UTS

PENGOLAHAN CITRA

NAMA : Sunu Ananto Widodo

NIM : 202331155

KELAS : F

DOSEN : Rosida Nur Aziza, S.T., M.Eng. Sc

NO.PC : 07

ASISTEN :

1. Sasikirana Ramadhanty Setiawan Putri
2. Rizqy Amanda
3. Ridho Chaerullah
4. Sakura Amastasya Salsabila Setiyanto

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2024/2025

DAFTAR ISI

Contents

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

Bagaimana cara mengimplementasikan metode pengaturan backlight yang efektif dan responsif menggunakan OpenCV dan/atau teknik pemrosesan citra lainnya untuk meningkatkan kualitas visual gambar atau video dalam berbagai kondisi pencahayaan?

1.2 Tujuan Masalah

1. Mengidentifikasi dan mengimplementasikan algoritma pemrosesan citra berbasis OpenCV dan/atau metode lainnya yang sesuai untuk mengatur tingkat backlight.
2. Menganalisis efektivitas algoritma yang diimplementasikan dalam menyesuaikan kecerahan gambar atau video secara dinamis berdasarkan kondisi pencahayaan.
3. Mengevaluasi responsivitas dan kinerja algoritma dalam skenario perubahan pencahayaan yang berbeda.

(Opsional) Membandingkan kinerja beberapa metode pengaturan backlight yang berbeda.

1.3 Manfaat Masalah

1. Peningkatan Kualitas Visual: Menghasilkan gambar atau video dengan tingkat kecerahan yang optimal, sehingga meningkatkan detail dan kenyamanan visual bagi pengguna.

2. Adaptasi Terhadap Kondisi Pencahayaan: Sistem dapat secara otomatis menyesuaikan backlight dalam berbagai lingkungan pencahayaan (misalnya, dalam ruangan redup, di luar ruangan yang terang).

3. Potensi Aplikasi Luas: Dapat diterapkan dalam berbagai aplikasi seperti sistem pengawasan, pemrosesan video, antarmuka pengguna grafis, dan sistem pencitraan lainnya.

4. Pengembangan Teknik Pemrosesan Citra: Memberikan wawasan dan kontribusi pada pengembangan teknik pemrosesan citra untuk penyesuaian pencahayaan otomatis.

BAB II

LANDASAN TEORI

2.1.Dasar-Dasar Citra Digital Citra digital

Citra digital adalah representasi visual dari suatu adegan atau objek dalam bentuk digital, yang terdiri dari piksel-piksel yang tersusun dalam baris dan kolom.[1] Piksel, atau elemen gambar, adalah unit dasar dari citra digital. Setiap piksel memiliki nilai numerik yang menggambarkan intensitas atau warna pada posisi tertentu dalam citra.[1]

Proses akuisisi citra melibatkan pengambilan citra analog dan konversinya menjadi format digital melalui proses sampling (pencuplikan) dan kuantisasi.[2] Sampling adalah proses mendiskretkan citra kontinu menjadi sejumlah titik sampel (piksel), sedangkan kuantisasi adalah proses mendiskretkan nilai intensitas piksel kontinu menjadi sejumlah nilai diskrit. Kualitas citra digital dipengaruhi oleh jumlah piksel (resolusi) dan rentang intensitas warna yang tersedia.[2] Konsep fundamental citra digital sebagai matriks diskrit dari nilai piksel secara konsisten ditekankan di berbagai sumber, menyoroti sifat intinya untuk pemrosesan komputasional. Proses digitalisasi (sampling dan kuantisasi) merupakan langkah penting yang menjembatani kesenjangan antara citra analog dunia nyata dan representasi digitalnya yang dapat diproses oleh komputer.[2]

2.2. Citra Skala Keabuan (Grayscale Images)

Citra skala keabuan hanya mengandung informasi intensitas cahaya.4 Setiap piksel dalam citra skala keabuan direpresentasikan oleh satu nilai numerik, biasanya dalam rentang 0 hingga 255 untuk citra 8-bit.6 Nilai 0 mewakili warna hitam, sedangkan nilai 255 mewakili warna putih, dengan nilai di antaranya merepresentasikan berbagai tingkat keabuan. Konversi dari citra berwarna menjadi skala keabuan sering kali menyederhanakan proses pengolahan untuk tugas-tugas tertentu.16 Representasi skala keabuan berfungsi sebagai penyederhanaan fundamental data citra, sering digunakan sebagai langkah awal dalam berbagai tugas pengolahan citra sebelum analisis atau manipulasi warna yang lebih kompleks. Untuk mengubah citra berwarna (RGB) menjadi citra skala keabuan dengan nilai intensitas X,

rumusnya adalah:

$$X=3R+G+B$$

dengan representasi warna skala keabuan adalah RGB(X, X, X).14

2.3. Model Warna RGB (Red, Green, Blue)

Model warna RGB adalah representasi paling umum untuk citra berwarna.[3] Dalam model ini, setiap piksel direpresentasikan oleh tiga nilai yang sesuai dengan intensitas komponen warna merah, hijau, dan biru.6 Biasanya, setiap komponen warna memiliki nilai dalam rentang 0 hingga 255. Kombinasi ketiga warna primer ini menghasilkan berbagai macam warna.[4] Model RGB banyak digunakan dalam tampilan layar, kamera digital, dan pengolahan citra.[5] Korelasi langsung model RGB dengan persepsi visual manusia dan kemudahan implementasi perangkat kerasnya menjadikannya standar dominan untuk representasi citra berwarna dalam sistem digital.

2.4. Model Warna HSV (Hue, Saturation, Value)

Model warna HSV merepresentasikan warna berdasarkan Hue (corak warna), Saturation (kejenuhan warna), dan Value (kecerahan).20 Hue mendefinisikan warna itu sendiri (misalnya, merah, ungu, kuning), saturasi mengukur kemurnian atau intensitas warna, dan value menunjukkan tingkat kecerahan.20 Model HSV berguna untuk segmentasi berbasis warna dan mengidentifikasi kontras warna.20 Model HSV menawarkan cara yang lebih intuitif untuk menggambarkan dan memanipulasi warna berdasarkan persepsi manusia, sehingga menguntungkan untuk tugas-tugas seperti pemilihan warna dan segmentasi di mana karakteristik warna sangat penting. Konversi dari RGB ke HSV dapat dilakukan dengan langkah-langkah berikut 28: 1. Normalisasi nilai R, G, B ke dalam rentang : r=R+G+BR ,g=R+G+BG ,b=R+G+BB 2. Hitung Value (V): V=max(r,g,b) 3. Hitung Saturation (S): S={VV−min(r,g,b) 0 jika V=0jika V=0 4. Hitung Hue (H): ○ Jika r=V, maka H=60×(0+V−min(r,g,b)g−b) ○ Jika g=V, maka H=60×(2+V−min(r,g,b)b−r) ○ Jika b=V, maka H=60×(4+V−min(r,g,b)r−g) ○ Jika H

2.5.Operasi Berbasis Piksel (Pixel-based Operations)

Operasi berbasis piksel memodifikasi intensitas piksel berdasarkan nilai individualnya. Operasi titik mengubah nilai intensitas piksel secara langsung berdasarkan nilai piksel itu sendiri.6 Contohnya termasuk penyesuaian kecerahan dan kontras. Contrast stretching memperluas rentang intensitas dalam citra untuk meningkatkan visibilitas. Histogram equalization mendistribusikan kembali intensitas piksel untuk meningkatkan kontras secara keseluruhan.6 Operasi lokal melibatkan pemrosesan piksel berdasarkan nilai piksel tetangganya.6 Spatial filtering menggunakan mask atau kernel untuk memodifikasi nilai piksel untuk penghalusan (smoothing), penajaman (sharpening), atau deteksi tepi. Filter mean menghaluskan citra dengan mengambil rata-rata nilai piksel tetangga. Filter median efektif untuk mengurangi noise dengan mengganti nilai piksel dengan nilai median dari piksel tetangganya. Filter Gaussian menggunakan rata-rata berbobot berdasarkan distribusi Gaussian untuk penghalusan. Filter penajaman meningkatkan tepi dan detail dalam citra. Operasi berbasis piksel membentuk dasar manipulasi citra, memungkinkan penyesuaian fundamental pada tampilan citra dan karakteristik noise. Perbedaan antara operasi titik dan lokal menyoroti skala di mana modifikasi ini diterapkan.

Secara umum, operasi filtering spasial dapat dinyatakan sebagai konvolusi antara citra f(x,y) dan kernel (filter) w(i,j):

$g(x,y)=i=-a\sum a\ j=-b\sum b\ w(i,j)\times f(x+i,y+j)$

dengan ukuran kernel (2a+1)×(2b+1).

Beberapa contoh filter spasial meliputi:

- Filter Rerata (Mean Filter): Menghitung nilai rata-rata piksel tetangga dalam kernel: $g(x,y)=(2a+1)(2b+1)1\ i=-a\sum a\ j=-b\sum b\ f(x+i,y+j)$
- Filter Median: Mengganti nilai piksel dengan nilai median dari piksel tetangganya dalam kernel.
- Filter Gaussian: Menggunakan kernel dengan bobot yang mengikuti distribusi Gaussian: $w(i,j)=2\pi\sigma 21\ e-2\sigma 2i2+j2$ di mana σ adalah standar deviasi dari distribusi Gaussian.53

BAB III

HASIL

Soal 1

```
B, G, R = cv2.split(image)

image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

faktor = 0.1
G_pudar = (G * faktor).astype(np.uint8)

fig, axs = plt.subplots(2, 2, figsize=(15, 8))

axs[0, 0].imshow(image_rgb)
axs[0, 0].set_title("CITRA KONTRAS")
axs[0, 0].axis('on')       # Menampilkan axis ticks
axs[0, 0].grid(False)      # Nonaktifkan grid

axs[0, 1].imshow(B, cmap='gray')
axs[0, 1].set_title("BIRU")
axs[0, 1].axis('on')
axs[0, 1].grid(False)

axs[1, 0].imshow(R, cmap='gray')
axs[1, 0].set_title("MERAH")
axs[1, 0].axis('on')
axs[1, 0].grid(False)

axs[1, 1].imshow(G, cmap='gray')
axs[1, 1].set_title("HIJAU")
axs[1, 1].axis('on')
axs[1, 1].grid(False)

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

1. Import Libraries:

- import cv2: Mengimpor library OpenCV untuk operasi pengolahan gambar.
- import matplotlib.pyplot as plt: Mengimpor library Matplotlib untuk membuat plot gambar.
- import numpy as np: Mengimpor NumPy untuk operasi numerik, terutama untuk manipulasi array gambar.

2. Split Image Channels:

- B, G, R = cv2.split(image): Fungsi cv2.split() memisahkan image menjadi tiga channel warna: Blue, Green, dan Red. Setiap channel disimpan dalam variabel B, G, dan R sebagai array terpisah. Jika image tidak didefinisikan sebelumnya, kode ini akan error.

3. Convert Color Space:

- image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB): OpenCV secara default menggunakan format warna BGR (Blue, Green, Red). Matplotlib menggunakan format RGB (Red, Green, Blue). Kode ini mengonversi format warna gambar dari BGR ke RGB sehingga dapat ditampilkan dengan benar

oleh Matplotlib.

4. Modify Green Channel:

- faktor = 0.1: Menentukan faktor skala sebesar 0.1.
- G_pudar = (G * faktor).astype(np.uint8): Kode ini mengalikan nilai intensitas channel Green (G) dengan 0.1. Ini secara efektif mengurangi intensitas warna hijau dalam gambar. Hasilnya disimpan dalam variabel G_pudar.

. astype(np.uint8) mengubah tipe data array menjadi unsigned 8-bit integer, yang merupakan tipe data yang umum untuk merepresentasikan intensitas pixel.

5. Create Figure and Axes:

- fig, axs = plt.subplots(2, 2, figsize=(15, 8)): Membuat sebuah figure dan sebuah grid dari axes (subplot).
  - 2, 2: Menentukan grid dengan 2 baris dan 2 kolom, sehingga total ada 4 subplot.
  - figsize=(15, 8): Menentukan ukuran figure menjadi 15 inci (lebar) x 8 inci (tinggi).

6. Display Images in Subplots:

Kode berikut menggunakan loop untuk menampilkan gambar-gambar di dalam subplot:

- axs[0, 0].imshow(image_rgb): Menampilkan gambar berwarna yang telah dikonversi ke RGB di subplot pertama (baris 1, kolom 1).
- axs[0, 0].set_title("CITRA KONTRAS"): Memberikan judul "CITRA KONTRAS" pada subplot pertama.
- axs[0, 0].axis('on'): Menampilkan sumbu koordinat (ticks) pada subplot pertama.
- axs[0, 0].grid(False): Menonaktifkan grid lines.
- axs[0, 1].imshow(B, cmap='gray'): Menampilkan channel Blue sebagai gambar grayscale di subplot kedua (baris 1, kolom 2). cmap='gray' mengatur colormap menjadi grayscale.
- axs[0, 1].set_title("BIRU"): Memberikan judul "BIRU" pada subplot kedua.

○ axs[0, 1].axis('on'): Menampilkan sumbu koordinat.

○ axs[0, 1].grid(False): Menonaktifkan grid lines.

○ axs[1, 0].imshow(R, cmap='gray'): Menampilkan channel Red sebagai gambar grayscale di subplot ketiga (baris 2, kolom 1).

○ axs[1, 0].set_title("MERAH"): Memberikan judul "MERAH" pada subplot ketiga.

○ axs[1, 0].axis('on'): Menampilkan sumbu koordinat.

○ axs[1, 0].grid(False): Menonaktifkan grid lines.

○ axs[1, 1].imshow(G, cmap='gray'): Menampilkan channel Green sebagai gambar grayscale di subplot keempat (baris 2, kolom 2).

○ axs[1, 1].set_title("HIJAU"): Memberikan judul "HIJAU" pada subplot keempat.

○ axs[1, 1].axis('on'): Menampilkan sumbu koordinat.

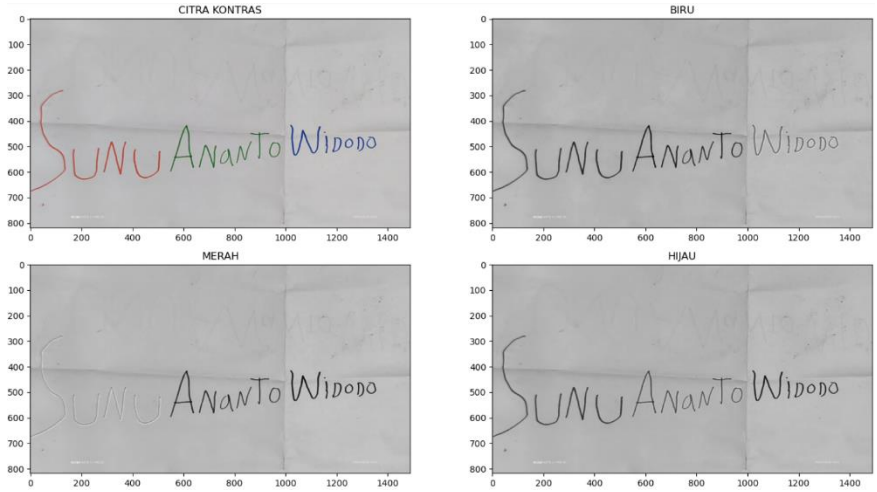
○ axs[1, 1].grid(False): Menonaktifkan grid lines.

7. Adjust Layout and Show Plot:

○ plt.tight_layout(): Menyesuaikan layout subplot agar tidak tumpang tindih.

○ plt.show(): Menampilkan plot yang berisi semua subplot gambar.

8. output



9.Ambang batas yang ditemukan

Ambang batas yang say temukan adalah (sekitar 150 ke atas) berdasarkan histogram ini adalah:

Puncak Distribusi: Histogram gabungan RGB dan masing-masing histogram warna (merah, hijau, biru) menunjukkan puncak distribusi piksel berada pada nilai-nilai intensitas yang

relatif tinggi. Puncak ini menandakan bahwa sebagian besar piksel dalam gambar memiliki intensitas di sekitar nilai tersebut atau lebih rendah, namun jumlah piksel dengan intensitas tinggi juga sangat signifikan.

Dominasi Piksel Terang: Jumlah frekuensi (ketinggian batang) pada sisi kanan histogram (intensitas tinggi) secara umum lebih besar dibandingkan sisi kiri (intensitas rendah). Ini secara visual menegaskan bahwa ada lebih banyak piksel dengan nilai intensitas yang lebih terang dalam gambar.

Potensi Pemisahan Objek Terang: Jika kita ingin memisahkan objek yang tampak lebih terang dari latar belakang yang lebih gelap, maka ambang batas di area intensitas tinggi akan secara efektif menangkap piksel-piksel terang tersebut.

Secara sederhana, histogram menunjukkan bahwa mayoritas piksel dalam gambar ini cenderung memiliki nilai intensitas yang lebih tinggi. Oleh karena itu, untuk melakukan segmentasi berdasarkan intensitas yang memisahkan area "terang" dari area "gelap", masuk akal untuk memilih ambang batas di dalam atau di sekitar rentang intensitas di mana piksel-piksel terang ini mendominasi.

Soal 2

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

```
path = "jankrik.jpg"
img = cv2.imread(path)

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

bright_gray = cv2.convertScaleAbs(img_gray, alpha=1, beta=50)

contrast_gray = cv2.convertScaleAbs(img_gray, alpha=1.5, beta=0)

bright_contrast_gray = cv2.convertScaleAbs(img_gray, alpha=1.5, beta=50)

plt.figure(figsize=(15, 10))

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

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

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

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

plt.subplot(2, 3, 5)
plt.imshow(bright_contrast_gray, cmap='gray')
plt.title("Gambar Gray Dipercerah & Diperkontras")
plt.axis("off")

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

1. Import Libraries

- cv2: Library OpenCV untuk operasi pengolahan gambar.
- numpy: Library untuk operasi array numerik.
- matplotlib.pyplot: Library untuk menampilkan gambar.

2. Fungsi display_image(image, title="Image")

- Fungsi ini mengambil sebuah gambar dan judul sebagai input.
- cv2.cvtColor(image, cv2.COLOR_BGR2RGB): OpenCV menggunakan format BGR (Blue, Green, Red) secara default. Fungsi ini mengubah format warna gambar dari BGR ke RGB, yang merupakan format yang digunakan oleh matplotlib.
- plt.imshow(...): Menampilkan gambar menggunakan matplotlib.
- plt.title(title): Menambahkan judul ke gambar yang ditampilkan.
- plt.axis('on'): Menampilkan sumbu koordinat pada gambar. Jika diganti 'off' maka sumbu tidak akan ditampilkan.
- plt.show(): Menampilkan plot gambar.

3. Load Gambar dan Konversi ke HSV

- hsv_image = cv2.imread('halo.jpg', cv2.IMREAD_COLOR): Membaca gambar 'halo.jpg' dalam format warna. cv2.IMREAD_COLOR memastikan gambar dibaca dalam format berwarna. Jika file halo.jpg tidak ada di folder yang sama dengan script Python Anda, maka akan muncul error.
- hsv_image = cv2.cvtColor(hsv_image, cv2.COLOR_BGR2HSV): Mengonversi gambar dari format BGR ke HSV (Hue, Saturation, Value). HSV lebih mudah digunakan untuk deteksi warna karena memisahkan informasi warna (Hue) dari kecerahan (Value) dan intensitas (Saturation).

4. Definisikan Rentang Warna

- Bagian ini mendefinisikan rentang nilai HSV untuk warna biru, merah, dan hijau yang akan dideteksi. Setiap warna memiliki batas bawah (lower_warna) dan batas atas (upper_warna).
- Nilai-nilai ini ditentukan dalam format HSV:
  - Hue (Warna): 0-180 (OpenCV menggunakan rentang 0-180 untuk Hue)

- Saturation (Kejenuhan): 0-255
- Value (Kecerahan): 0-255

○ Perhatikan bahwa warna merah memiliki dua rentang karena melintasi batas 0 derajat pada lingkaran warna Hue.

5. Membuat Masker Warna

○ cv2.inRange(hsv_image, lower_blue, upper_blue): Fungsi ini membuat masker (mask) biner. Untuk setiap piksel dalam hsv_image, jika nilai HSV-nya berada dalam rentang lower_blue dan upper_blue, piksel yang sesuai dalam masker akan bernilai 255 (putih), jika tidak, akan bernilai 0 (hitam).

○ Proses ini diulangi untuk warna merah (dengan dua rentang) dan hijau.

○ red_mask1 dan red_mask2 digunakan untuk menangkap rentang warna merah yang melintasi batas hue 0.

6. Ekstraksi Warna (Percobaan Pertama)

○ color_image = cv2.imread('halo.jpg'): Membaca ulang gambar.

○ hsv = cv2.cvtColor(color_image, cv2.COLOR_BGR2HSV): Konversi ke HSV.

○ mask = cv2.inRange(hsv, lower_blue, upper_blue): Membuat mask untuk warna biru.

○ hasil = cv2.bitwise_and(color_image, color_image, mask=mask): Ini adalah operasi bitwise AND. Operasi ini mempertahankan piksel dari color_image hanya di mana piksel yang sesuai dalam mask adalah 255 (putih). Dengan kata lain, ini mengekstrak piksel biru dari gambar asli.

○ plt.show(): Menampilkan hasil ekstraksi warna biru.

7. Penggabungan Masker Merah (Percobaan Kedua)

○ Bagian kode ini bertujuan untuk menggabungkan dua mask merah (mask_red1 dan mask_red2) untuk mendapatkan deteksi warna merah yang lengkap.

○ mask_red = cv2.bitwise_or(mask_red1, mask_red2): Operasi bitwise OR menggabungkan dua mask. Jika piksel dalam mask_red1 atau mask_red2

adalah 255, piksel yang sesuai dalam mask_red akan menjadi 255. Ini menggabungkan dua rentang merah.

- mask_combined = cv2.bitwise_or(mask_blue, mask_red): Menggabungkan mask biru dan mask merah yang sudah digabung.
- plt.show(): Menampilkan mask yang dikombinasikan.

8. Ekstraksi Warna dengan Semua Masker (Percobaan Ketiga)

- Bagian kode ini mengekstrak piksel dari gambar asli yang sesuai dengan warna biru, merah, atau hijau.
- Masker untuk biru, dua rentang merah, dan hijau dibuat.
- mask_combined = cv2.bitwise_or(...): Masker-masker ini digabungkan menggunakan operasi bitwise OR untuk membuat satu masker yang mencakup semua warna yang diinginkan.
- hasil = cv2.bitwise_and(color_image, color_image, mask=mask_combined): Operasi bitwise AND digunakan untuk mengekstrak piksel berwarna dari gambar asli berdasarkan masker gabungan.
- plt.show(): Menampilkan hasil akhir, yaitu gambar yang hanya menampilkan piksel biru, merah, dan hijau.

Soal 3

1. Mengimpor Library:

- import cv2: Mengimpor library OpenCV untuk pengolahan gambar.
- import matplotlib.pyplot as plt: Mengimpor Matplotlib untuk menampilkan gambar.

2. Membaca Gambar:

- path = "jankrik.jpg": Menyimpan path atau lokasi gambar dalam variabel path. Diasumsikan gambar bernama "jankrik.jpg" ada di direktori yang sama dengan script Python Anda.
- img = cv2.imread(path): Membaca gambar dari lokasi yang ditentukan dan menyimpannya dalam variabel img. Secara default, OpenCV membaca gambar dalam format BGR (Blue, Green, Red).

3. Konversi Warna:

- img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB): Mengonversi gambar dari format BGR (yang digunakan OpenCV) ke format RGB (yang digunakan Matplotlib).
- img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY): Mengonversi gambar berwarna menjadi gambar grayscale (keabuan).

4. Memodifikasi Gambar Grayscale:

- bright_gray = cv2.convertScaleAbs(img_gray, alpha=1, beta=50): Menyesuaikan kecerahan gambar grayscale. Parameter alpha adalah faktor skala (1 berarti tidak ada perubahan skala), dan beta adalah nilai yang ditambahkan ke setiap piksel (50 menambahkan kecerahan).
- contrast_gray = cv2.convertScaleAbs(img_gray, alpha=1.5, beta=0): Menyesuaikan kontras gambar grayscale. Parameter alpha lebih besar dari 1 (1.5) untuk meningkatkan kontras, dan beta adalah 0 (tidak ada perubahan kecerahan).
- bright_contrast_gray = cv2.convertScaleAbs(img_gray, alpha=1.5, beta=50): Menggabungkan penyesuaian kecerahan dan kontras. Kontras ditingkatkan (alpha=1.5) dan kecerahan juga ditingkatkan (beta=50).

5. Menampilkan Gambar:

- plt.figure(figsize=(15, 10)): Membuat sebuah figure (area gambar) dengan ukuran 15x10 inci.
- Serangkaian plt.subplot(): Membuat subplot-subplot di dalam figure untuk menampilkan beberapa gambar dalam satu tampilan.
    - Argumen pertama dan kedua (2, 3) menentukan tata letak grid subplot (2 baris, 3 kolom).
    - Argumen ketiga menentukan posisi subplot saat ini (dimulai dari 1).
- plt.imshow(): Menampilkan gambar pada subplot yang dipilih.
    - Untuk gambar grayscale, digunakan cmap='gray' agar ditampilkan dalam skala keabuan.
- plt.title(): Memberi judul pada setiap subplot.

- plt.axis("off"): Mematikan tampilan sumbu x dan y pada setiap subplot agar gambar terlihat lebih bersih.
- plt.tight_layout(): Menyesuaikan tata letak subplot agar tidak ada tumpang tindih.
- plt.show(): Menampilkan figure yang berisi semua subplot gambar.

6.Output

BAB IV

PENUTUP

Dalam konteks pengaturan backlight menggunakan OpenCV dan teknik pemrosesan citra, pemahaman mengenai histogram citra memegang peranan penting. Histogram citra merepresentasikan distribusi intensitas piksel dalam sebuah gambar. Dengan menganalisis histogram, kita dapat memperoleh informasi mengenai tingkat kecerahan keseluruhan, kontras, dan distribusi pencahayaan dalam citra. Misalnya, histogram yang cenderung condong ke kiri mengindikasikan gambar yang lebih gelap, sementara histogram yang condong ke kanan menunjukkan gambar yang lebih terang.

Konsep ambang batas (thresholding) juga relevan dalam beberapa pendekatan pengaturan backlight. Meskipun thresholding secara langsung lebih sering digunakan untuk segmentasi citra (memisahkan objek dari latar belakang), informasi yang diperoleh dari aplikasi thresholding dapat memberikan indikasi mengenai kondisi pencahayaan. Misalnya, jika sebagian besar piksel berada di bawah ambang batas tertentu, ini bisa mengindikasikan kebutuhan untuk meningkatkan backlight.

Algoritma pengaturan backlight yang efektif seringkali memanfaatkan informasi dari histogram untuk menentukan penyesuaian kecerahan yang sesuai. Beberapa metode mungkin secara langsung memanipulasi histogram (misalnya, histogram equalization) untuk meningkatkan kontras dan kecerahan. Metode lain mungkin menggunakan metrik statistik dari histogram (seperti nilai rata-rata intensitas) sebagai dasar untuk menyesuaikan tingkat backlight.

Dengan demikian, analisis histogram dan penerapan ambang batas (secara tidak langsung sebagai indikator kondisi pencahayaan) dapat menjadi komponen penting dalam merancang sistem pengaturan backlight yang adaptif dan responsif menggunakan OpenCV dan teknik pemrosesan citra lainnya.

DAFTAR PUSTAKA

1. PENGOLAHAN CITRA DIGITAL UNTUK IDENTIFIKASI OBJEK MENGGUNAKAN METODE HIERARCHICAL AGGLOMERATIVE CLUSTERING - Ejournal Undiksha, diakses Mei 7, 2025, https://ejournal.undiksha.ac.id/index.php/JST/article/download/33636/19968/99701

2. Computer Vision dan Pengolahan Citra Digital - Universitas Negeri Medan, diakses Mei 7, 2025, http://digilib.unimed.ac.id/id/eprint/53012/1/Book.pdf

3. Buku Ajar Pengolahan Citra Digital - Umsida Press, diakses Mei 7, 2025, https://press.umsida.ac.id/index.php/umsidapress/article/download/978-623-464075-5/1125

4. A Review of Digital Image Processing Techniques and Future Prospects - ResearchGate, diakses Mei 7, 2025, https://www.researchgate.net/publication/389180343_A_Review_of_Digital_Image

5. Aulia Akhrian Syahidi - Poliban Press, diakses Mei 7, 2025, https://press.poliban.ac.id/uploads/file/Akhrian_Aulia_Pengolahan_Citra_Digital.pd