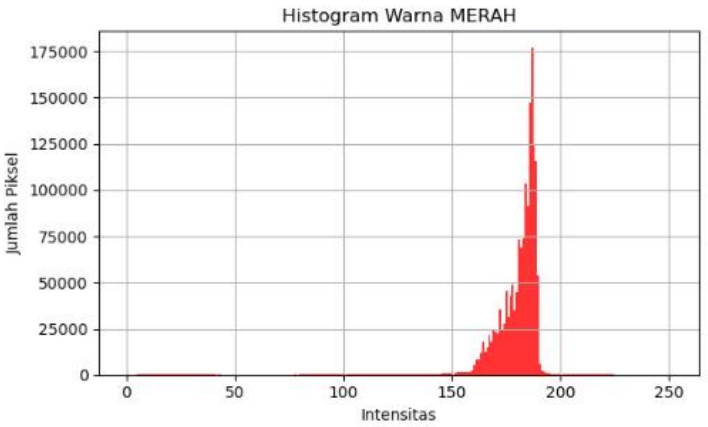
```
for title, channel, color in channel_data:
    fig, axs = plt.subplots(1, 2, figsize=(12, 4))

    axs[0].imshow(channel, cmap='gray')
    axs[0].set_title(title)
    axs[0].axis("off")

    axs[1].hist(channel.ravel(), bins=256, color=color, alpha=0.8)
    axs[1].set_title(f"Histogram Warna {title}")
    axs[1].set_xlabel("Intensitas")
    axs[1].set_ylabel("Jumlah Piksel")
    axs[1].grid(True)

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

Perulangan { for title, channel, color in channel_data: } digunakan untuk menampilkan gambar grayscale dan histogram untuk setiap channel warna. Pada setiap siklus perulangan, title akan berisi nama warna channel seperti 'MERAH', 'HIJAU', atau 'BIRU', yang digunakan untuk memberi judul pada gambar dan histogram. channel adalah data channel yang telah diekstraksi, seperti channel_merah, channel_hijau, atau channel_biru, yang digunakan untuk menampilkan gambar grayscale sesuai warna channel. Sementara itu, color digunakan untuk menentukan warna histogram yang ditampilkan, seperti 'red' untuk merah, 'green' untuk hijau, dan 'blue' untuk biru. Dengan cara ini, setiap perulangan memungkinkan tampilan visual yang terpisah untuk gambar grayscale dan histogram

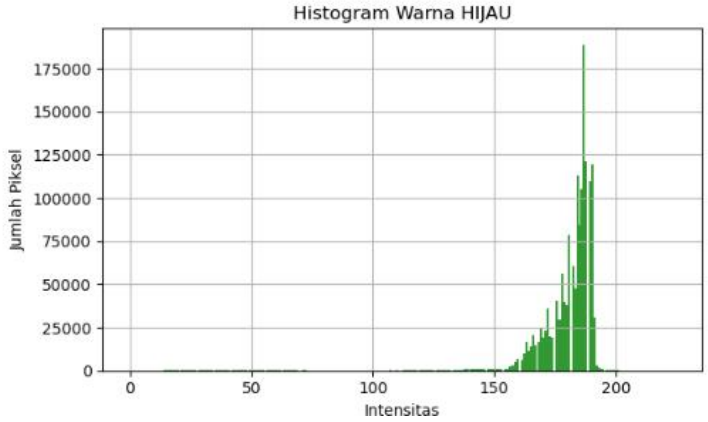
masing-masing channel warna, yang memberikan pemahaman yang lebih jelas mengenai distribusi intensitas warna dalam citra tersebut.

1.6 Gambar citra dan Histogram

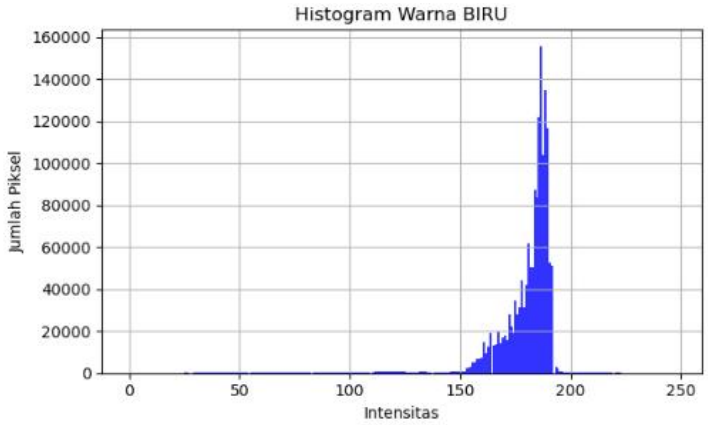
MERAH



HIJAU



BIRU



1.7 Analisis Citra dan Histogram

1. **Channel Merah (Red)**
   - **Citra**: Pada citra channel merah, gambar tampil dalam gradasi abu-abu yang hanya menggambarkan intensitas warna merah. Terlihat bahwa tulisan pada gambar tampak lebih gelap, yang menandakan bahwa warna merah lebih dominan di bagian tertentu gambar.

- **Histogram**: Histogram untuk channel merah menunjukkan puncak yang jelas di rentang intensitas 150-200. Hal ini menunjukkan bahwa mayoritas piksel dalam gambar memiliki intensitas merah yang cukup tinggi. Dengan demikian, warna merah memang sangat mendominasi gambar, meskipun ada sedikit area dengan intensitas rendah.

2. Channel Hijau (Green)

- **Citra**: Citra untuk channel hijau menunjukkan tampilan grayscale yang hanya mencerminkan intensitas warna hijau. Dibandingkan dengan channel merah, gambar ini terlihat lebih terang, yang menunjukkan bahwa distribusi intensitas warna hijau lebih merata di seluruh citra.

- **Histogram**: Histogram warna hijau memiliki distribusi yang mirip dengan channel merah, dengan puncak pada rentang intensitas 150-200. Ini mengindikasikan bahwa warna hijau cukup dominan, hampir setara dengan warna merah dalam citra, meskipun intensitas hijau sedikit lebih tersebar secara merata.

3. Channel Biru (Blue)

- **Citra**: Gambar untuk channel biru menunjukkan tampilan grayscale yang menggambarkan intensitas biru dalam citra. Gambar ini terlihat lebih terang dibandingkan dengan gambar channel merah dan hijau, yang menunjukkan bahwa intensitas warna biru tidak begitu kuat dalam citra ini.

- **Histogram**: Histogram channel biru menunjukkan puncak yang lebih rendah daripada merah dan hijau, meskipun masih ada distribusi di rentang intensitas 150-200. Ini menunjukkan bahwa biru tetap berkontribusi dalam gambar, tetapi dalam jumlah yang lebih sedikit dibandingkan dengan warna merah dan hijau.

2. Ambang Batas

2.1.Membaca dan Mengonversi Citra

```
#202331103_Nazif Alfathir Siregar
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread("citra1.jpg")
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

Program dimulai dengan memuat gambar yang bernama "citra1.jpg" menggunakan perintah cv2.imread(). Setelah itu, gambar yang awalnya dalam format BGR diubah ke format RGB dengan cv2.cvtColor(img, cv2.COLOR_BGR2RGB), agar lebih mudah untuk ditampilkan. Selain itu, gambar juga dikonversi ke format HSV dengan perintah cv2.cvtColor(img, cv2.COLOR_BGR2HSV). Penggunaan format HSV memudahkan dalam mendeteksi warna tertentu karena komponen warna bisa dipisahkan dengan lebih jelas.

2.2.Penggunaan Masking untuk Deteksi Warna

```
mask_none = np.zeros(img_rgb.shape[:2], dtype="uint8")
```

Selanjutnya, program membuat mask untuk mendeteksi warna spesifik dalam gambar. Mask untuk warna biru dibuat dengan menentukan rentang nilai HSV yang sesuai menggunakan dua parameter, lower_blue dan upper_blue, kemudian cv2.inRange() digunakan untuk menandai piksel yang berada dalam rentang tersebut. Untuk warna merah, karena terletak di dua area berbeda dalam ruang warna HSV, mask merah dibuat dengan dua rentang, yaitu lower_red1 dan upper_red1 untuk rentang pertama, serta lower_red2 dan upper_red2 untuk rentang kedua. Kedua rentang ini digabungkan menggunakan operator | (bitwise OR) untuk membentuk mask merah. Untuk warna hijau, mask dibuat dengan rentang nilai yang sudah ditentukan melalui lower_green dan upper_green yang juga diproses dengan fungsi cv2.inRange().

```
lower_blue = np.array([100, 100, 50])
upper_blue = np.array([130, 255, 255])
mask_blue = cv2.inRange(img_hsv, lower_blue, upper_blue)

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_red = cv2.inRange(img_hsv, lower_red1, upper_red1) | cv2.inRange(img_hsv, lower_red2, upper_red2)

lower_green = np.array([40, 100, 50])
upper_green = np.array([90, 255, 255])
mask_green = cv2.inRange(img_hsv, lower_green, upper_green)
```

2.3.Penggabungan

```
mask_rb = cv2.bitwise_or(mask_red, mask_blue)

mask_rgb = cv2.bitwise_or(mask_rb, mask_green)
```

Setelah masing-masing warna memiliki masknya sendiri, program kemudian menggabungkan mask tersebut. Mask biru dan merah digabungkan menggunakan cv2.bitwise_or(mask_red, mask_blue), sehingga menghasilkan mask yang menampilkan kedua warna tersebut. Kemudian, mask yang menggabungkan merah dan biru tersebut digabungkan lagi dengan mask hijau menggunakan cv2.bitwise_or() untuk menghasilkan mask final yang mencakup ketiga warna: merah, biru, dan hijau.

2.4.Menampilkan Mask

```
titles = ["NONE", "BLUE", "RED-BLUE", "RED-GREEN-BLUE"]
masks = [mask_none, mask_blue, mask_rb, mask_rgb]

plt.figure(figsize=(12, 8))
for i in range(4):
    plt.subplot(2, 2, i + 1)
    plt.imshow(masks[i], cmap='gray')
    plt.title(titles[i])
    plt.axis("off")
plt.tight_layout()
plt.show()
```

Setelah masker dibuat, hasil deteksi warna ditampilkan menggunakan matplotlib.pyplot. Empat gambar yang menunjukkan hasil deteksi warna ditampilkan dalam satu grid 2x2, masing-masing menggambarkan hasil deteksi untuk tidak ada warna (mask_none), biru, gabungan merah-biru, dan gabungan merah-biru-hijau. Setiap gambar diberi judul sesuai dengan jenis masker yang diterapkan, dan sumbu pada gambar dihilangkan agar fokus tetap pada visualisasi warna yang terdeteksi.

2.5.Analisis Ambang Batas

Nilai Ambang Batas:

a. **Warna Biru**:
   - Rentang HSV:
     - lower_blue = [100, 100, 50]
     - upper_blue = [130, 255, 255]

b. **Warna Merah**:
   - Rentang HSV:
     - lower_red1 = [0, 100, 50]
     - upper_red1 = [10, 255, 255]
     - lower_red2 = [160, 100, 50]
     - upper_red2 = [180, 255, 255]

c. **Warna Hijau**:
   - Rentang HSV:
     - lower_green = [40, 100, 50]
     - upper_green = [90, 255, 255]

Penentuan nilai ambang batas untuk mendeteksi warna dalam citra ini dilakukan dengan memilih rentang nilai dalam ruang warna HSV yang sesuai untuk warna yang ingin diproses, yaitu merah, hijau, dan biru. Proses ini dilakukan dengan melakukan eksperimen terhadap citra yang ada, menguji berbagai rentang nilai untuk setiap warna, dan menyesuaikan dengan karakteristik warna yang terlihat. Sebagai contoh, warna merah memiliki dua rentang ambang batas karena berada di kedua ujung spektrum warna dalam ruang HSV, yaitu antara 0° hingga 10° dan 160° hingga 180°. Dengan membagi rentang ini, program dapat mendeteksi berbagai variasi warna merah yang ada dalam citra dengan lebih tepat, sehingga menghindari kesalahan deteksi.

Selain itu, rentang ambang batas untuk warna biru dan hijau juga ditentukan berdasarkan intensitas dan kejenuhan warna dalam citra. Penentuan ambang batas yang tepat sangat penting agar deteksi warna dapat dilakukan dengan akurat tanpa tercampur dengan warna lain yang tidak relevan. Dengan menggunakan rentang ambang batas yang telah disesuaikan ini, deteksi warna dapat tetap konsisten meskipun ada perubahan pencahayaan atau latar belakang. Proses ini memastikan bahwa warna yang diinginkan terdeteksi dengan tepat, bahkan dalam kondisi citra yang berbeda.

2.6.Hasil

3. Memperbaiki BackLight

3.1 Menampilkan Gambar Asli

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread("citra2.jpg")
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

plt.figure(figsize=(5, 5))
plt.imshow(img_rgb)
plt.title("Gambar Asli (RGB)")
plt.axis('off')
plt.show()
```

Langkah pertama dalam pengolahan citra ini adalah membaca dan menampilkan gambar yang akan diproses. Gambar dimuat menggunakan fungsi cv2.imread("citra2.jpg") dari pustaka OpenCV. Gambar yang dibaca ini awalnya berada dalam format BGR, namun untuk menampilkan gambar dengan warna yang akurat, formatnya diubah menjadi RGB dengan menggunakan cv2.cvtColor(img, cv2.COLOR_BGR2RGB). Setelah itu, gambar ditampilkan dengan menggunakan matplotlib agar kita bisa melihat kondisi asli gambar yang akan diproses. Pada tahap ini, gambar menunjukkan masalah pencahayaan karena efek backlight, di mana wajah objek tampak lebih gelap dibandingkan dengan latar belakang yang sangat terang, akibat sumber cahaya yang berada di belakang objek.

3.2 Konversi ke Grayscale

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

plt.figure(figsize=(5, 5))
plt.imshow(gray, cmap='gray')
plt.title("Grayscale")
plt.axis('off')
plt.show()
```

Setelah gambar asli ditampilkan, langkah berikutnya adalah mengubah gambar tersebut menjadi format grayscale. Proses ini mengurangi kompleksitas gambar dan memudahkan dalam penyesuaian kecerahan dan kontras. Menggunakan fungsi cv2.cvtColor(img, cv2.COLOR_BGR2GRAY), gambar yang sebelumnya berwarna diubah menjadi hitam-putih, di mana hanya informasi kecerahan yang dipertahankan, sementara warna dihilangkan. Konversi ini penting karena gambar dalam format grayscale lebih mudah untuk diproses dalam langkah-langkah selanjutnya, seperti peningkatan kecerahan dan kontras. Gambar yang sudah diubah menjadi grayscale kemudian ditampilkan untuk melihat perubahan yang telah diterapkan.

3.3 Meningkatkan Kecerahan

```
cerah = cv2.convertScaleAbs(gray, alpha=1, beta=50)

plt.figure(figsize=(5, 5))
plt.imshow(cerah, cmap='gray')
plt.title("Dipercerah")
plt.axis('off')
plt.show()
```

Setelah gambar dikonversi menjadi grayscale, langkah selanjutnya adalah meningkatkan kecerahan gambar yang gelap akibat backlight. Untuk itu, digunakan fungsi cv2.convertScaleAbs, yang memungkinkan kita menambah nilai konstan pada seluruh pixel gambar, tanpa mengubah skala warna atau kontras. Dengan menambahkan nilai beta=50, gambar yang awalnya gelap menjadi lebih terang, terutama pada area yang sebelumnya tertutup bayangan. Proses ini membantu menonjolkan wajah atau objek utama yang sebelumnya sulit dilihat karena pencahayaan yang tidak merata. Hasil gambar yang lebih cerah kemudian ditampilkan untuk menunjukkan perbedaan dengan gambar grayscale yang asli.

3.4 Meningkatkan Kontras (menggunakan CLAHE)

```
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
kontras = clahe.apply(gray)

plt.figure(figsize=(5, 5))
plt.imshow(contrast, cmap='gray')
plt.title("Ditingkatkan Kontras")
plt.axis('off')
plt.show()
```

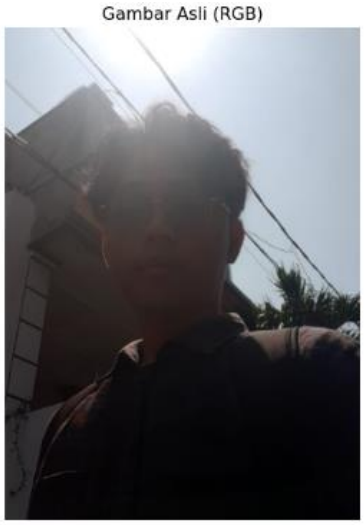
Setelah kecerahan ditingkatkan, langkah berikutnya adalah memperbaiki kontras gambar. Untuk itu, digunakan teknik CLAHE (Contrast Limited Adaptive Histogram Equalization), yang bertujuan meningkatkan kontras tanpa menyebabkan gambar menjadi terlalu terang atau hilang detail. Fungsi cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8)) digunakan untuk menerapkan metode CLAHE, di mana clipLimit mengatur batas peningkatan kontras dan tileGridSize mengatur ukuran blok yang diproses. Dengan teknik ini, gambar yang sebelumnya kurang kontras dapat diperbaiki, sehingga objek utama, seperti wajah, menjadi lebih jelas dan terpisah dari latar belakang yang terang. Gambar hasil peningkatan kontras kemudian ditampilkan untuk menunjukkan perubahan yang terjadi pada citra.

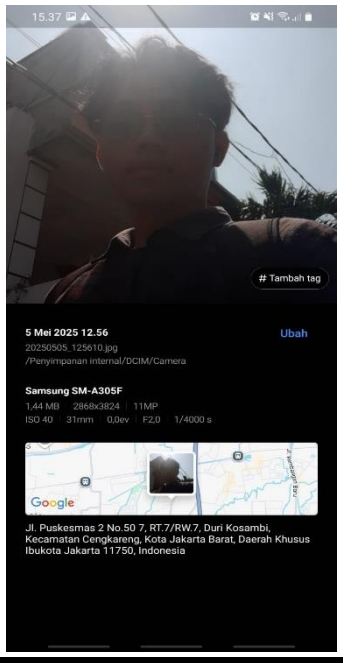
3.5 Meningkatkan Kecerahan dan Kontras

```
gabung_kontras = clahe.apply(bright)

plt.figure(figsize=(5, 5))
plt.imshow(gabung_kontras, cmap='gray')
plt.title("Dipercerah + Kontras")
plt.axis('off')
plt.show()
```

Pada langkah terakhir, gambar yang telah diperbaiki kecerahannya kemudian diproses lebih lanjut dengan CLAHE untuk meningkatkan kontras secara keseluruhan. Penggabungan kedua teknik ini menghasilkan gambar yang lebih terang dengan kontras yang lebih tajam, sehingga objek utama (wajah atau tubuh) lebih menonjol dan jelas terlihat. Gambar yang telah melalui proses perbaikan kecerahan dan kontras ini kemudian ditampilkan sebagai hasil akhir dari pengolahan citra. Gambar ini menunjukkan wajah atau objek utama yang lebih terang dan jelas, dengan perbedaan kontras yang lebih baik antara objek dan latar belakang.

3.6 Hasil Pengolahan Citra :





BAB IV

PENUTUP

Melalui mengerjakan UTS praktikum pengolahan citra digital ini, saya memperoleh pemahaman yang lebih baik tentang bagaimana warna dalam sebuah gambar dapat dianalisis dan diolah secara terpisah. Proses deteksi warna seperti merah, hijau, dan biru memberikan gambaran bahwa tiap channel warna menyimpan informasi tersendiri yang bisa dimanfaatkan, terutama saat divisualisasikan dalam bentuk grayscale dan histogram. Selain itu, penggunaan metode thresholding, khususnya dalam ruang warna HSV, terbukti lebih efektif dalam memisahkan warna spesifik dibandingkan metode RGB.

Pada bagian perbaikan citra dengan pencahayaan backlight, saya belajar bahwa peningkatan kecerahan saja tidak selalu cukup untuk menonjolkan detail objek yang gelap. Dengan bantuan teknik CLAHE, detail pada area gelap menjadi lebih terlihat tanpa harus mengorbankan bagian lain yang sudah terang. Dari pengalaman praktikum ini, saya merasa mendapatkan banyak pengetahuan praktis yang mendukung pemahaman teori, dan sekaligus menyadari pentingnya pengolahan citra dalam dunia nyata yang membutuhkan visualisasi data yang jelas dan informatif.

DAFTAR PUSTAKA

[1] Tiwari, P., & Singh, A. (2022). Comparative analysis of histogram equalization techniques for image enhancement. Materials Today: Proceedings, 56, 1725–1730.

[2] Gautam, A., & Shukla, S. (2021). Image segmentation using color thresholding in HSV color space. Materials Today: Proceedings, 47, 1182–1187.

[3] Mustafidah, U., & Pertiwi, R. (2021). Penerapan CLAHE untuk peningkatan kualitas citra digital pada gambar backlight menggunakan Python. Jurnal Teknik ITS, 10(2), A238–A242.

[4] Hassan, M., Akhtar, N., & Majeed, I. (2019). Color image enhancement using RGB and HSV color models. Journal of King Saud University – Computer and Information Sciences, 31(3), 328–336.