# UAS

# PENGOLAHAN CITRA


NAMA : Muhamad Fauzi Akbar

NIM : 202331192

KELAS : F

DOSEN : Dr.Dra.Dwina Kuswardani,M.Kom

NO.PC : 12

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

# BAB I
# PENDAHULUAN

## 1.1 Rumusan Masalah

Berdasarkan praktikum yang telah dilakukan, rumusan masalah dapat dijabarkan sebagai berikut:

1. Bagaimana melakukan segmentasi objek buah dan daun pada gambar hasil foto sendiri menggunakan teknik masking dan thresholding warna?
2. Bagaimana menerapkan metode deteksi tepi (Canny Edge Detection) dan deteksi kontur (Contours) pada citra wajah dengan ketentuan jarak pengambilan tertentu?
3. Bagaimana mengaplikasikan metode kompresi lossy dan kuantisasi warna RGB dengan 4 level warna/channel untuk mengurangi ukuran dan kompleksitas gambar tanpa menghilangkan informasi visual penting?

## 1.2 Tujuan Masalah

Tujuan dari praktikum ini adalah:

1. Menerapkan teknik segmentasi warna untuk mendeteksi objek daun dan buah secara terpisah pada gambar hasil tangkapan kamera sendiri.
2. Menggunakan metode Canny Edge dan deteksi kontur untuk mengekstraksi bentuk dan pola dari citra wajah manusia secara akurat.
3. Mengurangi ukuran gambar dan kompleksitas warna melalui kompresi lossy dan kuantisasi RGB, serta membandingkan hasil visual dari setiap tahapan proses.

## 1.3 Manfaat Masalah

Manfaat dari praktikum yang dilakukan ini antara lain:

1. Memahami dan mengaplikasikan teknik segmentasi citra berdasarkan rentang warna dalam ruang warna HSV.
2. Mengembangkan kemampuan deteksi fitur tepi dan kontur untuk digunakan dalam aplikasi pengenalan wajah atau objek.
3. Mengetahui prinsip kompresi citra digital untuk efisiensi penyimpanan dan transmisi data multimedia.
4. Melatih keterampilan pemrograman pengolahan citra menggunakan pustaka komputer visi seperti OpenCV.

# BAB II
# LANDASAN TEORI

## 2.1 Citra Digital

Citra digital merupakan representasi dari citra yang dihasilkan melalui proses sampling dan kuantisasi. Sampling menyatakan besar kecilnya ukuran piksel (titik) pada citra, yang disusun dalam bentuk $m$baris dan $n$kolom. Sedangkan kuantisasi menyatakan besarnya nilai tingkat kecerahan yang dinyatakan dalam nilai grayscaleberdasarkan jumlah bit biner yang digunakan. Semakin banyak bit biner, semakin banyak warna yang dapat direpresentasikan oleh citra (Marleny, 2021)(Al Kautsar et al., 2025)

## 2.2 Jenis Citra Digital

Piksel dalam citra digital memiliki nilai dalam rentang tertentu, biasanya dari 0 hingga 255, tergantung pada jenis warnanya. Citra biner, atau monokrom, menggunakan nilai 0 untuk hitam dan 1 untuk putih, di mana 0 menunjukkan tidak adanya cahaya dan 1 menunjukkan adanya cahaya. Citra grayscale menawarkan lebih banyak kemungkinan warna daripada citra biner, tergantung pada jumlah bit yang digunakan. Sebagai contoh, skala keabuan 4 bit memiliki 16 nilai ($0 - 15$), sedangkan skala 8 bit memiliki 256 nilai ($0 - 255$) (Jannah, dkk, 2023).

Citra RGB, menggabungkan tiga komponen warna yaitu merah, hijau, dan biru dengan rentang intensitas $0 - 255$ per komponen, yang memungkinkan hingga 16.581.375 variasi warna (Marleny, 2021). Dalam sistem RGB, warna ditulis sebagai vektor $(r, g, b)$. Jika semua komponen bernilai 0 (0,0,0), warna yang dihasilkan adalah hitam dan jika semua komponen bernilai 255 (255,255,255), warna yang dihasilkan adalah putih (Cahyono, 2017). (Al Kautsar et al., 2025)

## 2.3 Metode Segmentasi

Segmentasi citra adalah proses pembagian citra menjadi beberapa wilayah atau objek yang memiliki atribut atau karakteristik yang serupa, sementara wilayah yang berbeda memiliki atribut yang berbeda. Segmentasi citra melibatkan langkah-langkah untuk memberikan label pada setiap piksel dalam suatu gambar, sehingga piksel yang memiliki karakteristik visual serupa diberikan label yang sama. Pemilihan metode segmentasi citra merupakan langkah penting terkhususnya dalam proses mengelolah data citra digital dan penglihatan komputer. Tujuan

utama dari segmentasi citra adalah untuk menyederhanakan representasi citra ke dalam bentuk yang memiliki makna dan lebih mudah untuk dianalisis. Tujuan dari segmentasi citra yaitu dapat membagi gambar citra ke beberapa sub atau bagian yang disebut dengan segmen dan akan memisahkan daun yang terinfeksi penyakit dari daun yang sehat. Metode thresholding merupakan suatu proses yang mengubah citra dengan skala keabuan menjadi citra biner atau citra hitam putih, dengan tujuan untuk memperjelas dan memisahkan wilayah yang termasuk ke dalam obyek dan latar belakang citra. Algoritma thresholding digunakan untuk memisahkan bagian-bagian dari citra digital sehingga dapat diinterpretasikan sebagai citra yang telah disegmentasi. Deteksi tepi, pada tahapan ini dilakukan untuk menemukan lokasi di mana perubahan intensitas citra secara tajam terjadi. Ini memungkinkan kita untuk mengidentifikasi batas atau tepi antara objek yang diinginkan dan latar belakangnya dalam citra. Deteksi tepi dalam proses segmentasi citra mencakup konsep-konsep berikut: Perubahan intensitas, ketajaman transisi, pembentukkan kontur, operator deteksi tepi dan noise handling[9].Untuk membentuk segmen sesuai dengan warna yang diinginkan maka ditentukan nilai toleransi pada setiap dimensi warna HSV, kemudian nilai toleransi tersebut digunakan dalam perhitungan proses adaptive threshold. Hasil dari proses threshold tersebut akan membentuk segmen area dengan warna sesuai toleransi yang diinginkan. Secara garis besar, gambaran proses segmetnasi dapat dilihat pada Gambar 1 dan berikut ini merupakan proses segmentasi menurut Giannakupoulos (2008). (Enggari et al., 2022)

2.4 Deteksi Tepi dan Kontur

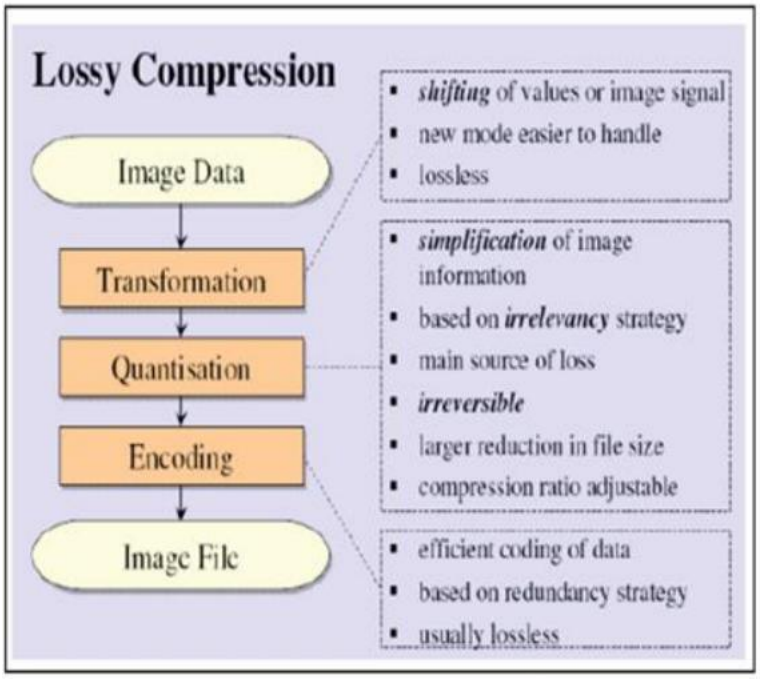
Pendeteksian tepi akan menghasilkan citra tepi yang berupa citra biner (pixel tepi berwarna putih, sedangkan pixel bukan tepi berwarna hitam). Tetapi, tepi belum memberikan informasi yang berguna karena belum terdapat keterkaitan antara suatu tepi dengan tepi lainya. Citra tepi ini harus diproses lebih lanjut untuk menghasilkan informasi yang lebih berguna yang dapat digunakan dalam mendeteksi bentuk-bentuk yang sederhana (misalnya garis lurus, lingkaran, elips, dan sebagainya) pada proses analisis citra. Rangkaian dari pixel-pixel tepi yang membentuk batas daerah (region boundary) disebut kontur (Contour).

Deteksi tepi merupakan salah satu teknik segmentasi gambar yang bertujuan untuk mengidentifikasi garis tepi suatu objek dalam citra. Teknik ini berfungsi untuk menandai bagian-bagian penting dari gambar tersebut. Selain itu, proses ini juga berperan dalam meningkatkan ketajaman detail pada citra yang mengalami kekaburan akibat kesalahan atau efek dari proses akuisisi gambar [12].

Dalam penerapannya, metode deteksi tepi (*edge detection*) mengubah area tertentu dalam citra menjadi dua nilai intensitas, yaitu rendah dan tinggi, yang biasanya direpresentasikan dalam bentuk biner, seperti nol dan satu [13]. (Qisthiano & Pratiwi, 2025)

2.5 Kompresi Lossy

Kompresi merupakan proses pengubahan sekumpulan data menjadi suatu bentuk kode untuk menghemat kebutuhan tempat penyimpanan dan waktu untuk transmisi data. Metode kompresi data dapat dibagi ke dalam dua kelompok besar yaitu (Saragih & Harahap, 2019) : Lossy compression Yaitu suatu metode kompresi data yang menghilangkan sebagian "informasi" dari data asli selama proses kompresi dengan tidak menghilangkan secara signifikan informasi yang ada dalam data secara keseluruhan. Kompresi jenis ini dapat mengurangi ukuran data secara signifikan, dan data yang sudah terkompresi dapat dikompres lagi sampai batasbatas tertentu. Lossy compression biasanya digunakan dalam bidang Multimedia karena sering digunakan Untuk mengurangi ukuran data yang bersifat audio–visual. Misalnya untuk memperkecil ukuran file gambar, musik digital, dan untuk enkoding film dari format High Definition seperti Blu-Ray menjadi format MPEG (Moving Picture Experts Group). Contoh: discrete cosine transform, vector quantization, waveled compression, distributed source coding (DSC). Berikut ini adalah ilustrasi dari kompresi lossy di jelaskan pada gambar 1:




Gambar 1. Lossy compression

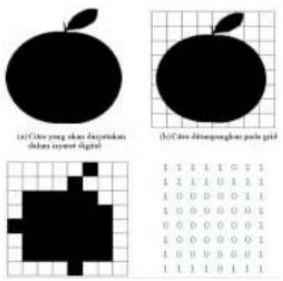
Lossless compression Yaitu suatu metode kompresi data dengan tidak ada "informasi" data yang hilang atau berkurang jumlahnya selama proses kompresi. Sehingga setelah proses dekompresi jumlah bit (byte) data atau informasi dalam keseluruhan data hasil sama persis dengan data aslinya. Kompresi jenis ini tidak selalu dapat mengurangi ukuran data secara berarti, karena tidak semua data mengandung informasi yang tidak perlu. Selain itu, data yang telah dikompresi tidak dapat dikompresi lagi. Lossless compression biasanya digunakan untuk mengurangi ukuran data-data yang penting, data-data yang isinya tidak boleh berubah sedikitpun. Misalnya datadata yang berbentuk dokumen, teks, spreadsheet, kode sumber (source code), dan datadata program. Contoh: Huffman coding, dynamic markov compression (DMC), Lempel-Ziv- Welch (LZW), aritmetic coding, Run-Length encoding. Berikut ini

adalah gambar ilustrasi dari kompresi lossless di jelaskan pada gambar 2(Pangesti et al., 2020):


Gambar 2. *Lossless compression*

2.6 Kuantisasi citra

Citra digital terbentuk melalui pendekatan yang dinamakan kuantisasi. Kuantisasi merupakan prosedur yang digunakan untuk membuat suatu isyarat yang bersifat kontinu ke dalam bentuk diskret. Untuk memudahkan pemehaman ini, dapat dilihat pada gambar 2, Gambar 2a menyatakan isyarat analog menurut perjalanan waktu t, sedangkan gambar 2b menunjukan isyarat deskret. Pada isyarat digital, nilai intensitas citra dibuat diskret atau terkuantisasi dalam nilai bulat. Gambar 2a menunjukan contoh citra biner dua nilai instensitas berupa 0 (hitam) dan 1 (putih). Kemudian gambar tersebut ditumpangkan pada grid 8 x 8 seperti diperlihatkan di gambar 2b bagian gambar yang jatuh pada kotak kecil dengan luas lebih kecil dibandingkan warnaputih latar belakang, seluruh isi kotak dibuat putih. Sebaliknya jika mayoritas hitam, isi kotak seluruhnya dibuat hitam.



Gambar 2 Digitalisasi citra biner 8 x 8 piksel
untuk memperlihatkan bentuk piksel ideal.

Hasil pengubahan citra digital tampak Digambar 2c adapun gambar 2d memperlihatkan bilangan yang mewakili warna hitam dan putih. Dengan demikian citra digital akan lebih baik apabila ukuran piksel di perkecil atau jumlah piksel di perbanyak.n Bagaimana halnya kalua gambar mengandung unsur warna ( bukan hitam putih), prinsipnya sama saja, tetapi sebagai pengecualian, warna hitam diberikan tiga unsur dasar, yaitu merah ( R ), hijau ( H ), dan Biru (B ).(Sipan, 2023)

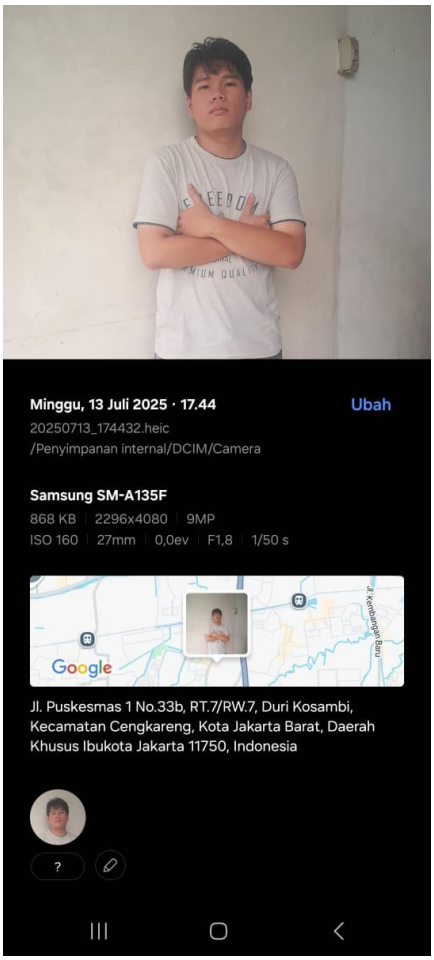
2.7 OpenCV

OpenCV merupakan pustaka *open-source* yang dikembangkan secara khusus untuk mendukung berbagai proses pengolahan citra digital dan *computer vision* [9]. Pustaka ini menyediakan beragam algoritma pemrosesan citra, seperti transformasi geometris, *filtering*, serta berbagai teknik lainnya yang telah dioptimalkan agar bekerja secara efisien dan cepat. Dibandingkan dengan pustaka lain seperti PIL atau SciPy dalam Python, OpenCV memiliki keunggulan dalam mendukung pemrosesan *real-time* serta dilengkapi modul yang dioptimalkan untuk berbagai jenis perangkat keras. Selain itu, OpenCV juga menawarkan fitur deteksi objek menggunakan metode *computer vision*, menjadikannya pilihan utama dalam pengembangan aplikasi berbasis visi komputer [10]. (Qisthiano & Pratiwi, 2025)

# BAB III

# HASIL

## 3.1 Deteksi Tepi




### 3.1.1 Canny Edge Detection

#### 3.1.1.1 Deskripsi Tujuan

Tujuan dari tahap ini adalah melakukan deteksi tepi pada citra menggunakan metode **Canny Edge Detection**. Deteksi tepi merupakan langkah awal dalam berbagai aplikasi pengolahan citra, seperti segmentasi objek, deteksi bentuk, dan pengenalan pola.

#### 3.1.1.2 Penjelasan Kode dan Teori

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

- cv2 adalah pustaka OpenCV yang digunakan untuk pemrosesan citra.
- matplotlib.pyplot digunakan untuk menampilkan gambar dalam format visual interaktif.

```
image_diri = "diri.jpg"
image = cv2.imread(image_diri)
```

- Membaca gambar dari file "diri.jpg" dan menyimpannya dalam variabel image dalam format BGR (Blue-Green-Red).

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

- Karena OpenCV membaca citra dalam format BGR, konversi ke RGB diperlukan agar hasil tampilannya benar ketika divisualisasikan dengan Matplotlib.

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

- Mengonversi citra berwarna ke grayscale. Deteksi tepi hanya memerlukan satu kanal (intensitas cahaya), bukan tiga kanal warna.

```
# Canny Edge Detection
edges = cv2.Canny(gray, threshold1=100, threshold2=200)
```

- Canny Edge Detection: Algoritma ini bekerja dalam beberapa tahap:
  1. *Noise Reduction* menggunakan Gaussian blur (otomatis dalam cv2.Canny).
  2. Menghitung gradien intensitas di tiap piksel.
  3. *Non-maximum suppression* untuk menipiskan garis tepi.
  4. *Hysteresis thresholding*: hanya garis yang intensitasnya melewati ambang ganda (100 dan 200) yang dipertahankan sebagai tepi.

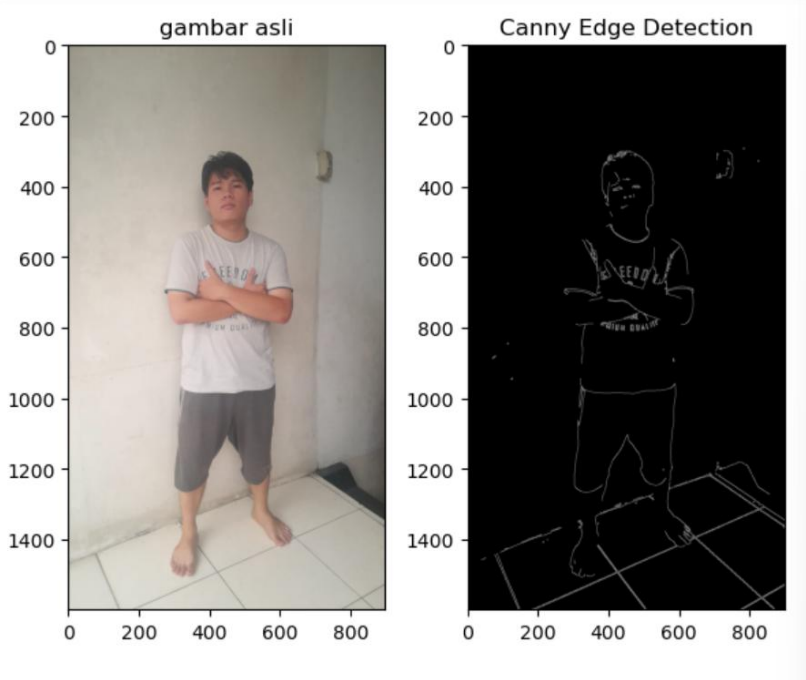
```
fig, axes = plt.subplots(1, 2, figsize=(6, 10))
```

- Membuat dua subplot sejajar (1 baris, 2 kolom) untuk menampilkan citra asli dan hasil deteksi tepi.

```
axes[0].imshow(image_rgb)
axes[0].set_title("gambar asli")
axes[0].axis('on')
axes[1].imshow(edges, cmap='gray')
axes[1].set_title("Canny Edge Detection")
axes[1].axis('on')

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

- Menampilkan citra asli di kiri dan hasil tepi (dalam skala abu-abu) di kanan. cmap='gray' digunakan karena hasil Canny adalah citra biner (hitam-putih)

3.1.1.3 Hasil dan analisis




- Tepi objek-objek seperti wajah, rambut, pakaian, atau background dengan kontras tinggi berhasil ditangkap.
- Bagian dalam objek yang memiliki gradasi atau perbedaan intensitas kecil mungkin tidak terdeteksi, ini wajar karena metode Canny sensitif terhadap kontras.

3.1.2 Contours Detection

3.1.2.1 Deskripsi Tujuan

Tujuan dari tahap ini adalah untuk mendeteksi dan menggambarkan **kontur** dari objek dalam citra, berdasarkan hasil dari Canny Edge Detection sebelumnya. Kontur digunakan untuk mendeskripsikan bentuk, posisi, dan ukuran objek.

3.1.2.2 Penjelasan kode dan teori

```
contours, hierarchy = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

- cv2.findContours() mencari kontur pada gambar biner edges.
- cv2.RETR_EXTERNAL: hanya mengambil kontur luar (tidak termasuk kontur dalam objek).
- cv2.CHAIN_APPROX_SIMPLE: menyederhanakan representasi kontur untuk efisiensi.

```
gambar_diri_contours = image.copy()
cv2.drawContours(gambar_diri_contours, contours, -1, (0, 255, 0), 2)
```

- Membuat salinan citra asli untuk digambari kontur.
- cv2.drawContours() menggambar semua kontur (-1) dengan warna hijau (0, 255, 0) dan ketebalan 2 piksel.

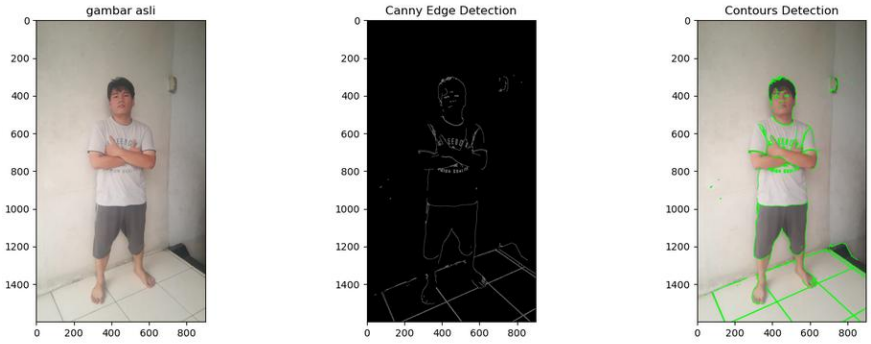
```
image_contours_rgb = cv2.cvtColor(gambar_diri_contours, cv2.COLOR_BGR2RGB)
```

- Konversi hasil kontur ke RGB untuk visualisasi.

```
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# Kolom 1: Original
axes[0].imshow(image_rgb)
axes[0].set_title("gambar asli")
axes[0].axis('on')

# Kolom 2: Canny Edge
axes[1].imshow(edges, cmap='gray')
axes[1].set_title("Canny Edge Detection")
axes[1].axis('on')

# Kolom 3: Contours
axes[2].imshow(image_contours_rgb)
axes[2].set_title("Contours Detection")
axes[2].axis('on')

# Rapikan tata letak
plt.tight_layout()
plt.show()
```

- Menampilkan tiga gambar dalam satu baris: gambar asli, hasil Canny, dan hasil kontur.

3.1.2.3 Hasil Dan analisis

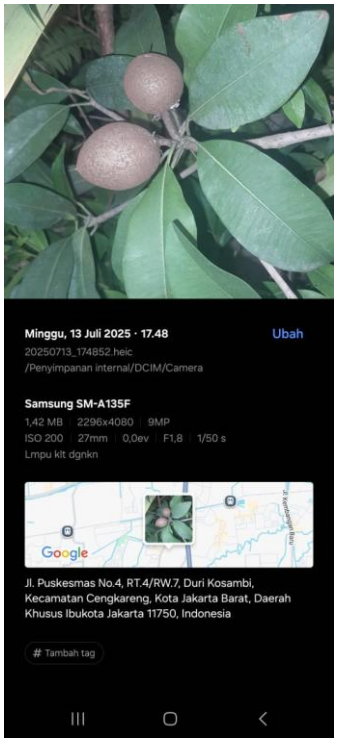



- Kontur yang ditampilkan dalam warna hijau mengikuti bentuk objek yang terlihat di hasil Canny.
- Karena menggunakan RETR_EXTERNAL, hanya bentuk luar yang digambar, menghindari cluttering dari kontur kecil di dalam objek.
- Kontur bisa kita gunakan untuk pengukuran dimensi objek, pemotongan objek, atau pelacakan dalam video.

3.1.3 Hubungan Teori dan implementasi Praktik

| **Teori** | **Implementasi & Hasil Praktik** |
|---|---|
| Canny Mendeteksi Tepi Berdasarkan Perubahan Intensitas. | Terlihat Garis Putih Di Tepi Objek Utama (Wajah, Rambut). |
| Kontur Merupakan Garis Tertutup Hasil Ekstraksi Bentuk. | Kontur Berhasil Mengikuti Bentuk Luar Objek. |
| Threshold Canny Mempengaruhi Hasil. | Ambang 100-200 Digunakan, Hasil Cukup Optimal. |
| Kontur Tergantung Pada Hasil Biner. | Jika Canny Gagal Mendeteksi, Kontur Pun Hilang. |

3.2 Segmentasi Buah dan Daun




3.2.1 Deskripsi Tujuan

Tujuan dari tahap ini adalah:

1. Menganalisis komponen warna citra buah menggunakan model warna HSV (Hue, Saturation, Value) melalui histogram.

2. Melakukan segmentasi daun dan non-daun (buah & batang) berdasarkan rentang nilai HSV tertentu yang mewakili warna hijau daun.

3.2.2 Penjelasan Kode

```
img = cv2.imread('buah.jpg')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

- Membaca citra buah.jpg dan melakukan dua konversi:
    - BGR → RGB untuk tampilan visual dengan Matplotlib.
    - BGR → HSV untuk keperluan analisis warna dan segmentasi.

Teori Model Warna HSV:

- Hue (H): menentukan jenis warna (merah, hijau, biru, dll), dalam derajat (0–180 di OpenCV).
- Saturation (S): intensitas warna (0 = abu-abu, 255 = sangat jenuh).
- Value (V): kecerahan warna (0 = hitam, 255 = sangat terang).

```
hue = img_hsv[:, :, 0]
sat = img_hsv[:, :, 1]
val = img_hsv[:, :, 2]
```

- Ekstraksi masing-masing kanal HSV dari citra: hue, saturation, dan value.

```
fig, axes = plt.subplots(1, 3, figsize=(18, 5))

axes[0].hist(hue.ravel(), bins=180, range=(0, 180), color='green')
axes[0].set_title("Histogram Hue (Warna)")
axes[0].set_xlabel("Hue")
axes[0].set_ylabel("Jumlah Piksel")

axes[1].hist(sat.ravel(), bins=256, range=(0, 256), color='blue')
axes[1].set_title("Histogram Saturation (Intensitas Warna)")
axes[1].set_xlabel("Saturation")

axes[2].hist(val.ravel(), bins=256, range=(0, 256), color='gray')
axes[2].set_title("Histogram Value (Kecerahan)")
axes[2].set_xlabel("Value")

plt.suptitle("Analisis Warna Citra dengan Histogram HSV")
plt.tight_layout()
plt.show()
```

- Membuat histogram dari masing-masing komponen HSV untuk menganalisis distribusi warnanya.

Interpretasi Histogram:

- Hue: menunjukkan warna dominan dalam gambar (misalnya, warna hijau untuk daun).
- Saturation: memperlihatkan seberapa kuat warna disaturasi.

- Value: menunjukkan seberapa terang atau gelap citra.

```
lower_green = np.array([35, 40, 40])
upper_green = np.array([85, 255, 255])
```

```
mask_daun = cv2.inRange(img_hsv, lower_green, upper_green)
```

- Masking daun: Membuat citra biner (putih-hitam) berdasarkan rentang HSV untuk warna hijau.
    - [35, 40, 40] s/d [85, 255, 255] adalah kisaran warna hijau pada daun.

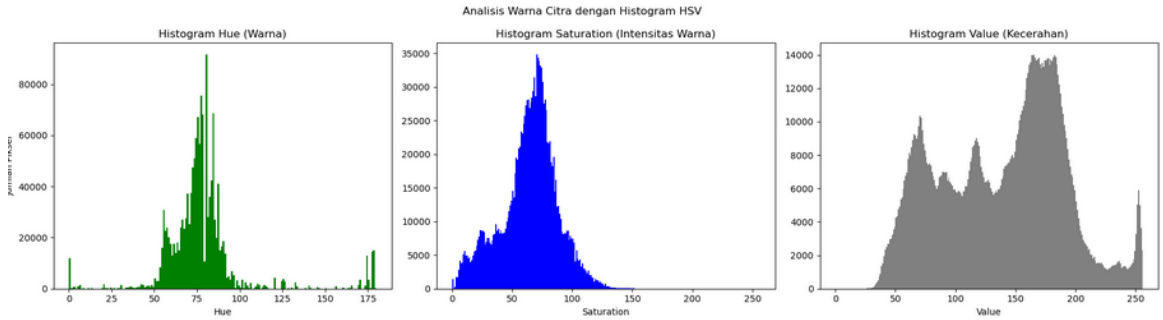
```
segmentasi_daun = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_daun)
```

- Segmentasi Daun: hanya bagian yang termasuk dalam masker hijau yang dipertahankan.

```
mask_non_daun = cv2.bitwise_not(mask_daun)
segmentasi_non_daun = cv2.bitwise_and(img_rgb, img_rgb, mask=mask_non_daun)
```

- Segmentasi Buah dan Batang: menggunakan kebalikan dari masker daun (bitwise_not), sehingga objek selain daun seperti buah atau batang dapat dipisahkan.

3.2.3 Analisis Histogram Untuk penetuan warna




**A. Histogram Hue (Warna)**

- Sumbu X: Nilai Hue (0–180), sumbu Y: jumlah piksel.
- Terlihat **puncak dominan pada kisaran 60–85**, menunjukkan bahwa warna hijau sangat dominan dalam citra ini.
- Ini sesuai dengan warna umum dari **daun** atau area hijau lain dalam gambar buah-buahan.

**Interpretasi**:
Nilai hue sekitar **60–85** identik dengan warna hijau segar. Ini menjadi dasar kuat untuk menetapkan batas segmentasi daun pada:

```
lower_green = np.array([35, 40, 40])
upper_green = np.array([85, 255, 255])
```

Sehingga, semua piksel dengan hue dalam rentang tersebut akan dikenali sebagai **daun**.

**B. Histogram Saturation (Intensitas Warna)**

- Puncak histogram berada pada rentang **50–110**, dengan jumlah piksel yang sangat tinggi.
- Hal ini menunjukkan bahwa mayoritas piksel dalam citra memiliki intensitas warna yang **cukup jenuh**, namun tidak ekstrem (bukan warna yang terlalu pekat).

**Interpretasi**:
Objek seperti buah atau daun yang alami biasanya memiliki saturasi sedang hingga tinggi, tidak terlalu rendah (abu-abu) dan tidak terlalu tinggi (neon/menyala). Ini mengonfirmasi bahwa objek dalam gambar adalah elemen alami seperti buah dan daun segar.

**C. Histogram Value (Kecerahan)**

Histogram Value menunjukkan distribusi kecerahan dengan tiga puncak:

- Sekitar **60–90**: bagian bayangan/gelap dari daun atau latar.
- Sekitar **120–160**: objek utama (buah/daun dengan pencahayaan sedang).
- Sekitar **180–240**: bagian gambar yang sangat terang (kemungkinan highlight pada buah atau background putih).

**Interpretasi**:
Tingginya distribusi kecerahan memperlihatkan bahwa gambar ini memiliki pencahayaan beragam, tetapi tetap berada pada rentang yang ideal untuk segmentasi. Citra tidak terlalu gelap atau terlalu terang secara keseluruhan.

**D. Validasi Terhadap Rentang Segmentasi Hijau**

```
lower_green = np.array([35, 40, 40])
upper_green = np.array([85, 255, 255])
```

- Rentang ini sangat tepat karena mencakup puncak utama di histogram Hue (hijau).
- Saturation ≥ 40 dan Value ≥ 40 memastikan hanya warna hijau yang cukup terang dan jenuh yang diambil, menghindari shadow atau noise.
- Dengan parameter ini, segmentasi daun akan efektif, tanpa banyak kesalahan deteksi pada latar belakang atau buah.

3.2.4 Hasil visualiasi dan analisis

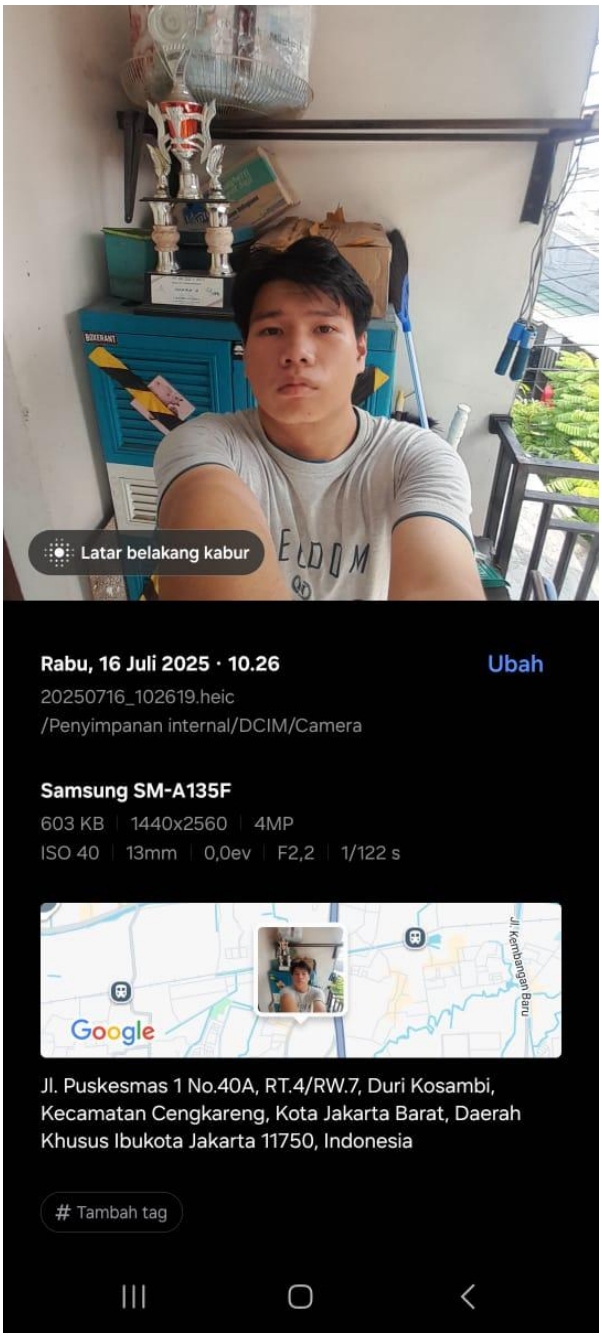

**Penjelasan Visual:**

- Gambar kiri atas adalah citra asli buah dan daun.
- Gambar kanan atas adalah masker non-daun, yaitu bagian yang *bukan hijau* (putih menunjukkan objek non-daun).
- Gambar kiri bawah adalah hasil segmentasi buah dan batang.
- Gambar kanan bawah adalah hasil segmentasi daun berdasarkan deteksi warna hijau.

3.2.5 Hubunga Teori Dan Implementasi Praktik

| Aspek Teori | Implementasi | **Hasil** |
|---|---|---|
| Warna Hijau memiliki rentang Hue sekitar 35–85 | np.array([35, 40, 40]) s/d np.array([85, 255, 255]) | Daun berhasil dipisahkan dari objek lain |
| Segmentasi warna menggunakan cv2.inRange() | Menyaring piksel sesuai warna tertentu | Daun tersegmentasi akurat |
| bitwise_and() mempertahankan piksel yang sesuai masker | Digunakan untuk ekstraksi daun dan non-daun | Memberi dua hasil berbeda: daun dan selain daun |

3.3 Kompresi dan Kuantisasi Citra



### 3.3.1 Kompresi Lossy (JPEG Quality 10%)

Melakukan kompresi citra menggunakan teknik lossy JPEG dengan kualitas rendah (quality=10). Tujuannya untuk melihat sejauh mana ukuran file bisa diperkecil dengan pengorbanan kualitas visual.

#### 3.3.1.1 Penjealasn kode dan teori

```
buffer_lossy = io.BytesIO()
original_img.save(buffer_lossy, format="JPEG", quality=10)
lossy_img = Image.open(buffer_lossy)
lossy_size_kb = len(buffer_lossy.getvalue()) / 1024
```

- quality=10: nilai kualitas sangat rendah, menghasilkan kompresi maksimal tetapi menurunkan kualitas gambar secara visual.
- **JPEG lossy compression** bekerja dengan menghapus detail halus yang tidak terlalu penting bagi persepsi manusia melalui:
    - Transformasi *Discrete Cosine Transform (DCT)*.
    - *Quantization* terhadap koefisien DCT.
    - *Entropy coding* (misalnya Huffman).
- BytesIO digunakan untuk menyimpan file dalam memori sehingga kita bisa mengukur ukuran file secara langsung tanpa menyimpan ke disk.

### 3.3.2 Kuantisasi Warna RGB (4 level Per channel)

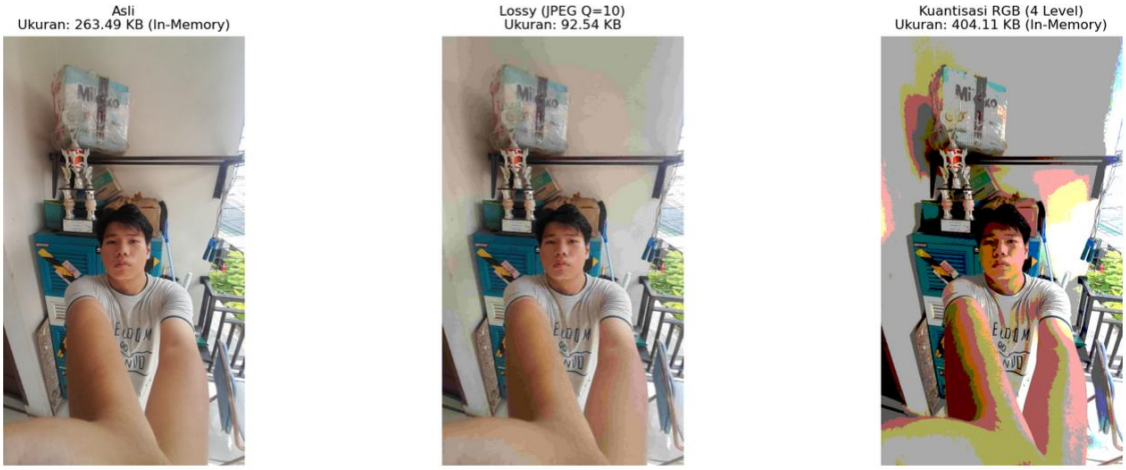
Melakukan kuantisasi warna pada citra dengan mengurangi jumlah level warna dari 256 level menjadi 4 level per channel (red, green, blue). Total kombinasi warna menjadi $4^3$ = 64 warna saja.

#### 3.3.2.1 Penjelasan Kode dan Teori

```
img_array = np.array(original_img)
quantized_array = ((img_array // 64) * 85).astype(np.uint8)
quantized_img = Image.fromarray(quantized_array)
```

- **Teori Kuantisasi**:
    - Warna asli pada tiap kanal berada dalam rentang 0–255.
    - Dibagi menjadi 4 level: 0–63, 64–127, 128–191, 192–255 → masing-masing direpresentasikan oleh titik tengah (0, 85, 170, 255).
    - // 64 membagi ke dalam 4 *bin*, * 85 mengatur ke representasi level tengah agar warna tidak terlalu gelap.
- Dengan kuantisasi ini, citra menjadi **posterized** (warna terlihat blok-blok), kehilangan gradasi warna halus.
- Membuat kembali gambar dari array numpy hasil kuantisasi warna.

## 3.3.3. Hasil Dan Analisis




### 3.3.3.1 Kompresi Lossy

- Terjadi penurunan ukuran file secara signifikan.
- Gambar menunjukkan penurunan kualitas visual, terutama pada bagian yang memiliki gradasi warna atau tekstur halus.
- Cocok untuk kasus di mana ukuran file lebih penting daripada detail visual, seperti thumbnail atau pratinjau gambar.

### 3.3.3.2 Kuantisasi RGB :

- Warna menjadi lebih kasar, banyak detail warna halus hilang.
- Ukuran file berkurang meskipun tidak seekstrem kompresi lossy JPEG, karena jumlah kombinasi warna jauh lebih sedikit.
- Cocok untuk aplikasi yang mengutamakan reduksi warna, misalnya kartunisasi gambar, klasifikasi warna, atau segmentasi warna kasar.

### 3.3.3.3 Perbandingan Hasil Kompresi

| **Teknik** | **Ukuran File (KB)** | **Kualitas Visual** | **Keuntungan** | **Kekurangan** |
|---|---|---|---|---|
| Asli | {original_size_kb:.2f} | Sempurna | Tidak ada penurunan kualitas | Ukuran besar |
| **Lossy (JPEG Q=10)** | {lossy_size_kb:.2f} | Turun drastis | Ukuran sangat kecil | Visual rusak |
| **Kuantisasi RGB** | {quantized_size_kb:.2f} | Turun sedang | Warna jadi 64 saja | Tidak cocok untuk foto alami |

# BAB IV
# PENUTUP

## 4.1 Kesimpulan

Berdasarkan landasan teori dan hasil praktikum yang telah dilakukan, dapat disimpulkan bahwa metode-metode dasar dalam pengolahan citra digital memiliki peran penting dalam berbagai proses analisis visual. Penerapan metode Canny Edge Detection terbukti mampu mengekstraksi tepi objek dalam citra dengan baik, berkat tahapan multistage yang mencakup reduksi noise, perhitungan gradien, dan thresholding ganda. Hasil deteksi tepi ini menjadi dasar bagi proses selanjutnya, yaitu deteksi kontur (Contours Detection), yang memungkinkan sistem untuk mengenali dan menggambarkan batas luar objek secara lebih presisi. Dengan menggambar kontur di atas citra asli, informasi bentuk dan posisi objek dapat divisualisasikan secara lebih jelas.

Selanjutnya, analisis warna menggunakan histogram HSV memberikan gambaran komprehensif mengenai distribusi komponen warna (Hue), intensitas warna (Saturation), dan kecerahan (Value) dalam citra. Pendekatan ini sangat membantu dalam proses segmentasi objek berdasarkan warna. Sebagai contoh, pada segmentasi daun dan buah, penggunaan rentang nilai hue antara 35 hingga 85 secara efektif berhasil memisahkan objek berwarna hijau (daun) dari objek lainnya seperti buah dan batang. Teknik segmentasi berbasis warna ini memperlihatkan bahwa pemilihan parameter yang tepat, yang didukung oleh data histogram, dapat meningkatkan akurasi deteksi objek secara signifikan.

Selain itu, dilakukan pula percobaan terhadap dua teknik kompresi citra, yaitu kompresi lossy JPEG dan kuantisasi warna RGB. Kompresi lossy dengan kualitas 10% mampu secara drastis mengurangi ukuran file citra, meskipun disertai penurunan kualitas visual yang cukup mencolok, terutama pada bagian dengan gradasi halus. Sementara itu, kuantisasi warna dengan mengurangi jumlah level warna RGB menjadi hanya empat per channel (sehingga total hanya 64 warna), menunjukkan penurunan kompleksitas warna secara signifikan namun tetap mempertahankan struktur bentuk citra secara keseluruhan. Meskipun ukuran file tidak sekecil hasil kompresi JPEG, pendekatan ini cocok digunakan pada aplikasi yang tidak memerlukan gradasi warna detail.

Secara keseluruhan, praktikum ini menunjukkan bahwa pemahaman dan penerapan konsep-konsep dasar seperti deteksi tepi, kontur, analisis HSV, segmentasi warna, dan kompresi citra merupakan landasan penting dalam pengembangan sistem computer vision. Implementasi teknik-teknik ini dapat disesuaikan dengan kebutuhan spesifik, baik dalam konteks efisiensi penyimpanan, akurasi segmentasi, maupun analisis visual lanjutan.

# DAFTAR PUSTAKA

Al Kautsar, M. T., Cahyono, E., & Jufra. (2025). *Jurnal Matematika , Komputasi dan Statistika PENERAPAN KONSEP FUNGSI CONTRAST STRETCHING TERHADAP PENAJAMAN KONTRAS CITRA DIGITAL Diterbitkan oleh Jurusan Matematika FMIPA UHO Jurnal Matematika , Komputasi dan Statistika Diterbitkan oleh Jurusan Matematik*. *5*(April), 857–867.

Enggari, S., Ramadhanu, A., & Marfalino, H. (2022). Peningkatan Digital Image Processing Dalam Mendeskripsikan Tumbuhan Jamur Dengan Segmentasi Warna, Deteksi Tepi Dan Kontur. *Jurnal Teknologi Dan Sistem Informasi Bisnis*, *4*(1), 70–75. https://doi.org/10.47233/jteksis.v4i1.358

Pangesti, W. E., Widagdo, G., Riana, D., & Hadianti, S. (2020). Implementasi Kompresi Citra Digital Dengan Membandingkan Metode Lossy Dan Lossless Compression Menggunakan Matlab. *Jurnal Khatulistiwa Informatika*, *8*(1), 53–58. https://doi.org/10.31294/jki.v8i1.7759

Qisthiano, M. R., & Pratiwi, A. O. (2025). Deteksi Tepi Pada Citra Objek Benda Menggunakan Algoritma Sobel Dan Prewitt Dengan Python. *Jurnal Informatika Dan Teknik Elektro Terapan*, *13*(2), 1115–1122. https://doi.org/10.23960/jitet.v13i2.6407

Sipan, M. (2023). *Mengenali Jenis Ayam Kampung Menggunakan Filter Gabor*. *14*(1), 6–13.