

UAS
PENGOLAHAN CITRA



NAMA : Dino Adianto Silalahi

NIM : 202331193

KELAS : A

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

NO.PC : 32

ASISTEN : 1. Clarenca Sweetdiva Pereira

2. Viana Salsabila Fairuz Syahla

3. Kashrina Masyid Azka

4. Sasikirana Ramadhanty Setiawan Putri

INSTITUT TEKNOLOGI PLN
TEKNIK INFORMATIKA
2024/2025

DAFTAR ISI

DAFTAR ISI.....	2
BAB I.....	3
PENDAHULUAN	3
1.1 Latar Belakang	3
1.2 Rumusan Masalah.....	3
1.3 Tujuan Masalah.....	3
1.4 Manfaat Masalah	4
BAB II.....	5
LANDASAN TEORI.....	5
2.1 Pengolahan Citra Digital	5
2.2 Filtering	6
2.3 Masking dan Segmentasi.....	6
2.4 Kompresi Citra	6
BAB III	8
HASIL.....	8
BAB IV	17
PENUTUP.....	17
4.1 KESIMPULAN	17
4.2 SARAN	17
DAFTAR PUSTAKA	18

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam era digital saat ini, pengolahan citra digital menjadi bidang penting yang digunakan dalam berbagai aplikasi, seperti pengawasan, medis, pertanian, hingga sistem keamanan. Salah satu aspek utama dalam pengolahan citra adalah kemampuan untuk memanipulasi citra guna meningkatkan kualitas atau mengurangi ukuran penyimpanan tanpa menghilangkan informasi penting. Praktikum ini berfokus pada tiga teknik utama: **filtering**, **masking**, dan **kompresi**.

Filtering digunakan untuk mereduksi noise dan meningkatkan kualitas visual dengan mempertahankan tepi (edges) penting pada citra. Masking atau segmentasi bertujuan menyorot area tertentu dari citra berdasarkan informasi warna, seperti HSV atau RGB thresholding, yang digunakan dalam deteksi objek. Sedangkan kompresi citra berfungsi untuk mengurangi ukuran file dengan tetap menjaga kualitas visual, seperti melalui kompresi JPEG (lossy) dan kuantisasi warna.

Melalui praktikum ini, mahasiswa tidak hanya belajar implementasi teknik dasar pengolahan citra, tetapi juga memahami efek visual, ukuran file, serta efektivitas teknik tersebut dalam konteks dunia nyata.

1.2 Rumusan Masalah

1. Bagaimana cara menerapkan filtering citra menggunakan metode median dan mean untuk mereduksi noise?
2. Bagaimana masking warna dapat digunakan untuk mengekstrak objek dari latar belakang?
3. Sejauh mana efektivitas kompresi JPEG dan kuantisasi warna dalam mengurangi ukuran file dengan mempertahankan kualitas citra?

1.3 Tujuan Masalah

1. Menerapkan dan membandingkan teknik filtering citra menggunakan metode **median** dan **mean**.
2. Mengimplementasikan proses **masking warna** untuk segmentasi objek pada citra.
3. Menerapkan teknik **kompresi lossy** dan **kuantisasi warna RGB** untuk mengevaluasi dampak terhadap kualitas dan ukuran citra.

1.4 Manfaat Masalah

1. Memahami konsep dasar pengolahan citra digital secara praktis dan teoretis.
2. Melatih kemampuan implementasi algoritma dasar citra menggunakan Python dan OpenCV.
3. Memberikan wawasan tentang bagaimana filtering, masking, dan kompresi memengaruhi kualitas visual dan efisiensi penyimpanan.

BAB II

LANDASAN TEORI

2.1 Pengolahan Citra Digital

Pengolahan citra digital merupakan bidang dalam ilmu komputer dan teknik elektro yang mempelajari teknik manipulasi gambar secara numerik dengan bantuan komputer. Gambar digital terdiri dari piksel-piksel (elemen gambar) yang memiliki nilai intensitas warna dan posisi dalam bidang dua dimensi. Tujuan utama dari pengolahan citra digital adalah untuk meningkatkan kualitas gambar atau mengekstrak informasi penting dari citra tersebut agar dapat digunakan dalam berbagai aplikasi seperti pengenalan wajah, deteksi objek, pengawasan visual, analisis medis, pemetaan citra satelit, dan masih banyak lagi.

Secara umum, pengolahan citra digital dibagi menjadi tiga tahap utama, yaitu:

1. Preprocessing (Pra-pemrosesan):

Tahapan ini bertujuan untuk meningkatkan kualitas citra sebelum dilakukan analisis lebih lanjut. Contohnya termasuk perbaikan kontras, pengurangan noise, filtering, dan normalisasi warna. Filtering (penyaringan) seperti median dan mean filter biasa digunakan di tahap ini.

2. Processing (Pemrosesan Tingkat Menengah):

Pada tahap ini dilakukan analisis struktur gambar untuk mengekstrak fitur-fitur penting. Contohnya seperti deteksi tepi, segmentasi, masking objek, atau kuantisasi warna. Dalam praktikum ini, masking warna pada ruang HSV digunakan untuk segmentasi objek (seperti buah atau daun).

3. Postprocessing (Pasca-pemrosesan):

Setelah fitur diperoleh, tahapan ini digunakan untuk interpretasi hasil atau aplikasi spesifik seperti kompresi, klasifikasi, pengenalan objek, atau pengambilan keputusan. Kompresi citra JPEG dan kuantisasi warna masuk ke dalam tahap ini untuk mengurangi ukuran file tanpa kehilangan terlalu banyak kualitas visual.

Keunggulan dari pengolahan citra digital adalah fleksibilitas dan efisiensinya dalam menangani berbagai jenis citra secara sistematis dan otomatis. Teknologi ini memanfaatkan kombinasi antara teori matematika (transformasi sinyal), algoritma komputer, dan teknik statistik.

Dalam konteks praktikum ini, pengolahan citra dilakukan menggunakan bahasa pemrograman Python dengan pustaka OpenCV dan NumPy, yang memungkinkan mahasiswa untuk langsung menerapkan konsep teori ke dalam implementasi nyata, seperti:

- **Filtering** (untuk membersihkan noise)
- **Masking/Segmentasi** (untuk menyorot bagian citra)
- **Kompresi** (untuk menghemat ruang penyimpanan)

Dengan demikian, pengolahan citra digital bukan hanya sebagai sarana visualisasi, namun juga menjadi fondasi penting dalam kecerdasan buatan, sistem penglihatan komputer (*computer vision*), dan pengenalan pola.

2.2 Filtering

Filtering adalah proses untuk menghilangkan noise atau gangguan dari citra. Dua jenis filtering yang umum digunakan:

- **Mean Filter (Rata-rata):** Menghaluskan citra dengan mengganti nilai piksel dengan rata-rata dari tetangganya.
- **Median Filter:** Mengganti nilai piksel dengan nilai median dari lingkungan sekitarnya, efektif untuk menghilangkan noise tipe "salt and pepper".

Menurut Shi et al. (2020), filtering juga dapat ditingkatkan dengan pendekatan deep learning berbasis **guided filtering** untuk mempertahankan struktur penting seperti tepi.

2.3 Masking dan Segmentasi

Masking adalah teknik untuk memilih bagian tertentu dari citra berdasarkan warna atau intensitas tertentu. Biasanya dilakukan pada ruang warna **HSV** untuk memisahkan objek dari latar belakang. Segmentasi bertujuan untuk membagi citra menjadi area-area yang lebih bermakna.

Menurut Bello et al. (2021), segmentasi instan seperti **Mask R-CNN** dapat digunakan untuk memisahkan objek kompleks seperti ternak dalam citra, tetapi pendekatan sederhana seperti masking warna tetap relevan untuk kasus dasar.

2.4 Kompresi Citra

Kompresi citra bertujuan mengurangi ukuran file dengan atau tanpa mengurangi kualitas visual.

- **Lossy Compression** seperti JPEG menghilangkan informasi visual yang dianggap tidak penting.

- **Color Quantization** mengurangi jumlah warna unik pada citra. Misalnya, mengubah 256 level RGB menjadi hanya 4 level per channel (sehingga total hanya 64 kombinasi warna).

Menurut Hammad et al. (2024), teknik seperti **quantization**, **DCT**, dan **LSB masking** juga digunakan dalam konteks steganografi dan kompresi citra untuk mencapai efisiensi dan keamanan data.

2.5 Evaluasi Kualitas Citra

Untuk mengevaluasi kualitas setelah filtering atau kompresi, digunakan parameter seperti:

- **PSNR (Peak Signal to Noise Ratio)**: Mengukur perbedaan antara citra asli dan hasil.
- **Ukuran file**: Menggambarkan efisiensi penyimpanan.
- **Visual Inspection**: Melihat perbedaan nyata secara kasat mata.

BAB III

HASIL

```
jupyter UAS_PCD_Kelas A_202331193_Dino Adianto_Silalahi_Deteksi Daun Last Checkpoint: 3 hours ago
File Edit View Run Kernel Settings Help Trusted
JupyterLab Python 3 (ipykernel)
```

Import Library

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

Baca gambar dari file 'buah.jpg'

```
[2]: buah = cv2.imread('buah.jpg')
```

Konversi BGR ke RGB

```
[3]: buah_rgb = cv2.cvtColor(buah, cv2.COLOR_BGR2RGB)
```

Konversi RGB ke HSV

```
[4]: buah_hsv = cv2.cvtColor(buah_rgb, cv2.COLOR_RGB2HSV)
```

Buat mask buah

```
[5]: lower_red1 = np.array([0, 70, 50])
upper_red1 = np.array([10, 255, 255])

lower_red2 = np.array([160, 70, 50])
upper_red2 = np.array([180, 255, 255])

mask_buah1 = cv2.inRange(buah_hsv, lower_red1, upper_red1)
mask_buah2 = cv2.inRange(buah_hsv, lower_red2, upper_red2)
mask_buah = cv2.bitwise_or(mask_buah1, mask_buah2)
```

Buat Mask daun

```
[6]: lower_daun = np.array([35, 40, 40])
upper_daun = np.array([85, 255, 255])
mask_daun = cv2.inRange(buah_hsv, lower_daun, upper_daun)
```

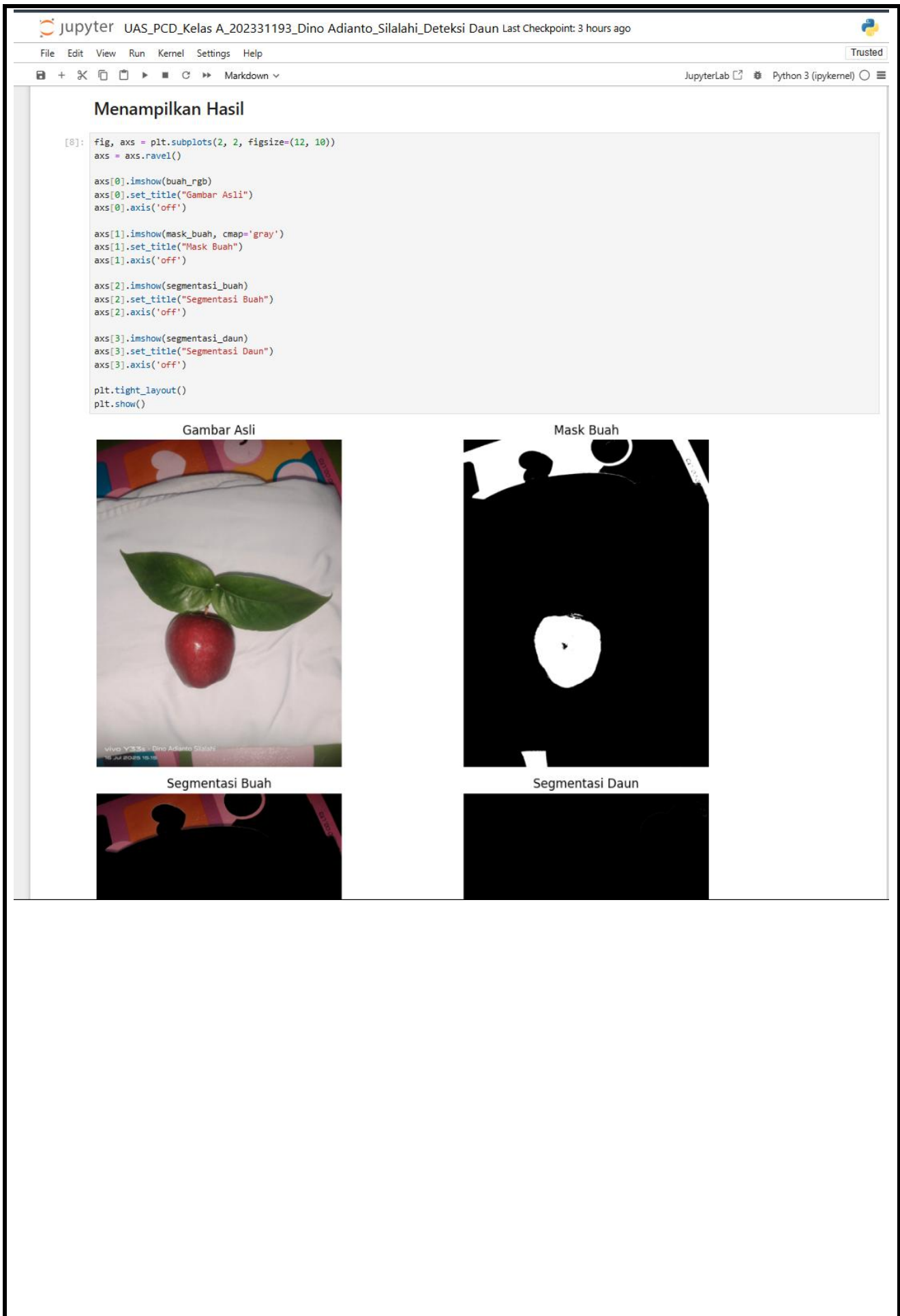
Segmentasi buah dan daun

```
[7]: segmentasi_buah = cv2.bitwise_and(buah_rgb, buah_rgb, mask=mask_buah)
segmentasi_daun = cv2.bitwise_and(buah_rgb, buah_rgb, mask=mask_daun)
```

Menampilkan Hasil

```
[8]: fig, axes = plt.subplots(2, 2, figsize=(12, 10))
axes = axes.ravel()

axes[0].imshow(buah_rgb)
axes[0].set_title("Gambar Asli")
axes[0].axis('off')
```

UAS_PCD_Kelas A_202331193_Dino Adianto_Silalahi_Deteksi Daun Last Checkpoint: 3 hours ago

File Edit View Run Kernel Settings Help

Trusted

JupyterLab
Python 3 (ipykernel)

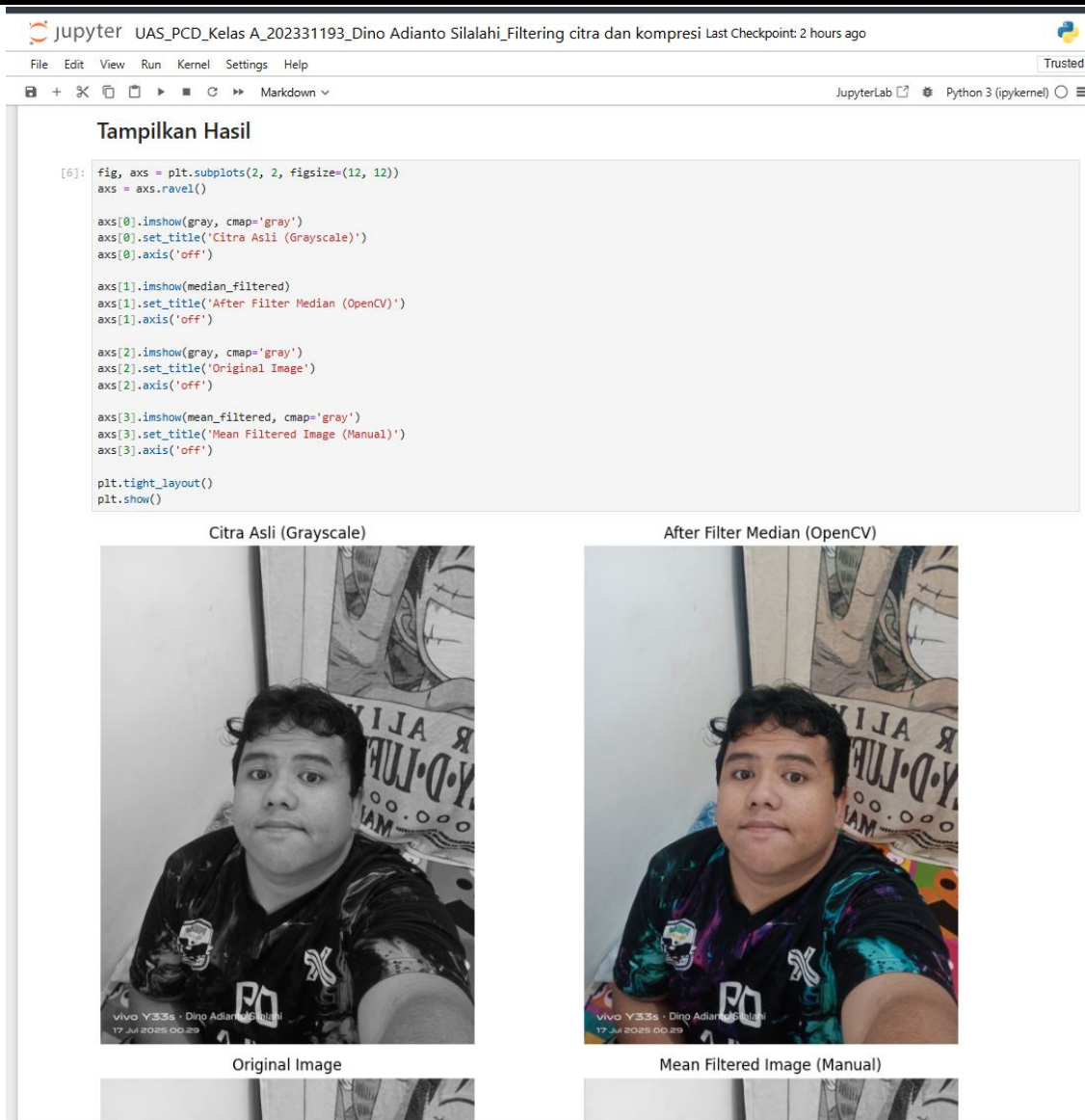
Gambar Asli

Mask Buah

Segmentasi Buah

Segmentasi Daun

[]:



Citra Asli (Grayscale)



After Filter Median (OpenCV)



Original Image



Mean Filtered Image (Manual)



Jupyter UAS_PCD_Kelas A_202331193_Dino Adianto Silalahi_Filtering citra dan kompresi Last Checkpoint: 2 hours ago

File Edit View Run Kernel Settings Help Trusted

import library os

```
[7]: import os
from io import BytesIO
from PIL import Image
```

KOMPRESI LOSSY JPEG QUALITY

```
[8]: jpeg_encoded = cv2.imencode('.jpg', dino, [int(cv2.IMWRITE_JPEG_QUALITY), 10])
jpeg_decoded = cv2.imdecode(np.frombuffer(jpeg_encoded, np.uint8), cv2.IMREAD_COLOR)
jpeg_rgb = cv2.cvtColor(jpeg_decoded, cv2.COLOR_BGR2RGB)
jpeg_size_kb = len(jpeg_encoded) / 1024
```

KUANTISASI RGB KE 4 LEVEL

```
[9]: quantized_rgb = (dino_rgb // 64) * 85
buffer = BytesIO()
Image.fromarray(quantized_rgb).save(buffer, format='PNG')
quantized_size_kb = len(buffer.getvalue()) / 1024
```

UKURAN CITRA ASLI

```
[10]: original_buffer = BytesIO()
Image.fromarray(dino_rgb).save(original_buffer, format='PNG')
original_size_kb = len(original_buffer.getvalue()) / 1024
```

TAMPILKAN HASIL KOMPRESI & KUANTISASI

```
[11]: fig, axs = plt.subplots(1, 3, figsize=(18, 6))
axs = axs.ravel()


axs[0].imshow(dino_rgb)
axs[0].set_title(f'Citra Asli (RGB)\nUkuran: {original_size_kb:.2f} KB (In-Memory)')
axs[0].axis('off')

axs[1].imshow(jpeg_rgb)
axs[1].set_title(f'Lossy JPEG Q=10\nUkuran: {jpeg_size_kb:.2f} KB')
axs[1].axis('off')


axs[2].imshow(quantized_rgb)
axs[2].set_title(f'Kuantisasi RGB (4 Level)\nUkuran: {quantized_size_kb:.2f} KB (In-Memory)')
axs[2].axis('off')

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


Citra Asli (RGB)
Ukuran: 10387.32 KB (In-Memory)



Lossy JPEG Q=10
Ukuran: 328.46 KB



Kuantisasi RGB (4 Level)
Ukuran: 1130.57 KB (In-Memory)



Penjelasan tiap tiap code

A. Masking

1. cv2.imread() dan cv2.cvtColor()

- cv2.imread() membaca gambar dari file.

- `cv2.cvtColor()` mengubah warna dari BGR ke RGB (untuk matplotlib) dan ke HSV (untuk segmentasi warna berdasarkan Hue).

2. Masking Buah Merah

- Warna merah di HSV terletak di dua rentang ([0-10] dan [160-180]) karena warna merah "melingkar" pada hue.
- `cv2.inRange()` membuat gambar biner (hitam-putih) berdasarkan kisaran warna.

3. Masking Daun Hijau

- Daun biasanya berada di hue [35-85] dengan saturasi dan brightness yang cukup tinggi.
- Sama juga, digunakan `cv2.inRange()` untuk buat mask.

4. Segmentasi

- `cv2.bitwise_and()` digunakan untuk mengambil hanya bagian gambar asli yang sesuai dengan mask yang kita buat.

5. Menampilkan Hasil dengan `axs.ravel()`

- Gunanya agar layout subplot mudah diatur dan konsisten.
- Cocok untuk menampilkan hasil asli, mask, dan segmentasi secara rapi dalam 1 tampilan.

Filtering dan Kompresi

Filtering

1. `cv2.cvtColor()` dan Grayscale

- `cv2.cvtColor()` digunakan untuk mengubah gambar dari BGR ke grayscale.
- Grayscale mempermudah proses filtering karena hanya satu channel warna.

2. Mean Filtering (Manual)

- Mengambil rata-rata nilai piksel dari tetangga 3x3 di sekitar tiap piksel.
- Loop dilakukan secara manual dan hasil disimpan di matriks baru.

3. Median Filtering (OpenCV)

- `cv2.medianBlur()` mengganti setiap piksel dengan nilai tengah dari lingkungan 5x5.
- Efektif untuk menghilangkan noise seperti bintik hitam/putih (salt and pepper).

4. `copy()`, `shape`, dan `np.zeros()`

- `.copy()` membuat salinan gambar agar aslinya tidak berubah.
- `.shape` mengambil ukuran gambar.
- `np.zeros()` membuat matriks kosong untuk menyimpan hasil filter.

KOMPRESI DAN KUANTISASI

5. Kompresi JPEG (Lossy)

- `cv2.imencode()` mengubah gambar ke format JPEG dengan kualitas rendah (10%).
- `cv2.imdecode()` digunakan untuk membuka kembali hasil kompresi agar bisa ditampilkan.
- Kompresi mengurangi ukuran file dengan sedikit penurunan kualitas visual.

6. Kuantisasi Warna (4 Level RGB)

- Dibuat dengan `dino_rgb // 64 * 85`, membatasi nilai setiap channel ke 4 level: 0, 85, 170, 255.
- Digunakan untuk mengurangi jumlah warna dan menyederhanakan gambar.

7. Menghitung Ukuran File (In-Memory)

- `BytesIO()` digunakan untuk menyimpan citra dalam memori, bukan file.
- `len(buffer.getvalue()) / 1024` menghitung ukuran file dalam KB.

8. Menampilkan Hasil dengan `plt.subplots()`

- `plt.subplots()` membuat banyak area gambar sekaligus.
- `axs.ravel()` mempermudah pengaturan posisi subplot.
- Setiap gambar diberi judul dan axis disembunyikan agar fokus ke visual.

BAB IV

PENUTUP

4.1 KESIMPULAN

Berdasarkan hasil praktikum yang telah dilakukan, dapat disimpulkan bahwa:

1. **Filtering citra** dengan metode **mean** dan **median** efektif digunakan untuk mengurangi noise pada citra grayscale. Mean filter memberikan hasil yang lebih halus namun cenderung mengaburkan tepi, sedangkan median filter lebih baik dalam mempertahankan detail dan menghilangkan noise ekstrem seperti salt and pepper.
2. **Masking warna** pada ruang warna HSV berhasil digunakan untuk **segmentasi objek** berdasarkan rentang warna tertentu, seperti buah merah dan daun hijau. Teknik ini sederhana namun cukup efektif untuk mendeteksi objek dengan warna dominan.
3. **Kompresi lossy JPEG** dengan kualitas rendah (quality=10) menunjukkan penurunan ukuran file secara signifikan, namun disertai dengan sedikit penurunan kualitas visual. Hal ini membuktikan adanya trade-off antara ukuran dan kualitas.
4. **Kuantisasi warna RGB** ke 4 level per channel mampu menyederhanakan citra dengan pengurangan jumlah warna, sekaligus menghemat ukuran file tanpa menghilangkan struktur utama gambar. Citra hasil kuantisasi terlihat lebih posterized (kartun), namun masih dapat dikenali dengan baik.
5. Penggunaan library **OpenCV**, **NumPy**, dan **Matplotlib** sangat membantu dalam mengimplementasikan teknik-teknik dasar pengolahan citra digital secara praktis dan visual.

4.2 SARAN

1. Untuk segmentasi objek dengan warna yang kompleks atau mirip dengan latar belakang, sebaiknya dikombinasikan dengan metode lain seperti deteksi tepi atau model segmentasi berbasis deep learning (contohnya: Mask R-CNN).
2. Dalam filtering citra, penggunaan kernel yang lebih besar (misalnya 5x5 atau 7x7) dapat dicoba untuk kasus noise yang lebih berat, namun perlu dipertimbangkan pengaruhnya terhadap hilangnya detail gambar.
3. Evaluasi kualitas citra sebaiknya tidak hanya dilakukan secara visual, tetapi juga secara kuantitatif dengan menggunakan metrik seperti **PSNR** dan **SSIM** agar hasil lebih objektif.
4. Praktikum ini dapat dikembangkan lebih lanjut dengan membandingkan teknik kompresi dan segmentasi lain, atau dengan membuat pipeline otomatis pengolahan citra dari filtering hingga klasifikasi objek.

DAFTAR PUSTAKA

1. Bello, A. I., Fashoto, S. G., Afolabi, O. O., & Udoh, S. J. (2021). Contour extraction of individual cattle from an image using enhanced Mask R-CNN instance segmentation method. *Indonesian Journal of Electrical Engineering and Computer Science*, 23(1), 137–145. <https://doi.org/10.11591/ijeecs.v23.i1.pp137-145>
2. Hammad, O. M., Smaoui, I., Fakhfakh, A., & Hashim, M. M. (2024). Recent advances in digital image masking techniques: Future challenges and trends—A review. *SHIFRA*, 2024(008), 67–73. <https://doi.org/10.70470/SHIFRA/2024/008>
3. Lindner, M., & Stricker, D. (2021). Camera calibration using planar calibration patterns under circular motion. *arXiv preprint arXiv:2106.01428*. <https://arxiv.org/abs/2106.01428>