

**UTS**

**PENGOLAHAN CITRA**



NAMA : Nyimas Adinda Paradiza

NIM : 202331041

KELAS : PCD-B

DOSEN : Ir. Darma Rusjdi, M.Kom

NO.PC : 12

ASISTEN : 1. Davina Najwa Ermawan

2. Fakhrol Fauzi Nugraha Tarigan

3. Viana Salsabila Fairuz Syajla

4. Muhammad Hanief Febriansyah

**INSTITUT TEKNOLOGI PLN**

**TEKNIK INFORMATIKA**

**2024/2025**

**DAFTAR ISI****Table of Contents**

DAFTAR ISI.....	2
BAB I PENDAHULUAN.....	3
1.1    Rumusan Masalah.....	3
1.2    Tujuan Masalah.....	3
1.3    Manfaat Masalah.....	3
BAB II LANDASAN TEORI.....	4
2.1    Pengolahan Citra Digital.....	4
2.2    Operasi titik dalam oeningkatan citra .....	4
2.3    Ekstraksi Fitur warna dan tekstur.....	5
2.4    Jaringan syaraf Tiruan(Artifical Neural Network).....	5
2.5    Euclidean Distance dalam Klasifikasi Citra.....	6
BAB III HASIL.....	7
BAB IV PENUTUP .....	23
DAFTAR PUSTAKA .....	24

## **BAB I**

### **PENDAHULUAN**

#### **1.1 Rumusan Masalah**

- a. Bagaimana metode pengolahan citra digital dapat digunakan untuk mengklasifikasikan objek seperti buah berdasarkan warna dan tekstur?
- b. Seberapa efektif metode ekstraksi fitur warna (HSV, LAB) dan tekstur (GLCM, Moment Invariant) dalam membedakan kualitas dan jenis buah?
- c. Bagaimana pengaruh penerapan metode operasi titik (histogram, intensitas, kontras, negasi) dalam meningkatkan kualitas gambar digital?
- d. Sejauh mana akurasi dan efisiensi metode klasifikasi seperti Jaringan Syaraf Tiruan dan Euclidean Distance dalam mengidentifikasi objek berbasis citra?

#### **1.2 Tujuan Masalah**

- a. Menganalisis dan mengevaluasi pemanfaatan metode pengolahan citra digital dalam klasifikasi objek visual, khususnya buah-buahan.
- b. Menjelaskan penerapan teknik ekstraksi fitur warna dan tekstur dalam sistem klasifikasi citra digital.
- c. Menunjukkan efektivitas metode operasi titik dalam perbaikan kualitas citra dengan menggunakan perangkat lunak MATLAB.
- d. Membandingkan kinerja metode klasifikasi seperti Jaringan Syaraf Tiruan dan Euclidean Distance dalam konteks akurasi dan kecepatan proses.

#### **1.3 Manfaat Masalah**

- a. Memberikan gambaran ilmiah tentang penerapan teknologi pengolahan citra dalam dunia nyata, khususnya di bidang pertanian, perdagangan buah, dan sistem cerdas.
- b. Menjadi referensi atau acuan bagi mahasiswa, peneliti, atau praktisi teknologi informasi dalam membangun sistem klasifikasi atau peningkatan kualitas citra berbasis digital.
- c. Menyediakan solusi objektif dan terotomatisasi untuk masalah yang biasanya ditentukan secara subjektif oleh manusia, seperti prediksi rasa buah atau jenisnya.
- d. Meningkatkan efisiensi dan akurasi dalam pengolahan visual objek tanpa intervensi manual, sehingga dapat diterapkan dalam sistem berbasis AI atau IoT.

## **BAB II**

### **LANDASAN TEORI**

#### **2.1. Pengolahan Citra Digital**

Pengolahan citra digital adalah proses mengolah gambar melalui sistem komputer berbasis numerik untuk mendapatkan informasi visual yang berguna. Citra digital terdiri dari matriks dua dimensi yang berisi piksel-piksel dengan nilai intensitas tertentu. Proses ini melibatkan tahapan seperti akuisisi citra, preprocessing, segmentasi, ekstraksi fitur, dan klasifikasi. Dalam konteks penelitian buah, pengolahan citra digunakan untuk mengidentifikasi karakteristik objek seperti warna, bentuk, dan tekstur secara otomatis.

Dalam pengembangan sistem klasifikasi atau peningkatan kualitas gambar, pengolahan citra digital memiliki peran krusial. Citra buah yang buram, terlalu terang, atau mengandung noise dapat memengaruhi akurasi dalam proses identifikasi objek. Oleh karena itu, teknik pengolahan yang tepat akan membantu menyempurnakan tampilan citra sehingga fitur yang relevan dapat dikenali dengan jelas. Penerapan pengolahan citra ini sangat luas, mulai dari bidang pertanian, medis, hingga keamanan digital.

Berbagai studi terdahulu menunjukkan keberhasilan penggunaan pengolahan citra digital dalam klasifikasi buah, deteksi kematangan, dan penentuan kualitas. Contohnya adalah klasifikasi rasa jeruk siam (Lapendy et al., 2024), klasifikasi jenis apel merah (Pah et al., 2021), serta perbaikan kualitas citra menggunakan operasi titik di MATLAB (Renaldo et al., 2022). Dengan menggabungkan metode visual dan komputasional, pengolahan citra dapat mengurangi subjektivitas manusia dalam menilai objek.

#### **2.2. Operasi Titik dalam Peningkatan Kualitas Citra**

Operasi titik atau point operation adalah metode dasar dalam pengolahan citra yang memodifikasi intensitas tiap piksel secara individu. Operasi ini tidak mempertimbangkan piksel tetangga, sehingga sangat cepat dan efisien secara komputasi. Contoh umum operasi titik antara lain histogram equalization, negasi, peningkatan kecerahan (brightness), dan peningkatan kontras. Operasi ini sangat cocok untuk menangani perbaikan kualitas gambar yang mengalami gangguan seperti noise, blur, atau pencahayaan tidak merata.

Dalam penelitian Renaldo et al. (2022), operasi titik terbukti mampu meningkatkan kualitas citra secara signifikan. Histogram equalization digunakan untuk meratakan distribusi keabuan sehingga citra terlihat lebih tajam. Operasi negasi membantu menampilkan citra dalam bentuk invers seperti negatif film, sementara intensitas dan kontras mengontrol seberapa terang atau gelap suatu citra ditampilkan. Seluruh proses ini dilakukan dalam MATLAB karena kemudahan dan kelengkapan fungsi citra yang disediakan.

Efek dari operasi titik sangat berpengaruh terhadap hasil akhir citra. Gambar yang awalnya kurang informatif menjadi lebih jelas setelah dilakukan transformasi. Operasi titik sering menjadi langkah awal sebelum tahap segmentasi atau klasifikasi dilakukan. Dengan demikian, meskipun sederhana, operasi titik memainkan peran penting dalam pipeline pengolahan citra digital secara keseluruhan.

### **2.3. Ekstraksi Fitur Warna dan Tekstur**

Ekstraksi fitur dalam pengolahan citra merupakan proses untuk mengambil karakteristik penting dari suatu gambar yang dapat digunakan sebagai parameter klasifikasi. Dua fitur yang paling umum digunakan adalah warna dan tekstur. Warna bisa diekstrak dari ruang warna RGB, HSV, atau LAB, sedangkan tekstur biasanya dianalisis menggunakan metode seperti GLCM (Gray Level Co-occurrence Matrix) atau Moment Invariant. Kedua fitur ini sering digunakan untuk mengidentifikasi jenis objek yang memiliki karakteristik visual berbeda.

Dalam studi oleh Lapendy et al. (2024), digunakan fitur warna LAB dan tekstur seperti contrast, correlation, energy, dan homogeneity untuk mengklasifikasikan rasa jeruk. Warna LAB dipilih karena mampu menggambarkan persepsi warna manusia lebih baik dibanding RGB. Sementara itu, GLCM digunakan untuk menganalisis tekstur permukaan kulit jeruk. Kombinasi antara warna dan tekstur terbukti memberikan akurasi yang lebih tinggi dibanding hanya menggunakan satu jenis fitur saja.

Penelitian oleh Pah et al. (2021) juga menggunakan fitur warna HSV untuk membedakan jenis apel merah. Dalam penelitian tersebut, citra dikonversi dari RGB ke HSV, kemudian diambil nilai rata-rata dari masing-masing komponen. Untuk fitur bentuk, digunakan metode moment invariant yang tidak terpengaruh oleh rotasi, skala, atau translasi. Kedua fitur ini digunakan sebagai input untuk proses klasifikasi dengan metode Euclidean Distance, yang menghasilkan akurasi hampir 99%.

### **2.4. Jaringan Syaraf Tiruan (Artificial Neural Network)**

Jaringan Syaraf Tiruan (JST) adalah model komputasi yang terinspirasi dari cara kerja otak manusia. JST digunakan untuk menyelesaikan masalah klasifikasi, regresi, dan pengenalan pola dengan cara mempelajari hubungan antar data melalui proses pelatihan. Dalam konteks pengolahan citra, JST berperan sebagai pengklasifikasi yang mengelompokkan gambar berdasarkan fitur-fitur seperti warna, bentuk, atau tekstur. JST terdiri dari lapisan input, lapisan tersembunyi (hidden layer), dan lapisan output. Masing-masing lapisan ini memiliki neuron yang saling terhubung dengan bobot tertentu.

Dalam penelitian oleh Lapendy et al. (2024), JST digunakan untuk mengklasifikasikan rasa jeruk siam (manis atau asam) berdasarkan gabungan fitur warna LAB dan tekstur. Model JST dalam penelitian tersebut memiliki dua hidden layer dengan 10 dan 5 neuron, serta menggunakan algoritma pelatihan Levenberg-Marquardt dan fungsi aktivasi log sigmoid. Proses pelatihan dilakukan dengan membagi data menjadi 80% untuk latih dan 20% untuk uji. Hasil yang diperoleh menunjukkan akurasi sangat tinggi, yakni 98,75%.

Keunggulan JST terletak pada kemampuannya dalam menangani data non-linear dan kompleks, serta kemampuan generalisasi yang tinggi terhadap data baru. Meskipun memerlukan proses pelatihan yang cukup lama, hasil yang diperoleh sangat memuaskan apabila parameter dan fitur input dipilih dengan tepat. Oleh karena itu, JST menjadi salah satu metode favorit dalam klasifikasi berbasis citra digital, termasuk dalam bidang pertanian, medis, dan industri.

### **2.5. Euclidean Distance dalam Klasifikasi Citra**

Euclidean Distance adalah salah satu metode pengukuran jarak yang paling sederhana dan sering digunakan dalam klasifikasi citra berbasis fitur numerik. Rumus dasar Euclidean Distance menghitung akar kuadrat dari jumlah selisih kuadrat antar elemen fitur dua citra. Semakin kecil jarak antara dua vektor fitur, semakin besar kemungkinan bahwa kedua citra tersebut berada dalam kelas yang sama. Metode ini digunakan untuk klasifikasi berbasis perbandingan langsung antar nilai fitur.

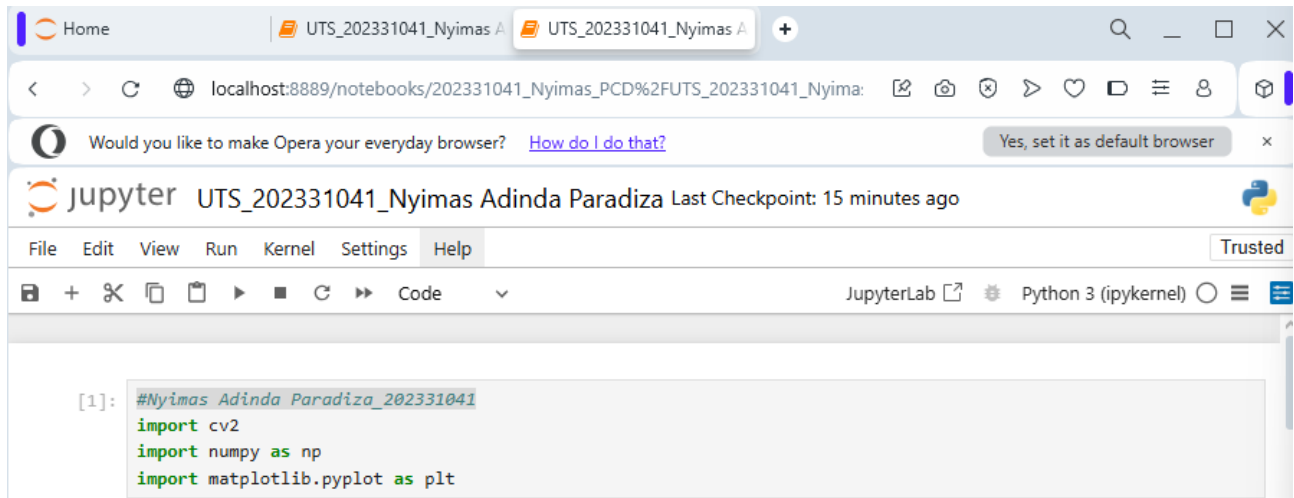
Dalam penelitian Pah et al. (2021), metode Euclidean Distance digunakan untuk mengklasifikasikan tiga jenis apel merah berdasarkan 10 fitur (3 fitur warna HSV dan 7 fitur bentuk moment invariant). Data dibagi menggunakan metode k-fold cross validation, dan hasilnya menunjukkan rata-rata akurasi tinggi sebesar 98,82%. Hal ini menunjukkan bahwa dengan fitur yang tepat, metode sederhana seperti Euclidean Distance tetap mampu menghasilkan hasil klasifikasi yang sangat akurat.

Keunggulan utama dari metode ini adalah kesederhanaan dan kecepatan komputasi. Metode ini tidak memerlukan proses pelatihan model seperti JST, sehingga sangat cocok untuk sistem yang membutuhkan respon cepat dan dataset dengan ukuran relatif kecil. Namun, kelemahannya adalah sensitivitas terhadap skala fitur, sehingga normalisasi atau standarisasi nilai fitur sering diperlukan sebelum digunakan dalam pengukuran jarak. Meski begitu, metode ini masih sangat relevan dalam banyak aplikasi pengolahan citra digital.

## BAB III

## HASIL

### 1. Praktikum Pertama



Disini saya akan jelaskan satu per satu :

- import cv2
  - o Modul: OpenCV (Open Source Computer Vision Library)
  - o Fungsi: Digunakan untuk berbagai operasi pengolahan citra dan video seperti:
    - Membaca/mengambil gambar dan video (cv2.imread(), cv2.VideoCapture())
    - Mengubah ukuran gambar (cv2.resize())
    - Deteksi wajah, tepi (edges), bentuk, dan objek lainnya
    - Mengakses kamera secara langsung
- import numpy as np
  - o Modul: NumPy (Numerical Python)
  - o Fungsi: Untuk perhitungan numerik dan manipulasi array, terutama array 2D dan 3D yang digunakan dalam citra digital.
    - Gambar digital disimpan sebagai array numerik (matriks piksel)
    - NumPy sangat berguna untuk operasi matematika pada citra
- import matplotlib.pyplot as plt
  - o Modul: Matplotlib (bagian pyplot)
  - o Fungsi: Untuk membuat grafik dan menampilkan citra dalam bentuk visual interaktif.
    - natif dari cv2.imshow(), lebih cocok untuk Jupyter Notebook

•[3]:

```
#Nyimas Adinda Paradiza_202331041
img = cv2.imread('Nama.jpg')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

- `img = cv2.imread('Nama.jpg')`
  - Fungsi: Membaca file gambar 'Nama.jpg' dari direktori saat ini.
  - Hasil: Menghasilkan array NumPy berukuran (tinggi, lebar, 3) yang mewakili gambar berwarna dalam format BGR (Blue, Green, Red), bukan RGB.
  - Catatan: Pastikan file 'Nama.jpg' ada di folder yang sama dengan script Python ini.
- `img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)`
  - Fungsi: Mengubah format warna gambar dari BGR (default OpenCV) menjadi RGB (standar umum seperti di Matplotlib).
  - Alasan: Jika kamu ingin menampilkan gambar dengan matplotlib.pyplot, kamu perlu mengubah BGR → RGB agar warna tidak terbalik.
- `gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)`
  - Fungsi: Mengubah gambar berwarna menjadi grayscale (hitam putih).
  - Hasil: Array 2D NumPy dengan 1 kanal (channel), biasanya digunakan untuk:
  - Pendeteksian tepi, objek, atau wajah
  - Mempercepat proses pengolahan citra karena hanya 1 channel

•[4]:

```
#Nyimas Adinda Paradiza_202331041
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

- HSV = Hue, Saturation, Value
  - Hue: Warna dasar (0–179 dalam OpenCV)
  - Saturation: Seberapa kuat/pekat warnanya (0–255)
  - Value: Keterangan/terangnya warna (0–255)

```
•[5]: #Nyimas Adinda Paradiza_202331041
mask_blue = cv2.inRange(hsv, np.array([100, 50, 50]), np.array([140, 255, 255]))
mask_red = cv2.inRange(hsv, np.array([0, 50, 50]), np.array([10, 255, 255])) + \
          cv2.inRange(hsv, np.array([160, 50, 50]), np.array([180, 255, 255]))
mask_green = cv2.inRange(hsv, np.array([40, 50, 50]), np.array([80, 255, 255]))
```

- Masker Biru :
  - Mendeteksi warna biru dalam rentang Hue: 100–140
  - Saturation & Value: 50–255 → menghindari bagian yang terlalu gelap atau pucat
- Masker Merah :
  - Warna merah terletak di ujung kiri dan kanan skala Hue HSV:
    - [0–10] dan [160–180]
  - Jadi, perlu dua rentang, lalu dijumlahkan (+) agar digabung jadi satu masker.
- Masker Hijau
  - Mendeteksi warna hijau dalam rentang Hue: 40–80



```
[6]: #Nyimas Adinda Paradiza_202331041
def highlight_color(mask, gray_img):
    result = gray_img.copy()
    result = cv2.normalize(result, None, None, 255, cv2.NORM_MINMAX)
    result[mask != 0] = 245
    return result
```

- def highlight\_color(mask, gray\_img):
  - o Fungsi: Mendefinisikan fungsi bernama highlight\_color dengan dua parameter:
    - mask: citra biner (hitam-putih), hasil dari deteksi warna (dengan cv2.inRange)
    - gray\_img: gambar grayscale (citra satu channel)
- result = gray\_img.copy()
  - o Fungsi: Membuat salinan dari gambar grayscale asli.
  - o Tujuan: agar gambar asli tidak berubah saat fungsi memodifikasi gambar.
- result = cv2.normalize(result, None, None, 255, cv2.NORM\_MINMAX)
  - o Fungsi: Menormalkan nilai piksel gambar (result) agar tersebar dari nilai minimum hingga 255.
  - o Efek: Meningkatkan kontras gambar grayscale.
  - o Catatan: Pemakaian None, None, 255 seharusnya secara umum lebih tepat.
- result[mask != 0] = 245
  - o Fungsi: Mengubah nilai piksel pada lokasi yang terdeteksi oleh mask (mask ≠ 0) menjadi 245 (sangat terang).
  - o Ini seperti menyorot atau mewarnai terang area warna tertentu dalam gambar grayscale.
- return result
  - o Fungsi: Mengembalikan gambar hasil akhir (result) yang sudah disorot bagian-bagian tertentu sesuai masker.

```
[7]: #Nyimas Adinda Paradiza_202331041
gray_blue = highlight_color(mask_blue, gray)
gray_red = highlight_color(mask_red, gray)
gray_green = highlight_color(mask_green, gray)
```

- gray\_blue = highlight\_color(mask\_blue, gray)
  - o Memanggil fungsi highlight\_color() untuk masker biru.
  - o Area gambar gray yang terdeteksi sebagai biru oleh mask\_blue akan disorot (diberi nilai piksel 245).
  - o Hasilnya disimpan dalam variabel gray\_blue.
- gray\_red = highlight\_color(mask\_red, gray)
  - o Memanggil fungsi untuk masker merah.
  - o Area yang sesuai warna merah disorot terang pada gambar grayscale.
  - o Hasil disimpan ke gray\_red.
- gray\_green = highlight\_color(mask\_green, gray)
  - o Melakukan hal serupa untuk warna hijau.
  - o Menyorot area warna hijau pada gambar gray, hasil disimpan dalam gray\_green.

```
[8]: #Nyimas Adinda Paradiza_202331041
fig, axs = plt.subplots(2, 2, figsize=(12, 9))

axs[0, 0].imshow(img_rgb)
axs[0, 0].set_title("CITRA KONTRAS")
axs[0, 0].axis("off")

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

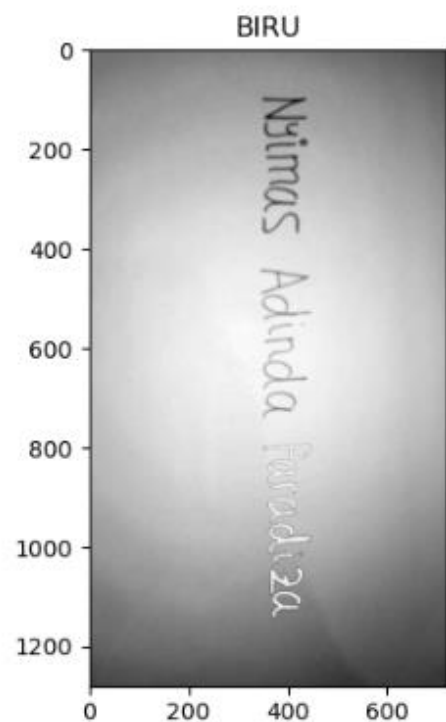
axs[1, 0].imshow(gray_red, cmap='gray')
axs[1, 0].set_title("MERAH")
axs[1, 0].axis("on")

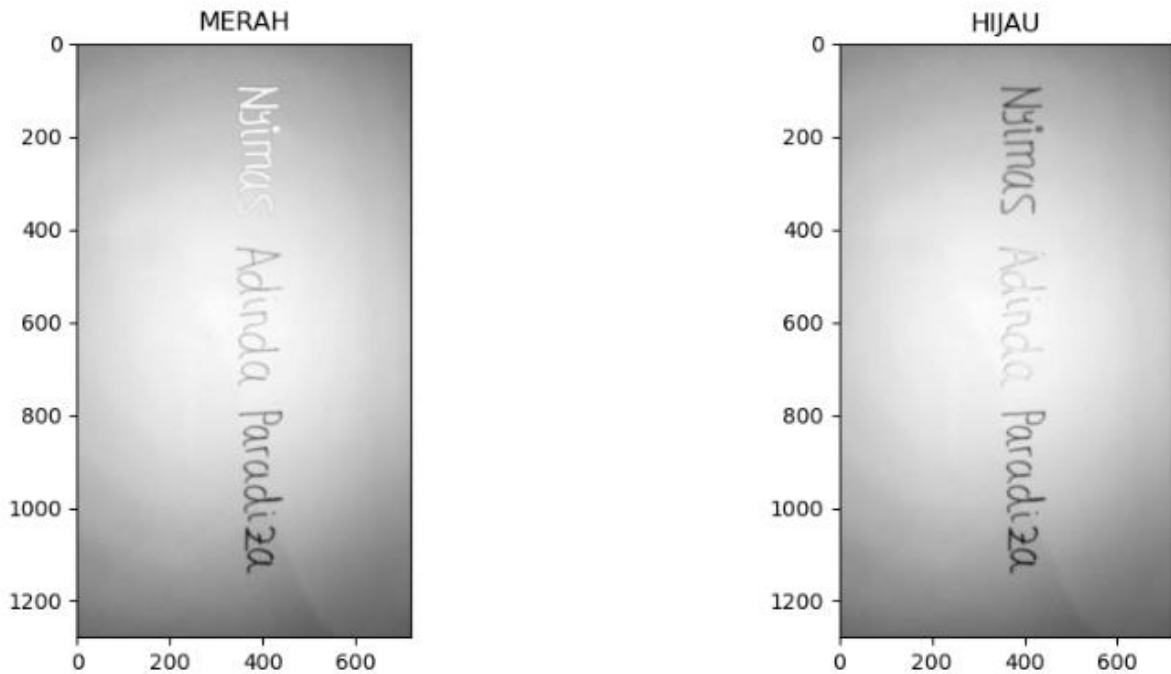
axs[1, 1].imshow(gray_green, cmap='gray')
axs[1, 1].set_title("HIJAU")
axs[1, 1].axis("on")

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

- `fig, axs = plt.subplots(2, 2, figsize=(12, 9))`
  - o Fungsi: Membuat grid subplots 2x2 dengan ukuran gambar 12x9 inci.
    - `fig`: objek figure yang berisi keseluruhan gambar.
    - `axs`: array 2D (2x2) berisi axis untuk setiap subplot.
- `axs[0, 0].imshow(img_rgb)`
  - o Fungsi: Menampilkan gambar `img_rgb` pada posisi subplot pertama (baris 0, kolom 0).
  - o Penjelasan: `imshow` digunakan untuk menampilkan citra, dalam hal ini gambar dengan format RGB (hasil konversi dari BGR ke RGB).
- `axs[0, 0].set_title("CITRA KONTRAS")`
  - o Fungsi: Menetapkan judul subplot pertama menjadi "CITRA KONTRAS".
- `axs[0, 0].axis("off")`
  - o Fungsi: Menonaktifkan axis di subplot pertama agar tidak ada angka atau grid yang ditampilkan.
- `axs[0, 1].imshow(gray_blue, cmap='gray')`
  - o Fungsi: Menampilkan gambar `gray_blue` (hasil highlight warna biru) di subplot posisi baris 0, kolom 1.
  - o `cmap='gray'`: Menggunakan colormap grayscale karena gambar ini adalah gambar biner atau grayscale.
- `axs[0, 1].set_title("BIRU")`
  - o Fungsi: Menetapkan judul subplot untuk gambar biru menjadi "BIRU".
- `axs[0, 1].axis("on")`
  - o Fungsi: Menampilkan axis di subplot biru, sehingga angka dan grid akan terlihat.
- `axs[1, 0].imshow(gray_red, cmap='gray')`
  - o Fungsi: Menampilkan gambar `gray_red` (highlight warna merah) pada subplot posisi baris 1, kolom 0.
- `axs[1, 0].set_title("MERAH")`
  - o Fungsi: Menetapkan judul subplot merah menjadi "MERAH".
- `axs[1, 0].axis("on")`

- Fungsi: Menampilkan axis di subplot merah.
- `axs[1, 1].imshow(gray_green, cmap='gray')`
  - Fungsi: Menampilkan gambar `gray_green` (highlight warna hijau) pada subplot posisi baris 1, kolom 1.
- `axs[1, 1].set_title("HIJAU")`
  - Fungsi: Menetapkan judul subplot hijau menjadi "HIJAU".
- `axs[1, 1].axis("on")`
  - Fungsi: Menampilkan axis di subplot hijau.
- `plt.tight_layout()`
  - Fungsi: Mengatur tata letak subplot agar tidak ada yang tumpang tindih atau terpotong.
- `plt.show()`
  - Fungsi: Menampilkan semua subplot pada layar.





```
[9]: #Nyimas Adinda Paradiza_202331041
fig2, axs2 = plt.subplots(1, 3, figsize=(15, 4))

axs2[0].hist(gray_blue.ravel(), bins=256, range=(0, 256), color='blue', alpha=0.7)
axs2[0].set_title("Histogram - BIRU")

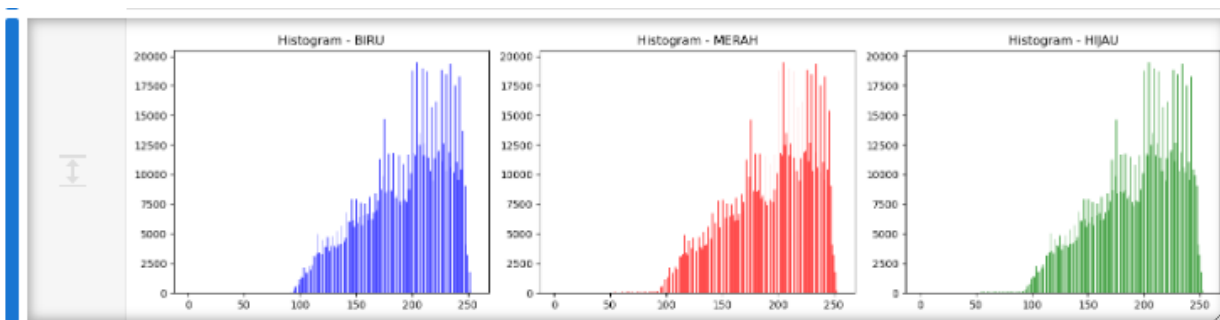
axs2[1].hist(gray_red.ravel(), bins=256, range=(0, 256), color='red', alpha=0.7)
axs2[1].set_title("Histogram - MERAH")

axs2[2].hist(gray_green.ravel(), bins=256, range=(0, 256), color='green', alpha=0.7)
axs2[2].set_title("Histogram - HIJAU")

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

- `fig2, axs2 = plt.subplots(1, 3, figsize=(15, 4))`
  - Fungsi: Membuat satu baris subplot dengan tiga kolom.
  - `fig2` adalah objek figure yang berisi seluruh gambar, dan `axs2` adalah array 1D yang berisi tiga axis (untuk masing-masing histogram warna biru, merah, dan hijau).
  - Ukuran figure ditetapkan menjadi 15x4 inci.
- `axs2[0].hist(gray_blue.ravel(), bins=256, range=(0, 256), color='blue', alpha=0.7)`
  - Fungsi: Membuat histogram dari gambar biru (`gray_blue`) dengan:
    - `ravel()` digunakan untuk meratakan array citra menjadi satu dimensi.
    - `bins=256`: Membagi nilai pixel ke dalam 256 interval (karena rentang nilai piksel dari 0 hingga 255).
    - `range=(0, 256)`: Rentang nilai piksel.
    - `color='blue'`: Memberikan warna biru pada histogram.
    - `alpha=0.7`: Mengatur transparansi histogram, sehingga lebih terlihat jelas jika ada overlay atau lebih halus.
  - `set_title("Histogram - BIRU")`: Memberi judul pada histogram biru.

- `axs2[1].hist(gray_red.ravel(), bins=256, range=(0, 256), color='red', alpha=0.7)`
  - o Fungsi: Membuat histogram untuk gambar merah (`gray_red`), dengan pengaturan serupa seperti histogram biru, namun dengan warna merah.
  - o `set_title("Histogram - MERAH")`: Memberi judul pada histogram merah.
- `axs2[2].hist(gray_green.ravel(), bins=256, range=(0, 256), color='green', alpha=0.7)`
  - o Fungsi: Membuat histogram untuk gambar hijau (`gray_green`), dengan pengaturan serupa seperti histogram biru dan merah, namun dengan warna hijau.
  - o `set_title("Histogram - HIJAU")`: Memberi judul pada histogram hijau.
- `plt.tight_layout()`
  - o Fungsi: Menyesuaikan tata letak subplot agar elemen-elemen di dalam figure tidak tumpang tindih dan ada jarak yang cukup antar subplots.
- `plt.show()`
  - o Fungsi: Menampilkan figure yang berisi tiga histogram dalam satu jendela.



## 2. Bagian kedua

```
[10]: #Nyimas Adinda Paradiza_202331041
def make_binary(mask):
    binary = np.zeros_like(mask)
    binary[mask > 0] = 255
    return binary
```

- `def make_binary(mask):`
  - o Fungsi ini bernama `make_binary`, yang menerima parameter `mask`.
  - o Parameter `mask` adalah gambar biner (biasanya hasil dari deteksi warna atau objek), yang mungkin berisi nilai-nilai selain 0 untuk area yang terdeteksi.
- `binary = np.zeros_like(mask)`
  - o Membuat array `binary` dengan bentuk dan tipe data yang sama persis seperti `mask`, namun semua elemen diinisialisasi ke 0.
  - o Ini menciptakan gambar kosong berisi nilai 0 di semua posisi.
- `binary[mask > 0] = 255`
  - o Untuk setiap piksel dalam `mask` yang nilainya lebih besar dari 0 (artinya piksel tersebut terdeteksi), ubah nilai piksel tersebut menjadi 255 pada array `binary`.
  - o Dengan kata lain, ini mengubah semua nilai non-zero di `mask` menjadi putih (255) di citra `binary`, sedangkan nilai 0 di `mask` tetap 0 di `binary`.
- `return binary`
  - o Mengembalikan gambar biner `binary`, yang sekarang berisi 255 untuk area yang terdeteksi dan 0 untuk area lainnya.

```
[11]: #Nyimas Adinda Paradiza_202331041
color_thresholds = [
    {"color": "Merah", "lower": [0, 50, 50], "upper": [10, 255, 255]},
    {"color": "Hijau", "lower": [40, 50, 50], "upper": [80, 255, 255]},
    {"color": "Biru", "lower": [100, 50, 50], "upper": [140, 255, 255]},
    {"color": "Merah", "lower": [160, 50, 50], "upper": [180, 255, 255]},
]
```

- Ini adalah list Python berisi beberapa dictionary, masing-masing berisi:
  - o "color": Nama warna yang ingin dideteksi.
  - o "lower": Nilai HSV bawah sebagai ambang batas minimum deteksi warna.
  - o "upper": Nilai HSV atas sebagai ambang batas maksimum deteksi warna.

```
[12]: #Nyimas Adinda Paradiza_202331041
color_thresholds_sorted = sorted(color_thresholds, key=lambda x: x["lower"][0])
```

- sorted(...): Fungsi Python untuk mengurutkan list.
- color\_thresholds: List yang berisi dictionary dengan informasi warna (color, lower, upper).
- key=lambda x: x["lower"][0]:
  - o Ini adalah fungsi kunci (key function) untuk menentukan kriteria pengurutan.
  - o x["lower"][0] berarti ambil elemen pertama dari nilai lower, yaitu nilai Hue (H) dari ambang bawah HSV.
  - o Pengurutan dilakukan berdasarkan nilai H (Hue) secara menaik.

```
[13]: #Nyimas Adinda Paradiza_202331041
print("Daftar Ambang Batas HSV (diurutkan berdasarkan Hue):")
for i, item in enumerate(color_thresholds_sorted, 1):
    print(f"{i}. Warna: {item['color']}")
    print(f"    Lower HSV: {item['lower']}")
    print(f"    Upper HSV: {item['upper']}\n")
```

Daftar Ambang Batas HSV (diurutkan berdasarkan Hue):

1. Warna: Merah  
 Lower HSV: [0, 50, 50]  
 Upper HSV: [10, 255, 255]
2. Warna: Hijau  
 Lower HSV: [40, 50, 50]  
 Upper HSV: [80, 255, 255]
3. Warna: Biru  
 Lower HSV: [100, 50, 50]  
 Upper HSV: [140, 255, 255]
4. Warna: Merah  
 Lower HSV: [160, 50, 50]  
 Upper HSV: [180, 255, 255]

- print("Daftar Ambang Batas HSV (diurutkan berdasarkan Hue):")
  - o Mencetak judul atau header untuk daftar warna yang akan ditampilkan.
- for i, item in enumerate(color\_thresholds\_sorted, 1):

- Melakukan iterasi terhadap `color_thresholds_sorted` yang sudah diurutkan berdasarkan nilai hue.
- `enumerate(..., 1)` menghasilkan pasangan (`index`, `item`), dimulai dari angka 1.
- `print(f"{i}. Warna: {item['color']}")`
  - Menampilkan nomor urut dan nama warna.
- `print(f" Lower HSV: {item['lower']}")`
  - Menampilkan nilai batas bawah HSV untuk warna tersebut.
- `print(f" Upper HSV: {item['upper']}\n")`
  - Menampilkan nilai batas atas HSV untuk warna tersebut, lalu memberi spasi baris.

[15]:

```
#Nyimas Adinda Paradiza_202331041
img = cv2.imread('Nama.jpg')
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

- `img = cv2.imread('Nama.jpg')`
  - Membaca file gambar bernama 'Nama.jpg' dari direktori saat ini.
  - `cv2.imread()` secara default membaca gambar dalam format BGR (Blue, Green, Red) — bukan RGB.
  - Variabel `img` sekarang berisi array NumPy berdimensi (tinggi, lebar, 3).
- `hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)`
  - Mengubah gambar dari format warna BGR ke HSV.
  - HSV (Hue, Saturation, Value) lebih cocok untuk deteksi warna karena:
    - Hue mewakili warna (misalnya merah, hijau, biru) dalam derajat (0–179 di OpenCV).
    - Saturation adalah intensitas warna.
    - Value adalah tingkat kecerahan.
  - Hasil konversi disimpan di variabel `hsv`, yang bisa digunakan untuk masking warna seperti merah, hijau, atau biru dengan `cv2.inRange()`.

[16]:

```
#Nyimas Adinda Paradiza_202331041
mask_red = cv2.inRange(hsv, np.array([0, 50, 50]), np.array([10, 255, 255])) + \
            cv2.inRange(hsv, np.array([160, 50, 50]), np.array([180, 255, 255]))
mask_green = cv2.inRange(hsv, np.array([40, 50, 50]), np.array([80, 255, 255]))
mask_blue = cv2.inRange(hsv, np.array([100, 50, 50]), np.array([140, 255, 255]))
```

- [17]:

- `Mask_red`
  - Warna merah dalam HSV terbagi dua rentang (karena hue merah ada di dua sisi spektrum HSV: 0–10 dan 160–180).
  - `cv2.inRange` akan menghasilkan masker biner (nilai 0 atau 255) di mana piksel jatuh dalam rentang tersebut.
  - Tanda + artinya menggabungkan dua masker merah jadi satu.
- `Mask_green`
  - Mendeteksi warna hijau.

- Hue antara 40–80 mencakup berbagai jenis hijau.
- Mask\_blue
  - Mendeteksi warna biru.
  - Hue antara 100–140 mencakup biru terang hingga biru gelap.

•[17]:

```
none = np.zeros_like(mask_blue)
only_blue = mask_blue
red_blue = cv2.bitwise_or(mask_red, mask_blue)
rgb = cv2.bitwise_or(red_blue, mask_green)
#Nyimas Adinda Paradiza_202331041
```

- none = np.zeros\_like(mask\_blue)
  - Membuat array kosong (semua nol) dengan ukuran dan tipe yang sama dengan mask\_blue.
  - Bisa digunakan sebagai latar belakang hitam atau referensi awal masker yang tidak berisi apapun.
- only\_blue = mask\_blue
  - Menyimpan masker biru ke dalam variabel baru. Ini hanya untuk kemudahan akses atau visualisasi.
- red\_blue = cv2.bitwise\_or(mask\_red, mask\_blue)
  - Menggabungkan masker merah dan biru menggunakan operasi OR bitwise.
  - Hasilnya: piksel yang memiliki warna merah atau biru akan bernilai 255 (putih); lainnya tetap 0 (hitam).
- rgb = cv2.bitwise\_or(red\_blue, mask\_green)
  - Menggabungkan hasil sebelumnya (red\_blue) dengan masker hijau.
  - Hasilnya adalah masker gabungan untuk merah, hijau, dan biru, mewakili seluruh bagian gambar yang berwarna salah satu dari tiga itu.

•[18]:

```
img_none = make_binary(none)
img_blue = make_binary(only_blue) #Nyimas Adinda Paradiza_202331041
img_rb = make_binary(red_blue)
img_rgb = make_binary(rgb)
```

- img\_none = make\_binary(none)
  - Mengubah masker kosong (semua nol) menjadi gambar biner (akan tetap hitam semua).
  - Hasilnya: citra biner tanpa objek (semua piksel = 0).
- img\_blue = make\_binary(only\_blue)
  - Mengubah masker biru menjadi citra biner.
  - Piksel yang mendeteksi biru akan menjadi putih (255), lainnya hitam (0).
- img\_rb = make\_binary(red\_blue)
  - Mengubah masker gabungan merah dan biru menjadi citra biner.
  - Menyoroti area pada gambar yang mengandung merah atau biru.
- img\_rgb = make\_binary(rgb)



- Mengubah masker gabungan merah, hijau, dan biru menjadi gambar biner.
- Artinya, area berwarna merah, hijau, atau biru akan ditampilkan putih (255), lainnya hitam.

[20]:

```
fig, axs = plt.subplots(2, 2, figsize=(10, 10))
axs[0, 0].imshow(img_none, cmap='gray')
axs[0, 0].set_title("NONE")
axs[0, 0].axis("off")

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

axs[1, 0].imshow(img_rb, cmap='gray') #Nyimas Adinda Paradiza_202331041
axs[1, 0].set_title("RED-BLUE")
axs[1, 0].axis("off")

axs[1, 1].imshow(img_rgb, cmap='gray')
axs[1, 1].set_title("RED-GREEN-BLUE")
axs[1, 1].axis("off")

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

- fig, axs = plt.subplots(2, 2, figsize=(10, 10))
  - Membuat figure dengan 4 area gambar (2 baris x 2 kolom).
  - Ukuran figure diatur ke 10x10 inci.
- axs[0, 0].imshow(img\_none, cmap='gray')
  - axs[0, 0].set\_title("NONE")
  - axs[0, 0].axis("off")
    - Menampilkan gambar biner kosong (img\_none) di posisi kiri atas.
    - cmap='gray' untuk tampilan hitam-putih.
    - axis("off") menyembunyikan sumbu koordinat.
- axs[0, 1].imshow(img\_blue, cmap='gray')
  - axs[0, 1].set\_title("BLUE")
  - axs[0, 1].axis("off")
    - Menampilkan hasil deteksi warna biru saja.
- axs[1, 0].imshow(img\_rb, cmap='gray') # Nyimas Adinda Paradiza\_202331041
  - axs[1, 0].set\_title("RED-BLUE")
  - axs[1, 0].axis("off")
    - Menampilkan hasil gabungan deteksi merah dan biru.
- axs[1, 1].imshow(img\_rgb, cmap='gray')
  - axs[1, 1].set\_title("RED-GREEN-BLUE")
  - axs[1, 1].axis("off")
    - Menampilkan hasil gabungan ketiga warna: merah, hijau, dan biru.
- plt.tight\_layout()
  - plt.show()
    - Menyesuaikan tata letak agar tidak saling tumpang tindih.
    - Menampilkan semua plot dalam satu jendela.



### 3. Praktikum ke

[1]:

```
import cv2
import numpy as np #Nyimas Adinda Paradiza_202331041
import matplotlib.pyplot as plt
```

- import cv2
  - o Mengimpor OpenCV (Open Source Computer Vision Library), yang digunakan untuk berbagai operasi pengolahan gambar dan video.
  - o Fungsi penting seperti cv2.imread(), cv2.cvtColor(), cv2.inRange(), cv2.bitwise\_or(), dan sebagainya berasal dari library ini.
- import numpy as np # Nyimas Adinda Paradiza\_202331041
  - o Mengimpor NumPy, library Python untuk operasi array dan numerik.
  - o Banyak digunakan dalam pengolahan citra karena citra digital disimpan sebagai array (matriks piksel).
  - o Digunakan untuk membuat array, manipulasi nilai, dan logika seperti:
    - np.array(), np.zeros\_like(), mask > 0, dsb.
- import matplotlib.pyplot as plt
  - o Mengimpor Matplotlib Pyplot, library untuk membuat grafik dan visualisasi gambar.
  - o Sangat berguna untuk:
    - Menampilkan gambar (plt.imshow)
    - Membuat histogram (plt.hist)
    - Menyusun banyak gambar dalam satu figure (plt.subplots)

[5]:

```
image_path = 'Gambar Diri.jpg'
img = cv2.imread(image_path) #Nyimas Adinda Paradiza_202331041
```

- image\_path = 'Gambar Diri.jpg'
  - o Menyimpan nama atau path (lokasi) file gambar ke dalam variabel image\_path.
  - o Gambar harus berada di folder kerja (working directory) atau beri path lengkap jika di folder lain.
- img = cv2.imread(image\_path)
  - o Membaca gambar dari path yang diberikan dan menyimpannya dalam variabel img sebagai array NumPy (matriks piksel).
  - o Gambar dibaca dalam format BGR secara default (bukan RGB).

[9]:

```
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) #Nyimas Adinda Paradiza_202331041
```

- cv2.cvtColor() adalah fungsi dari OpenCV untuk mengubah format warna gambar.
- img adalah gambar asli yang dibaca menggunakan cv2.imread(), yang secara default dalam format BGR (Blue-Green-Red).
- cv2.COLOR\_BGR2RGB menunjukkan bahwa kita ingin mengubah gambar dari BGR ke RGB, agar warnanya benar saat ditampilkan dengan matplotlib.pyplot.

[10]:

```

# 1. Gambar Asli
original_img = img_rgb

# 2. Grayscale (Abu-abu)
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray_rgb = cv2.cvtColor(gray_img, cv2.COLOR_GRAY2RGB)

# 3. Grayscale + Pencerahan
bright_gray = cv2.convertScaleAbs(gray_img, alpha=1.2, beta=50)
bright_gray_rgb = cv2.cvtColor(bright_gray, cv2.COLOR_GRAY2RGB)

# 4. Grayscale + Kontras
contrast_gray = cv2.convertScaleAbs(gray_img, alpha=2, beta=0)
contrast_gray_rgb = cv2.cvtColor(contrast_gray, cv2.COLOR_GRAY2RGB)

# 5. Grayscale + Pencerahan + Kontras
bright_contrast_gray = cv2.convertScaleAbs(gray_img, alpha=2, beta=50)
bright_contrast_gray_rgb = cv2.cvtColor(bright_contrast_gray, cv2.COLOR_GRAY2RGB)

#Nyimas Adinda Paradiza_202331041

```

- original\_img = img\_rgb
  - o Menyimpan gambar asli (dalam format RGB) ke variabel original\_img.
- gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 gray\_rgb = cv2.cvtColor(gray\_img, cv2.COLOR\_GRAY2RGB)
  - o gray\_img: Gambar diubah ke grayscale → hanya satu channel (intensitas terang-gelap).
  - o gray\_rgb: Dikonversi lagi ke RGB untuk kompatibel dengan matplotlib, meskipun warnanya tetap abu-abu.
- bright\_gray = cv2.convertScaleAbs(gray\_img, alpha=1.2, beta=50)  
 bright\_gray\_rgb = cv2.cvtColor(bright\_gray, cv2.COLOR\_GRAY2RGB)
  - o alpha = 1.2: Meningkatkan kontras sedikit.
  - o beta = 50: Menambahkan pencerahan (nilai pixel ditambah).
  - o Hasilnya gambar grayscale yang lebih cerah.
- contrast\_gray = cv2.convertScaleAbs(gray\_img, alpha=2, beta=0)  
 contrast\_gray\_rgb = cv2.cvtColor(contrast\_gray, cv2.COLOR\_GRAY2RGB)
  - o alpha = 2: Meningkatkan kontras dua kali lipat.
  - o beta = 0: Tidak menambahkan pencerahan.
  - o Hasilnya gambar abu-abu dengan kontras tinggi (gelap makin gelap, terang makin terang).
- bright\_contrast\_gray = cv2.convertScaleAbs(gray\_img, alpha=2, beta=50)  
 bright\_contrast\_gray\_rgb = cv2.cvtColor(bright\_contrast\_gray, cv2.COLOR\_GRAY2RGB)
  - o Kombinasi peningkatan kontras dan pencerahan.
  - o Gambar akan terlihat lebih tajam dan terang.


```
[11]:
plt.figure(figsize=(10, 10))
images = [
    original_img,
    gray_rgb,
    bright_gray_rgb,
    contrast_gray_rgb,
    bright_contrast_gray_rgb
]
titles = [
    "Gambar Asli",
    "Grayscale",
    "Grayscale + Cerah",
    "Grayscale + Kontras",
    "Grayscale yang dipercerah dan di perkontras"
]

for i in range(5):
    plt.subplot(3, 2, i + 1)
    plt.imshow(images[i])
    plt.title(titles[i])
    plt.axis('off')

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

- plt.figure(figsize=(10, 10))
  - o Membuat canvas ukuran 10x10 inci untuk menampung subplot.
- List images dan titles
  - o images: berisi 5 gambar RGB yang sudah diolah sebelumnya.
  - o titles: berisi judul yang sesuai untuk tiap gambar.
- for i in range(5):
  - o Perulangan sebanyak 5 kali untuk menempatkan masing-masing gambar ke subplot.
- plt.subplot(3, 2, i + 1)
  - o Membuat grid subplot 3 baris × 2 kolom.
  - o i + 1: menempatkan gambar pada urutan subplot ke-1 hingga ke-5.
- plt.imshow(images[i])
  - o Menampilkan gambar ke-i.
- plt.title(titles[i])
  - o Memberi judul pada subplot sesuai dengan judul yang sudah disiapkan.
- plt.axis('off')
  - o Menyembunyikan sumbu x dan y agar tampilan lebih bersih.
- plt.tight\_layout() dan plt.show()
  - o tight\_layout(): mengatur jarak antar subplot agar tidak saling tumpang tindih.
  - o show(): menampilkan semua gambar dalam satu window.


Gambar Asli



Kamu, 08 Mei 2023 - 17:24

Info gambar


Grayscale + Cerah



Kamu, 08 Mei 2023 - 17:24

Info gambar

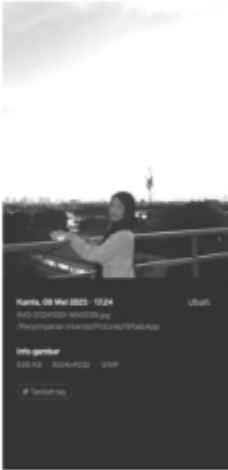
Grayscale + Kontras



Kamu, 08 Mei 2023 - 17:24

Info gambar

Grayscale yang dipercerah dan di perkontras



Kamu, 08 Mei 2023 - 17:24

Info gambar

## **BAB IV**

### **PENUTUP**

Berdasarkan kajian terhadap ketiga jurnal yang membahas pengolahan citra digital, dapat disimpulkan bahwa teknologi ini memiliki peranan penting dalam proses klasifikasi dan peningkatan kualitas gambar. Pengolahan citra digital mampu mengotomatisasi tugas-tugas visual seperti mengidentifikasi rasa buah jeruk, jenis apel, atau memperbaiki gambar buram dan bercacat. Dengan tahapan sistematis mulai dari akuisisi, preprocessing, segmentasi, ekstraksi fitur, hingga klasifikasi, sistem pengolahan citra digital memberikan solusi yang lebih objektif dan konsisten dibanding penilaian manual.

Penerapan metode ekstraksi fitur warna (HSV, LAB) dan tekstur (GLCM, moment invariant) terbukti efektif dalam membedakan objek berdasarkan ciri visualnya. Kombinasi dari kedua jenis fitur ini memberikan representasi yang lebih kuat terhadap citra, sehingga dapat meningkatkan akurasi klasifikasi. Di sisi lain, penggunaan metode klasifikasi seperti Jaringan Syaraf Tiruan (JST) dan Euclidean Distance mampu menghasilkan hasil prediksi yang akurat, masing-masing disesuaikan dengan kompleksitas data dan tujuan aplikasi.

Selain itu, metode operasi titik seperti histogram equalization, kontras, negasi, dan intensitas pada perangkat MATLAB memberikan kontribusi signifikan dalam peningkatan kualitas citra. Citra yang awalnya gelap, buram, atau memiliki noise dapat diperbaiki dengan teknik ini agar lebih layak digunakan untuk proses selanjutnya seperti pengenalan pola atau klasifikasi. Dengan demikian, pengolahan citra digital merupakan teknologi yang sangat bermanfaat untuk diterapkan di berbagai bidang, baik akademik maupun industri.

.

### **DAFTAR PUSTAKA**

- i. Jessica Crisfin Lapendy, et al. (2024). Klasifikasi Rasa Jeruk Siam Berdasarkan Warna dan Tekstur Berbasis Pengolahan Citra Digital. JIPI, Vol. 9 No. 2, Juni 2024.
- ii. Nikotesa E. R. Pah, et al. (2021). Ekstraksi Ciri Warna HSV dan Ciri Bentuk Moment Invariant untuk Klasifikasi Buah Apel Merah. J-ICON, Vol. 9 No. 2.
- iii. Edo Renaldo, et al. (2022). Operasi Titik pada Pengolahan Citra Digital untuk Peningkatan Kualitas Gambar Menggunakan Matlab. MDP Student Conference 2022.