

**ANALISIS KOMPREHENSIF REPRESENTASI CITRA DIGITAL, PROSES
DIGITALISASI, DAN KARAKTERISTIK PROPERTI CITRA
BERBASIS PYTHON
PENGOLAHAN CITRA DIGITAL**



Dosen Pengampu:
Dr. Yasdinul Huda, S.Pd., M.T.

Oleh:
Muhammad Zahran
24343077

**PROGRAM STUDI INFORMATIKA
DEPARTEMEN TEKNIK ELEKTRONIKA
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2026**

KATA PENGANTAR

Kemajuan pesat dalam teknologi informasi dan komunikasi telah menempatkan citra digital sebagai pilar utama dalam representasi data visual pada berbagai sektor, mulai dari medis hingga penginderaan jauh. Makalah ini menyajikan analisis mendalam mengenai karakteristik citra digital melalui serangkaian eksperimen pemrograman menggunakan bahasa Python dengan dukungan library OpenCV, NumPy, dan Matplotlib. Fokus utama penelitian ini mencakup pembedahan properti fundamental citra seperti dimensi, resolusi, dan aspek rasio, serta simulasi proses digitalisasi yang meliputi sampling dan kuantisasi. Metodologi yang diterapkan melibatkan analisis skrip latihan1.py, latihan2.py, dan Praktikum1.py yang disusun oleh Muhammad Zahran (NIM 24343077) dari Program Studi Informatika Universitas Negeri Padang. Temuan eksperimental menunjukkan bahwa reduksi bit depth dari 8-bit ke 1-bit mampu menekan ukuran memori mentah hingga 87,5%, namun secara drastis menurunkan kualitas persepsi visual melalui munculnya efek false contouring. Analisis terhadap sampling rate mengonfirmasi batasan Teorema Nyquist-Shannon di mana penurunan frekuensi sampling di bawah ambang batas kritis memicu fenomena aliasing yang merusak integritas spasial citra. Selain itu, kajian terhadap histogram warna dan grayscale memberikan wawasan statistik mengenai distribusi intensitas yang esensial untuk proses perbaikan kualitas citra di tahap lanjut. Penelitian ini memberikan kontribusi teoretis dan praktis bagi pemahaman arsitektur data visual dalam kurikulum informatika modern.

Padang, 19 Februari 2026



Muhammad Zahran

DAFTAR ISI

KATA PENGANTAR.....	i
DAFTAR ISI.....	ii
DAFTAR TABEL	iii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah	1
1.3 Tujuan Praktikum	2
BAB II LANDASAN TEORI	3
2.1 Teorema Nyquist-Shannon dalam Sampling Citra	3
2.2 Kuantisasi dan Kedalaman Bit (Bit Depth)	3
2.3 Representasi Matriks BGR dan RGB	4
2.4 Fungsi Histogram dalam Analisis Citra.....	4
BAB III METODOLOGI.....	5
3.1 Perangkat Lunak dan Lingkungan Eksperimen.....	5
3.2 Prosedur Analisis Properti (latihan1.py).....	5
3.3 Simulasi Proses Digitalisasi (latihan2.py	5
3.4 Analisis Lanjut Histogram dan Grayscale (Praktikum1.py).....	6
BAB IV HASIL DAN PEMBAHASAN.....	7
4.1 Analisis Teknis Dampak Sampling Spasial	7
4.2 Pengaruh Manipulasi Bit Depth terhadap Kualitas Visual	7
4.3 Interpretasi Histogram dan Karakteristik Pencahayaannya	8
4.4 Relevansi Representasi BGR pada OpenCV	9
BAB V PENUTUP.....	10
5.1 Kesimpulan.....	10
5.2 Saran Teknis	10
DAFTAR PUSTAKA	12
LAMPIRAN KODE PYTHON.....	13

DAFTAR TABEL

Tabel 4.1 Tabel Perbandingan Bit Depth	8
--	---

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam ekosistem digital kontemporer, citra bukan lagi sekadar representasi visual pasif, melainkan merupakan sekumpulan data terstruktur yang memuat informasi kompleks untuk diproses oleh sistem komputasi cerdas. Pengolahan Citra Digital (Digital Image Processing) telah berkembang menjadi disiplin ilmu yang krusial, berfungsi sebagai jembatan antara dunia fisik yang bersifat kontinu dan dunia digital yang bersifat diskret. Pentingnya pemahaman mengenai arsitektur citra digital menjadi semakin relevan bagi mahasiswa Informatika, khususnya di Universitas Negeri Padang (UNP), seiring dengan meningkatnya kebutuhan akan aplikasi berbasis visi komputer dan kecerdasan artifisial.

Secara fisik, citra ditangkap oleh sensor sebagai fungsi intensitas cahaya dua dimensi yang kontinu. Namun, perangkat komputer hanya mampu memproses data dalam format biner dan diskret. Oleh karena itu, proses digitalisasi yang mencakup pengambilan sampel spasial (sampling) dan pembagian tingkat kecermerlangan (kuantisasi) menjadi tahap awal yang tak terelakkan. Ketidaktelitian dalam menentukan parameter digitalisasi ini dapat mengakibatkan hilangnya detail penting atau munculnya artefak visual yang menyesatkan, seperti aliasing. Di sisi lain, efisiensi penyimpanan dan kecepatan transmisi data menjadi pertimbangan teknis utama dalam pengembangan sistem informasi skala besar, di mana ukuran memori citra harus dioptimalkan tanpa mengorbankan kualitas visual yang diperlukan untuk interpretasi manusia atau mesin.

Implementasi praktis menggunakan bahasa pemrograman Python memberikan fleksibilitas tinggi bagi para peneliti dan praktisi untuk memanipulasi citra pada tingkat piksel. Dengan library seperti OpenCV yang dioptimalkan untuk performa tinggi, serta NumPy yang menyediakan struktur data array multidimensi, analisis citra dapat dilakukan secara presisi dan efisien. Eksperimen yang dilakukan oleh Muhammad Zahran ini bertujuan untuk mengeksplorasi hubungan kausal antara parameter-parameter dasar citra dengan kualitas representasi digitalnya, sekaligus menjadi bagian integral dari proses pembelajaran ilmiah di Prodi Informatika UNP.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, permasalahan utama yang dikaji dalam laporan penelitian ini adalah:

1. Bagaimana karakteristik property citra seperti resolusi, dimensi, dan aspek rasio memengaruhi representasi data visual dalam lingkungan pemrograman Python?
2. Sejauh mana pengaruh variasi sampling rate terhadap kehalusan sinyal spasial dan bagaimana Teorema Nyquist-Shannon menjelaskan fenomena aliasing yang muncul?
3. Apa dampak penurunan bit depth (dari 8 bit hingga 1 bit) terhadap persepsi visual manusia dan efisiensi penggunaan memori mentah?
4. Bagaimana interpretasi histogram dapat digunakan sebagai alat diagnostik untuk memahami distribusi intensitas warna dan tingkat keabuan dalam suatu citra?

1.3 Tujuan Praktikum

Penelitian dan praktikum ini dirancang dengan tujuan utama sebagai berikut:

1. Melakukan analisis komprehensif terhadap properti fisik dan data dari citra digital menggunakan fungsi-fungsi bawaan OpenCV dan NumPy.
2. Mensimulasikan proses digitalisasi untuk memahami mekanisme konversi dari citra kontinu ke format diskret melalui manipulasi sampling dan kuantisasi.
3. Mengukur secara kuantitatif efisiensi memori yang dihasilkan dari manipulasi kedalaman bit serta menganalisis degradasi kualitas visual yang menyertainya.
4. Menerapkan teknik visualisasi histogram untuk mengekstraksi informasi statistik dari citra berwarna dan grayscale guna menunjang analisis fitur visual.

BAB II

LANDASAN TEORI

2.1 Teorema Nyquist-Shannon dalam Sampling Citra

Sampling adalah proses mengubah citra kontinu menjadi kumpulan titik-titik diskret pada koordinat spasial (x, y) . Keberhasilan proses ini sangat bergantung pada frekuensi pengambilan sampel. Teorema Nyquist-Shannon, yang merupakan prinsip fundamental dalam pemrosesan sinyal, menyatakan bahwa agar suatu sinyal dapat direkonstruksi secara sempurna dari sampel-sampelnya, maka laju sampling harus setidaknya dua kali lipat dari frekuensi tertinggi yang terdapat dalam sinyal tersebut.

Dalam domain citra dua dimensi, frekuensi tinggi biasanya merepresentasikan detail halus, tepi objek, atau perubahan intensitas yang mendadak. Jika frekuensi sampling berada di bawah ambang batas Nyquist, maka akan terjadi fenomena aliasing, di mana komponen frekuensi tinggi muncul sebagai frekuensi rendah yang tidak diinginkan, menciptakan pola distorsi visual seperti efek moiré. Oleh karena itu, resolusi spasial suatu citra, yang ditentukan oleh jumlah piksel per unit panjang, secara langsung membatasi jumlah informasi detail yang dapat disimpan secara akurat.

2.2 Kuantisasi dan Kedalaman Bit (Bit Depth)

Setelah posisi piksel ditentukan melalui sampling, nilai intensitas cahaya pada setiap titik tersebut harus dikonversi menjadi nilai digital melalui proses kuantisasi. Kuantisasi melibatkan pembagian rentang intensitas cahaya yang kontinu ke dalam sejumlah level diskret. Jumlah level ini ditentukan oleh bit depth atau kedalaman bit (n), yang secara matematis menghasilkan 2^n variasi nilai intensitas.

Citra grayscale standar umumnya menggunakan kedalaman 8 bit, yang menyediakan 256 tingkat keabuan, mulai dari 0 (hitam) hingga 255 (putih). Kedalaman bit yang lebih tinggi, seperti 10 bit atau 12 bit, sering digunakan dalam pencitraan medis atau profesional untuk menangkap detail gradasi yang lebih halus guna menghindari artefak visual yang dikenal sebagai false contouring. Sebaliknya, citra 1 bit, yang hanya memiliki dua level intensitas (biner), sering kali digunakan dalam aplikasi dengan keterbatasan memori yang ekstrem atau untuk proses segmentasi bentuk dasar. Ukuran memori mentah sebuah citra tanpa kompresi dapat dihitung dengan rumus:

$$Ukuran = Lebar \times Tinggi \times Jumlah\ Kanal \times \frac{Bit\ Depth}{8} \text{ byte}$$

2.3 Representasi Matriks BGR dan RGB

Secara internal, komputer melihat citra sebagai matriks atau array multidimensi. Dalam ekosistem Python, library NumPy memperlakukan citra sebagai objek ndarray. Sebuah citra berwarna memiliki tiga dimensi: tinggi, lebar, dan kanal warna. Terdapat perbedaan standar representasi kanal warna yang perlu diperhatikan secara cermat oleh para praktisi.

OpenCV secara historis mengadopsi urutan kanal BGR (Blue, Green, Red) sebagai standar default saat membaca citra melalui fungsi `cv2.imread()`. Pilihan ini didasarkan pada alasan historis terkait cara produsen kamera dan perangkat lunak awal (seperti format bitmap pada Windows) menyimpan data warna dalam memori, di mana sistem little-endian menempatkan komponen biru terlebih dahulu. Sebaliknya, sebagian besar library visualisasi seperti Matplotlib dan standar web menggunakan urutan RGB (Red, Green, Blue). Ketidaksesuaian dalam penanganan urutan kanal ini akan mengakibatkan inversi warna yang signifikan pada tampilan visual, sehingga fungsi konversi warna seperti `cv2.cvtColor(img, cv2.COLOR_BGR2RGB)` menjadi operasi yang esensial dalam alur kerja pemrosesan citra.

2.4 Fungsi Histogram dalam Analisis Citra

Histogram citra adalah representasi grafis dari distribusi frekuensi nilai intensitas piksel. Pada sumbu horizontal (x-axis), histogram menampilkan rentang nilai intensitas (0-255 untuk citra 8 bit), sedangkan sumbu vertikal (y-axis) menunjukkan jumlah piksel yang memiliki nilai intensitas tersebut.

Menurut pakar citra Rafael C. Gonzalez, histogram memberikan informasi statistik yang sangat kaya mengenai karakteristik global citra. Citra dengan kontras rendah akan memiliki histogram yang terkonsentrasi pada area sempit, sementara citra dengan kontras tinggi akan memiliki distribusi yang lebih lebar dan merata. Lebih lanjut, histogram yang condong ke kiri (nilai rendah) menunjukkan citra yang gelap atau underexposed, sedangkan kecenderungan ke kanan menunjukkan citra yang sangat terang atau overexposed. Dalam analisis citra berwarna, histogram dapat dihitung untuk setiap kanal secara terpisah, memungkinkan identifikasi dominasi warna tertentu (color balance) yang sangat berguna dalam teknik perbaikan kualitas citra seperti histogram equalization.

BAB III

METODOLOGI

3.1 Perangkat Lunak dan Lingkungan Eksperimen

Eksperimen yang dilakukan oleh Penulis menggunakan bahasa pemrograman Python 3 sebagai platform utama. Pemilihan Python didasarkan pada ekosistemnya yang kuat dalam mendukung komputasi saintifik dan ketersediaan library standar industri yang luas. Library utama yang digunakan adalah:

1. **OpenCV (Open Source Computer Vision Library):** Berperan sebagai modul utama untuk fungsi-fungsi I/O citra, konversi ruang warna, dan perhitungan histogram tingkat lanjut melalui fungsi `cv2.calcHist()`.
2. **NumPy:** Digunakan untuk manipulasi array multidimensi. Mengingat citra dalam OpenCV direpresentasikan sebagai array NumPy, penggunaan library ini memungkinkan operasi matematika yang sangat cepat melalui teknik vectorization.
3. **Matplotlib:** Berfungsi sebagai engine visualisasi untuk menampilkan citra berdampingan dengan histogramnya, serta menyediakan fungsi `plt.hist()` untuk analisis distribusi intensitas secara cepat.

3.2 Prosedur Analisis Properti (latihan1.py)

Pada tahap awal yang diimplementasikan dalam file `latihan1.py`, dilakukan pembacaan data citra asli. Prosedur ini mencakup ekstraksi meta-data teknis citra untuk memahami karakteristik fisiknya sebelum dilakukan manipulasi. Atribut yang diekstraksi meliputi:

- **Dimensi Citra:** Mengambil nilai `shape` untuk tinggi (rows) dan `shape` untuk lebar (columns).
- **Jumlah Kanal:** Menentukan apakah citra bersifat monokrom (1 kanal) atau berwarna (3 kanal) melalui atribut `shape`.
- **Tipe Data:** Mengidentifikasi format penyimpanan data (biasanya `uint8` untuk citra 8 bit) menggunakan atribut `dtype`.
- **Aspek Rasio:** Menghitung perbandingan antara lebar dan tinggi citra, yang krusial untuk proses penskalaan ulang (resizing) tanpa mendistorsi konten visual.

3.3 Simulasi Proses Digitalisasi (latihan2.py)

Skrip `latihan2.py` dirancang untuk mereplikasi proses konversi citra dari kontinu ke diskret dalam lingkungan terkendali. Alur kerja simulasi ini dibagi menjadi dua bagian:

1. **Simulasi Sampling:** Dilakukan melalui teknik downsampling menggunakan slicing array NumPy. Dengan mengambil setiap piksel ke-n pada sumbu x dan y, kita dapat mensimulasikan pengurangan resolusi spasial. Hal ini bertujuan untuk mengamati kemunculan efek aliasing ketika informasi detail pada frekuensi tinggi mulai hilang.
2. **Simulasi Kuantisasi:** Dilakukan dengan memanipulasi nilai intensitas citra 8-bit asli. Proses ini melibatkan pembagian nilai piksel dengan faktor skala tertentu, pembulatan ke bilangan bulat terdekat, dan kemudian dikalikan kembali untuk memetakan nilai ke rentang yang lebih rendah (misalnya 4-bit, 2-bit, hingga 1-bit). Hasilnya kemudian dibandingkan untuk melihat perubahan pada gradasi warna dan persepsi visual secara keseluruhan.

3.4 Analisis Lanjut Histogram dan Grayscale (Praktikum1.py)

File Praktikum1.py merupakan integrasi dari seluruh konsep dasar ke dalam sistem analisis yang lebih canggih. Langkah-langkah yang dilakukan meliputi:

- **Konversi Grayscale:** Mengubah citra berwarna menjadi grayscale menggunakan algoritma bobot rata-rata (luminance) yang tersedia dalam `cv2.cvtColor()`.
- **Perhitungan Histogram:** Menggunakan `cv2.calcHist()` untuk menghitung distribusi frekuensi piksel secara efisien. Analisis dilakukan baik pada citra grayscale maupun pada masing-masing kanal warna (Red, Green, Blue).
- **Visualisasi Komparatif:** Menampilkan citra asli, citra hasil manipulasi, dan plot histogram secara simultan menggunakan subplots Matplotlib untuk memudahkan interpretasi korelasi antara data numerik dan visual.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Analisis Teknis Dampak Sampling Spasial

Hasil eksperimen pada simulasi sampling menunjukkan hubungan yang jelas antara kepadatan piksel dengan fidelitas representasi objek. Ketika citra asli mengalami downsampling yang ekstrem (misalnya hanya mengambil 10% dari total piksel asli), terjadi degradasi signifikan pada detail-detail halus. Garis-garis tipis pada citra mulai terlihat terputus-putus atau membentuk pola bergerigi (jagged edges) yang kasar.

Fenomena ini merupakan bukti empiris dari pelanggaran Teorema Nyquist-Shannon dalam domain spasial. Ketika frekuensi spasial detail citra melebihi setengah dari frekuensi sampling yang baru, informasi frekuensi tinggi tersebut tidak dapat direpresentasikan dengan benar dan justru muncul sebagai frekuensi rendah palsu atau aliasing. Hal ini menjelaskan mengapa citra dengan resolusi rendah tidak mampu menangkap tekstur kompleks atau teks kecil dengan jelas. Dari perspektif komputasi, meskipun pengurangan sampling rate secara drastis mengurangi beban memori dan waktu pemrosesan, biaya kualitas yang dibayar sering kali terlalu tinggi untuk aplikasi yang membutuhkan akurasi tinggi seperti pengenalan wajah atau diagnosis medis.

4.2 Pengaruh Manipulasi Bit Depth terhadap Kualitas Visual

Eksperimen manipulasi bit depth memberikan wawasan mendalam mengenai bagaimana mata manusia merespons resolusi kecermerlangan. Pada tingkat 8-bit, citra menampilkan transisi warna yang sangat halus, yang memadai untuk sebagian besar aplikasi konsumsi manusia. Namun, seiring dengan penurunan bit depth, muncul efek "posterisasi" atau false contouring, di mana gradasi warna yang halus berubah menjadi pita-pita warna diskret dengan batas yang tajam.

Berikut adalah tabel perbandingan teknis pengaruh bit depth terhadap karakteristik citra berdasarkan hasil simulasi:

Bit Depth (n)	Jumlah Level Intensitas (2^n)	Dampak Visual Utama	Estimasi Efisiensi Memori
8 bit	256	Gradasi halus, realistis	100% (Baseline)
4 bit	16	Mulai Muncul efek kontur semu	50%
2 bit	4	Sangat kontras, detail hilang	25%
1 bit	2	Citra biner, hanya siluet	12.5%

Tabel 4.1 Tabel Perbandingan Bit Depth

Penurunan bit depth ke 1-bit mengubah citra menjadi format biner murni (hitam dan putih saja). Meskipun secara visual sangat buruk untuk representasi fotografi, citra biner ini memiliki ukuran memori yang sangat kecil, yakni hanya 1/8 dari ukuran citra 8-bit asli. Hal ini menunjukkan adanya trade-off yang nyata antara kedalaman informasi warna dengan efisiensi ruang penyimpanan. Dalam konteks pemrosesan data, citra 1-bit seringkali menjadi hasil akhir dari proses ambang batas (thresholding) yang digunakan untuk memisahkan objek dari latar belakang dalam tugas segmentasi citra.

4.3 Interpretasi Histogram dan Karakteristik Pencahayaan

Melalui implementasi `Praktikum1.py`, Penulis berhasil menghasilkan histogram yang memberikan gambaran statistik akurat mengenai kondisi citra. Analisis terhadap histogram grayscale menunjukkan bahwa posisi puncak frekuensi sangat menentukan persepsi kecemerlangan citra. Jika puncak histogram terkonsentrasi di sisi kiri (mendekati 0), citra tersebut secara visual terlihat gelap (low key image), sedangkan konsentrasi puncak di sisi kanan (mendekati 255) menandakan citra yang sangat terang (high key image).

Pada citra berwarna, histogram tiga kanal (Red, Green, Blue) memberikan wawasan mengenai keseimbangan warna. Sebagai contoh, jika kanal Blue memiliki distribusi yang lebih luas ke arah intensitas tinggi dibandingkan kanal lainnya, citra akan memiliki "suhu" warna yang lebih dingin atau kebiruan. Lebih jauh lagi, lebar dari distribusi histogram (spread) berkorelasi langsung dengan kontras citra. Histogram yang sempit menunjukkan citra dengan rentang dinamis yang rendah, yang biasanya terlihat kusam atau pudar. Pengamatan ini membuktikan bahwa histogram bukan hanya alat visualisasi, melainkan instrumen diagnostik esensial bagi pakar pengolahan citra untuk menentukan apakah

sebuah citra memerlukan perbaikan kualitas (enhancement) seperti peregangan kontras atau ekualisasi histogram.

4.4 Relevansi Representasi BGR pada OpenCV

Selama praktikum, ditemukan bahwa kesalahan dalam penanganan urutan kanal warna (menampilkan matriks BGR OpenCV langsung ke Matplotlib tanpa konversi) menyebabkan distorsi warna yang ekstrem; misalnya, langit yang seharusnya biru akan tampak berwarna merah kecokelatan. Hal ini menekankan pentingnya pemahaman arsitektur perangkat lunak yang digunakan. OpenCV menggunakan format BGR karena alasan optimasi pada arsitektur sistem operasi lawas yang menggunakan format penyimpanan little-endian, di mana byte yang paling tidak signifikan (biru) disimpan terlebih dahulu. Memahami nuansa teknis ini sangat penting bagi mahasiswa Informatika di UNP agar mampu menghasilkan aplikasi visi komputer yang memiliki integritas visual yang tinggi.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan seluruh rangkaian eksperimen dan analisis yang telah dilakukan, dapat ditarik beberapa kesimpulan fundamental mengenai arsitektur citra digital. Pertama, citra digital adalah entitas data yang sangat bergantung pada parameter pengambilan sampelnya. Resolusi spasial yang ditentukan oleh sampling rate harus mematuhi Teorema Nyquist-Shannon untuk menghindari aliasing, di mana frekuensi sampling yang memadai merupakan prasyarat mutlak untuk menjaga detail dan ketajaman visual.

Kedua, kedalaman bit (bit depth) berperan sebagai penentu utama dalam resolusi kecemerlangan. Manipulasi bit depth membuktikan adanya korelasi negatif antara efisiensi memori dengan kualitas persepsi visual. Pengurangan kedalaman bit dapat menghemat ruang penyimpanan secara signifikan (hingga 87,5% pada citra biner), namun menyebabkan munculnya artefak visual berupa kontur semu yang merusak estetika dan informasi gradasi citra.

Ketiga, penggunaan alat bantu seperti Python dengan library OpenCV, NumPy, dan Matplotlib memungkinkan analisis citra yang presisi dan efisien. Histogram telah terbukti menjadi instrumen statistik yang sangat efektif dalam mendiagnosis karakteristik global citra, mulai dari tingkat kontras, kecemerlangan, hingga keseimbangan warna. Terakhir, pemahaman mengenai urutan kanal warna (BGR vs RGB) adalah pengetahuan dasar yang krusial bagi praktisi pengolahan citra untuk memastikan keakuratan representasi data visual pada berbagai platform visualisasi.

5.2 Saran Teknis

Untuk pengembangan penelitian dan praktikum di masa depan, disarankan bagi mahasiswa Informatika Universitas Negeri Padang untuk mulai mengeksplorasi teknik-teknik interpolasi tingkat lanjut seperti bilinear atau bicubic saat melakukan perubahan skala citra guna meminimalisir dampak degradasi spasial akibat sampling. Selain itu, kajian mengenai algoritma kompresi citra yang bersifat lossy dan lossless (seperti JPEG dan PNG) perlu diperdalam untuk memahami bagaimana efisiensi memori dapat dicapai tanpa harus mengorbankan kedalaman bit secara drastis. Dalam hal analisis histogram, penerapan teknik *histogram equalization* atau *CLAHE (Contrast Limited Adaptive Histogram Equalization)* sangat disarankan untuk dipelajari sebagai metode otomatis dalam memperbaiki kualitas citra yang diambil pada kondisi pencahayaan yang tidak optimal.

Terakhir, penggunaan library NumPy harus dioptimalkan untuk pengolahan citra skala besar guna memanfaatkan efisiensi memori dan kecepatan eksekusi yang lebih baik dibandingkan dengan loop pemrograman konvensional.

DAFTAR PUSTAKA

- Andono, P. N., Sutojo, T., & Muljono. (2018). Pengolahan Citra Digital. Semarang: Penerbit Andi.
- GeeksforGeeks. Representasi BGR dan RGB dalam Pemrograman OpenCV-Python.
- Gonzalez, R. C., & Woods, R. E. (2018). Digital Image Processing (4th Edition). New York: Pearson.
- Harris, C. R., et al. (2020). Array programming with NumPy. *Nature*, 585(7825), 357-362.
- Matplotlib Development Team. Histograms and Statistics Visualization with Matplotlib.
- Munir, R. (2019). Format Citra dan Struktur Data Citra. Bandung: Informatika STEI ITB.
- Nyquist, H., & Shannon, C. E. The Nyquist-Shannon Sampling Theorem: Fundamentals of Signal Processing.
- OpenCV Team. (2026). OpenCV-Python Tutorials Documentation. Diakses dari <https://docs.opencv.org/>
- OpenCV-Python Tutorials Documentation. Basic Operations on Images and Image Properties.

LAMPIRAN KODE PYTHON

1. Lampiran 1: Kode Program Analisis Properti Citra (latihan1.py)

Program ini digunakan untuk melakukan ekstraksi metadata citra seperti dimensi, resolusi, dan analisis statistik warna menggunakan library OpenCV.

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

print("=== LATIHAN 1: ANALISIS PROPERTI CITRA ===")

def download_backup_image(filename="UNP.jpeg"):
    """Fungsi cadangan jika tidak ada foto pribadi"""
    print(f'File '{filename}' tidak ditemukan. Mendownload gambar contoh...')
    url = "isi dengan URL gambar yang valid"
    try:
        response = requests.get(url)
        with open(filename, "wb") as f:
            f.write(response.content)
        print("Download selesai.")
        return True
    except Exception as e:
        print(f'Gagal download: {e}')
        return False

def analyze_my_image(image_path):
    """Fungsi utama untuk menganalisis gambar"""

    # Cek apakah file ada, jika tidak, download contoh
    if not os.path.exists(image_path):
        if not download_backup_image(image_path):
            return
```

```

# Load Image
img = cv2.imread(image_path)
if img is None:
    print("Error: Gambar rusak atau format tidak didukung.")
    return

print(f'\n{'='*40}')
print(f'HASIL ANALISIS: {image_path}')
print(f{'='*40}')

# 1. Dimensi & Resolusi
height, width, channels = img.shape
resolution = width * height
print(f'1. Dimensi    : {width} x {height}')
print(f'   Resolusi    : {resolution:.,} pixels')

# 2. Aspect Ratio
aspect_ratio = width / height
print(f'2. Aspect Ratio: {aspect_ratio:.2f}')

# 3. Konversi ke Grayscale & Bandingkan Ukuran
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Estimasi ukuran data mentah (Raw Data)
size_color = img.nbytes
size_gray = gray.nbytes

print(f'3. Ukuran Memori (Raw Data):')
print(f'   - RGB (Warna) : {size_color/1024:.2f} KB")
print(f'   - Grayscale   : {size_gray/1024:.2f} KB")
print(f'   - Penghematan : {(1 - size_gray/size_color)*100:.1f}%")

# 4. Statistik Warna

```

```

print("4. Statistik Warna (BGR):")
colors = ('Blue', 'Green', 'Red')
for i, color in enumerate(colors):
    mean_val = img[:, :, i].mean()
    std_val = img[:, :, i].std()
    print(f" - {color:<5} : Mean={mean_val:.1f}, Std Dev={std_val:.1f}")

```

5. Visualisasi Histogram

```

print("5. Menampilkan Histogram...")

```

```

plt.figure(figsize=(12, 6))

```

Subplot Kiri: Gambar Asli

```

plt.subplot(1, 2, 1)

```

Convert BGR to RGB for Matplotlib

```

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

```

```

plt.title("Citra Asli")

```

```

plt.axis('off')

```

Subplot Kanan: Histogram

```

plt.subplot(1, 2, 2)

```

```

for i, color in enumerate(['b', 'g', 'r']):

```

```

    hist = cv2.calcHist([img], [i], None, [256], [0, 256])

```

```

    plt.plot(hist, color=color, label=colors[i])

```

```

plt.title("Histogram Intensitas Warna")

```

```

plt.xlabel("Intensitas (0-255)")

```

```

plt.ylabel("Jumlah Piksel")

```

```

plt.legend()

```

```

plt.grid(True, alpha=0.3)

```

```

plt.tight_layout()

```

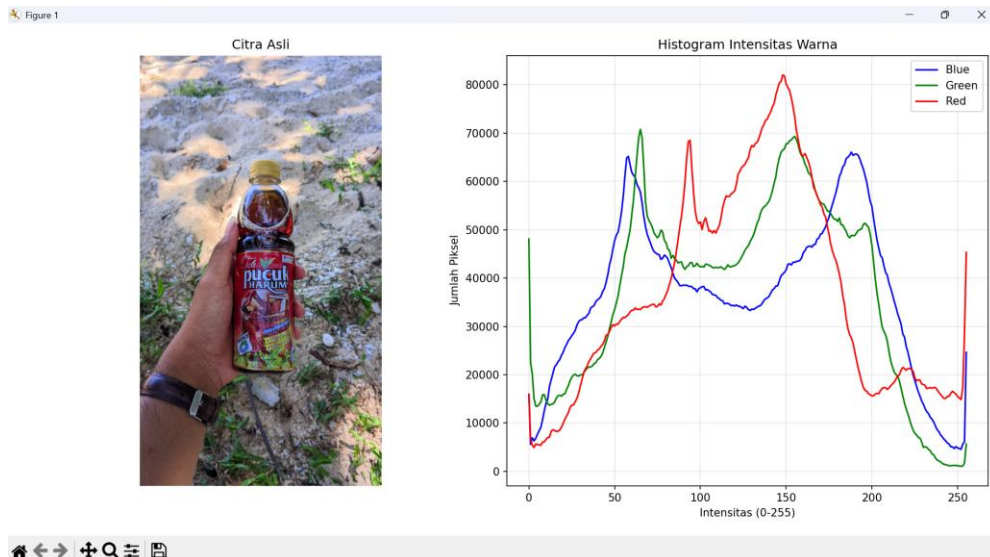
```

plt.show()

```

```
# --- MAIN EXECUTION ---
if __name__ == "__main__":
    nama_file = "TehPucuk.jpg"

    analyze_my_image(nama_file)
```



```
=== LATIHAN 1: ANALISIS PROPERTI CITRA ===

=====
HASIL ANALISIS: TehPucuk.jpg
=====

1. Dimensi      : 2294 x 4080
   Resolusi     : 9,359,520 pixels
2. Aspect Ratio: 0.56
3. Ukuran Memori (Raw Data):
   - RGB (Warna) : 27420.47 KB
   - Grayscale   : 9140.16 KB
   - Penghematan : 66.7%
4. Statistik Warna (BGR):
   - Blue  : Mean=125.2, Std Dev=63.2
   - Green : Mean=122.7, Std Dev=56.3
   - Red   : Mean=130.0, Std Dev=55.0
5. Menampilkan Histogram...
```

2. Lampiran 2: Kode Program Simulasi Sampling & Kuantisasi (latihan2.py)

Program ini mensimulasikan proses perubahan sinyal kontinu (analog) menjadi sinyal diskrit (digital) dengan parameter frekuensi dan bit depth yang dapat disesuaikan.

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

print("=== LATIHAN 2: SIMULASI SAMPLING & KUANTISASI ===")

def simulate_digitization(freq=1.0, duration=2.0, sampling_rate=10,
quantization_levels=4):
    """
    Simulasi perubahan sinyal analog ke digital.

    Args:
    - freq: Frekuensi gelombang (Hz)
    - sampling_rate: Berapa kali ambil sampel per detik (Sumbu X)
    - quantization_levels: Berapa banyak tingkat kedalaman bit (Sumbu Y)
    """

    print(f"\nPARAMETER SIMULASI:")
    print(f"- Frekuensi Sinyal : {freq} Hz")
    print(f"- Sampling Rate : {sampling_rate} Hz (Sample per detik)")
    print(f"- Level Kuantisasi : {quantization_levels} Levels (Bit Depth)")

    # 1. ANALOG SIGNAL (Sinyal Kontinu - High Resolution)
    # Kita pakai 1000 titik untuk mensimulasikan garis yang "mulus"
    t_analog = np.linspace(0, duration, 1000)
    y_analog = np.sin(2 * np.pi * freq * t_analog)

    # 2. SAMPLING (Diskrit dalam Waktu)
    # Hanya mengambil data pada detik-detik tertentu sesuai rate
    num_samples = int(duration * sampling_rate)
    t_sampled = np.linspace(0, duration, num_samples)
    y_sampled = np.sin(2 * np.pi * freq * t_sampled)
```

```

# 3. QUANTIZATION (Diskrit dalam Amplitudo)
# Membulatkan nilai y ke level terdekat

# Normalisasi dulu dari range (-1 s/d 1) ke (0 s/d 1)
y_normalized = (y_sampled + 1) / 2

# Skala ke integer (0 s/d levels-1)
max_level_val = quantization_levels - 1
y_discrete_int = np.round(y_normalized * max_level_val)

# Kembalikan ke range aslinya (-1 s/d 1) untuk ditampilkan
y_quantized = (y_discrete_int / max_level_val) * 2 - 1

# VISUALISASI
plt.figure(figsize=(14, 6))

# Plot 1: Proses Sampling (Titik-titik pengambilan)
plt.subplot(1, 2, 1)
plt.plot(t_analog, y_analog, label='Sinyal Asli (Analog)', color='lightgray',
linewidth=2)
plt.stem(t_sampled, y_sampled, linefmt='b-', markerfmt='bo', basefmt='r-',
label='Titik Sample')
plt.title(f'1. Proses Sampling\n(Rate: {sampling_rate} Hz)')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')
plt.legend()
plt.grid(True, alpha=0.3)

# Plot 2: Hasil Digital (Kotak-kotak)
plt.subplot(1, 2, 2)
plt.plot(t_analog, y_analog, label='Sinyal Asli', color='lightgray', linestyle='--')

# Step plot untuk menunjukkan efek digital yang "patah-patah"

```

```
plt.step(t_sampled, y_quantized, where='mid', label='Sinyal Digital', color='red',
linewidth=2)
```

```
plt.title(f'2. Hasil Digitalisasi\n(Bit Depth: {quantization_levels} Levels)')
```

```
plt.xlabel('Waktu (detik)')
```

```
# Tampilkan garis grid horizontal sesuai level kuantisasi
```

```
plt.yticks(np.linspace(-1, 1, quantization_levels))
```

```
plt.legend()
```

```
plt.grid(True, alpha=0.3)
```

```
plt.tight_layout()
```

```
plt.show()
```

```
# --- MAIN EXECUTION ---
```

```
if __name__ == "__main__":
```

```
    # SKENARIO 1: Kualitas Rendah (Patah-patah)
```

```
    print("\nMenampilkan Skenario 1: Kualitas Rendah...")
```

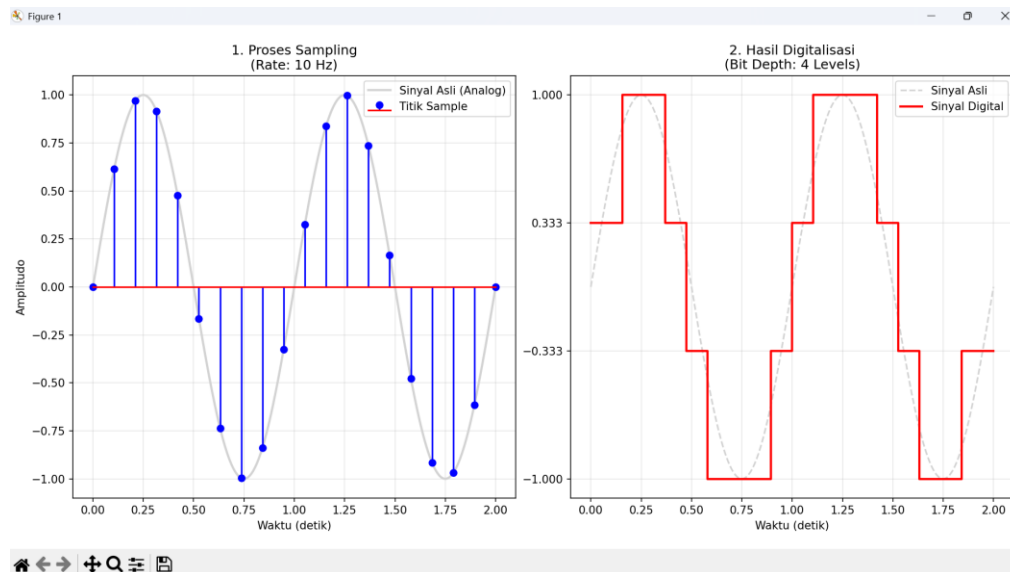
```
    simulate_digitization(freq=1.0,          duration=2.0,          sampling_rate=10,
quantization_levels=4)
```

```
    # SKENARIO 2: Kualitas Tinggi (Lebih halus)
```

```
    # Uncomment baris di bawah ini jika ingin mencoba skenario 2 setelah menutup
window pertama
```

```
    # print("\nMenampilkan Skenario 2: Kualitas Tinggi...")
```

```
    #      simulate_digitization(freq=1.0,      duration=2.0,      sampling_rate=50,
quantization_levels=32)
```



=== LATIHAN 2: SIMULASI SAMPLING & KUANTISASI ===

Menampilkan Skenario 1: Kualitas Rendah...

PARAMETER SIMULASI:

- Frekuensi Sinyal : 1.0 Hz
- Sampling Rate : 10 Hz (Sample per detik)
- Level Kuantisasi : 4 Levels (Bit Depth)

3. Lampiran 3: Kode Program Dasar-Dasar Citra Digital (Praktikum1.py)

Program ini mencakup implementasi manipulasi bit depth, separasi channel warna RGB, dan visualisasi histogram intensitas.

```
# =====
# PRAKTIKUM 1: DASAR-DASAR CITRA DIGITAL
# =====
```

```
import cv2
```

```
import numpy as np
```

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

```
import requests
```

```
from io import BytesIO
```

```
from PIL import Image
```

```
print("=== PRAKTIKUM 1: DASAR-DASAR CITRA DIGITAL ===")
```

```
print("Materi: Representasi Citra, Resolusi, Depth, Aspect Ratio\n")
```

```
# ===== FUNGSI BANTU =====

def download_sample_image():
    """Download sample image from internet"""
    url = "https://asset.tribunnews.com/ChI-
WuyDbEdFg3OosdCQaGawAV8=/1200x675/filters:upscale():quality(30):format(web
p):focal(0.5x0.5:0.5x0.5)/aceh/foto/bank/originals/Menteri-ESDM-Bahlil-Lahadalia-
saat-ditemui-di-Istana-Jakarta-Jumat-732025.jpg"
    response = requests.get(url)
    img = Image.open(BytesIO(response.content))
    return cv2.cvtColor(np.array(img), cv2.COLOR_RGB2BGR)

def analyze_image_properties(img, name="Image"):
    if len(img.shape) == 2:
        height, width = img.shape
        channels = 1
    else:
        height, width, channels = img.shape

    resolution = width * height
    aspect_ratio = width / height
    depth = img.dtype.itemsize * 8

    print(f'\n{' '*40}")
    print(f'ANALYSIS: {name}')
    print(f'{' '*40}")
    print(f'Dimensions: {width} x {height}')
    print(f'Channels: {channels}')
    print(f'Resolution: {resolution:,} pixels')
    print(f'Aspect Ratio: {aspect_ratio:.2f} ({width}:{height})')
    print(f'Bit Depth: {depth}-bit ({img.dtype})')

    # Calculate memory size
    memory_bytes = img.size * img.dtype.itemsize
```

```

memory_kb = memory_bytes / 1024
memory_mb = memory_kb / 1024

print(f"Memory Size: {memory_bytes:,} bytes")
print(f"        {memory_kb:.2f} KB")
print(f"        {memory_mb:.2f} MB")

# Calculate statistics
if channels == 1:
    print(f"Intensity Range: [{img.min()}, {img.max()}]")
    print(f"Mean Intensity: {img.mean():.2f}")
    print(f"Std Deviation: {img.std():.2f}")

return {
    'width': width,
    'height': height,
    'channels': channels,
    'resolution': resolution,
    'aspect_ratio': aspect_ratio,
    'depth': depth,
    'memory_bytes': memory_bytes
}

def display_image_grid(images, titles, rows, cols, figsize=(15, 10)):
    """Display multiple images in a grid"""
    fig, axes = plt.subplots(rows, cols, figsize=figsize)
    axes = axes.ravel() if rows > 1 or cols > 1 else [axes]

    for idx, (img, title) in enumerate(zip(images, titles)):
        if len(img.shape) == 2:
            axes[idx].imshow(img, cmap='gray')
        else:
            axes[idx].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        axes[idx].set_title(title)

```

```

axes[idx].axis('off')

plt.tight_layout()
plt.show()

# ===== MAIN PRAKTIKUM =====

# 1. LOAD DAN ANALISIS CITRA SAMPLE
print("\n1. LOADING SAMPLE IMAGE")
original_img = download_sample_image()
props_original = analyze_image_properties(original_img, "Original Color Image")

# 2. REPRESENTASI MATRIKS
print("\n2. REPRESENTASI SEBAGAI MATRIKS")
print("Mengakses nilai pixel pada posisi tertentu:")

# Contoh akses pixel
x, y = 100, 100 # Koordinat (baris, kolom)
pixel_value = original_img[x, y]
print(f"Pixel pada posisi ({x}, {y}): BGR = {pixel_value}")

# Tampilkan area kecil 5x5 pixel
print("\nArea 5x5 pixel dari posisi (100,100):")
print(original_img[100:105, 100:105])

# 3. KONVERSI GRAYSCALE
print("\n3. KONVERSI KE GRAYSCALE")
gray_img = cv2.cvtColor(original_img, cv2.COLOR_BGR2GRAY)
props_gray = analyze_image_properties(gray_img, "Grayscale Image")

# 4. ANALISIS BIT DEPTH YANG BERBEDA
print("\n4. PENGARUH BIT DEPTH")
# Buat citra dengan bit depth berbeda
img_8bit = gray_img.astype(np.uint8)

```

```

img_4bit = (gray_img // 16).astype(np.uint8) * 16 # Reduce to 4-bit (16 levels)
img_2bit = (gray_img // 64).astype(np.uint8) * 64 # Reduce to 2-bit (4 levels)
img_1bit = (gray_img // 128).astype(np.uint8) * 255 # Binary image

```

```

# Tampilkan perbedaan

```

```

images = [img_8bit, img_4bit, img_2bit, img_1bit]
titles = ['8-bit (256 levels)', '4-bit (16 levels)', '2-bit (4 levels)', '1-bit (2 levels)']
display_image_grid(images, titles, 1, 4, figsize=(16, 4))

```

5. ANALISIS ASPECT RATIO

```

print("\n\n5. PENGARUH ASPECT RATIO")

```

```

# Resize dengan aspect ratio berbeda

```

```

h, w = gray_img.shape[:2]
img_4_3 = cv2.resize(gray_img, (800, 600)) # 4:3
img_16_9 = cv2.resize(gray_img, (800, 450)) # 16:9
img_1_1 = cv2.resize(gray_img, (600, 600)) # 1:1
img_21_9 = cv2.resize(gray_img, (840, 360)) # 21:9

```

```

images = [img_4_3, img_16_9, img_1_1, img_21_9]
titles = ['4:3 (800x600)', '16:9 (800x450)', '1:1 (600x600)', '21:9 (840x360)']
display_image_grid(images, titles, 2, 2, figsize=(12, 8))

```

6. SEPARASI CHANNEL WARNA

```

print("\n\n6. SEPARASI CHANNEL WARNA RGB")

```

```

# Pisahkan channel B, G, R

```

```

b, g, r = cv2.split(original_img)

```

```

# Buat citra untuk masing-masing channel

```

```

zeros = np.zeros_like(b)
blue_channel = cv2.merge([b, zeros, zeros])
green_channel = cv2.merge([zeros, g, zeros])
red_channel = cv2.merge([zeros, zeros, r])

```

```

images = [original_img, blue_channel, green_channel, red_channel]

```

```
titles = ['Original', 'Blue Channel', 'Green Channel', 'Red Channel']
display_image_grid(images, titles, 2, 2, figsize=(12, 8))
```

```
# 7. HISTOGRAM INTENSITAS
```

```
print("\n\n7. ANALISIS HISTOGRAM INTENSITAS")
```

```
fig, axes = plt.subplots(1, 3, figsize=(15, 4))
```

```
# Histogram grayscale
```

```
axes[0].hist(gray_img.ravel(), 256, [0, 256], color='gray')
```

```
axes[0].set_title('Grayscale Histogram')
```

```
axes[0].set_xlabel('Intensity')
```

```
axes[0].set_ylabel('Frequency')
```

```
axes[0].grid(True, alpha=0.3)
```

```
# Histogram per channel warna
```

```
colors = ('b', 'g', 'r')
```

```
for i, color in enumerate(colors):
```

```
    hist = cv2.calcHist([original_img], [i], None, [256], [0, 256])
```

```
    axes[1].plot(hist, color=color)
```

```
axes[1].set_title('Color Channel Histograms')
```

```
axes[1].set_xlabel('Intensity')
```

```
axes[1].set_ylabel('Frequency')
```

```
axes[1].legend(['Blue', 'Green', 'Red'])
```

```
axes[1].grid(True, alpha=0.3)
```

```
# Cumulative histogram
```

```
cumulative_hist = np.cumsum(np.histogram(gray_img.ravel(), 256, [0, 256])[0])
```

```
axes[2].plot(cumulative_hist, color='purple', linewidth=2)
```

```
axes[2].set_title('Cumulative Histogram')
```

```
axes[2].set_xlabel('Intensity')
```

```
axes[2].set_ylabel('Cumulative Frequency')
```

```
axes[2].grid(True, alpha=0.3)
```

```
plt.tight_layout()
```

```

plt.show()

# 8. MEMORY ANALYSIS
print("\n\n8. ANALISIS UKURAN MEMORI")
print("Perbandingan ukuran memori untuk berbagai format:")

# Buat citra dengan ukuran berbeda
sizes = [(640, 480), (1920, 1080), (3840, 2160)]
formats = ['Grayscale (8-bit)', 'RGB (24-bit)', 'RGBA (32-bit)']

print("\n" + "="*60)
print(f'{{Resolution':<15}} {{Format':<20}} {{Memory':>15}}")
print("-"*60)

for w, h in sizes:
    for fmt_idx, fmt_name in enumerate(formats):
        if fmt_name == 'Grayscale (8-bit)':
            channels = 1
            depth = 1 # byte per pixel
        elif fmt_name == 'RGB (24-bit)':
            channels = 3
            depth = 3
        else: # RGBA
            channels = 4
            depth = 4

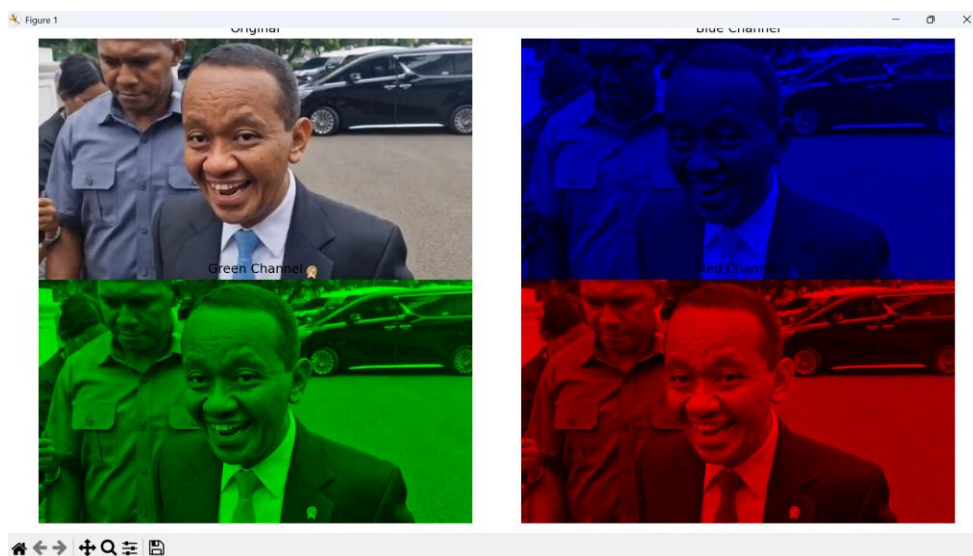
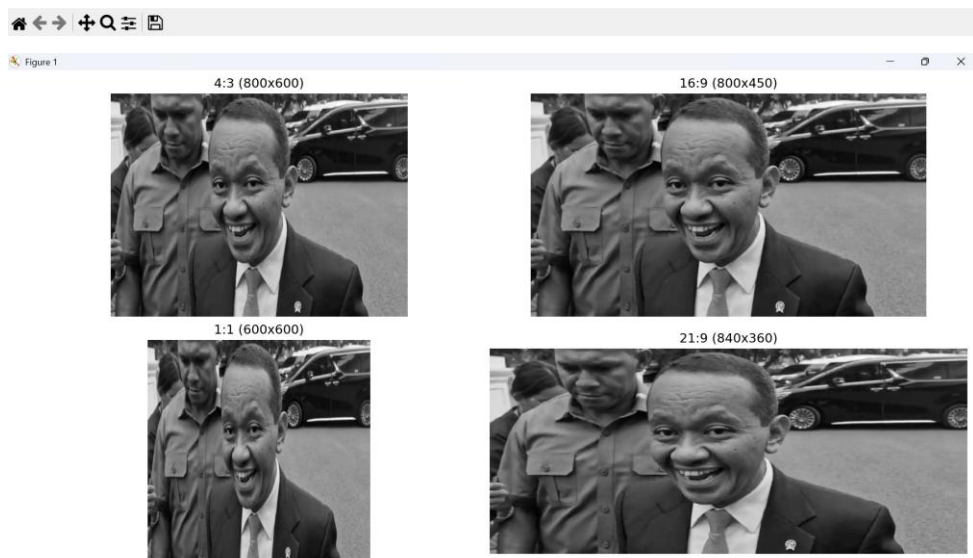
        memory = w * h * depth
        memory_kb = memory / 1024
        memory_mb = memory_kb / 1024

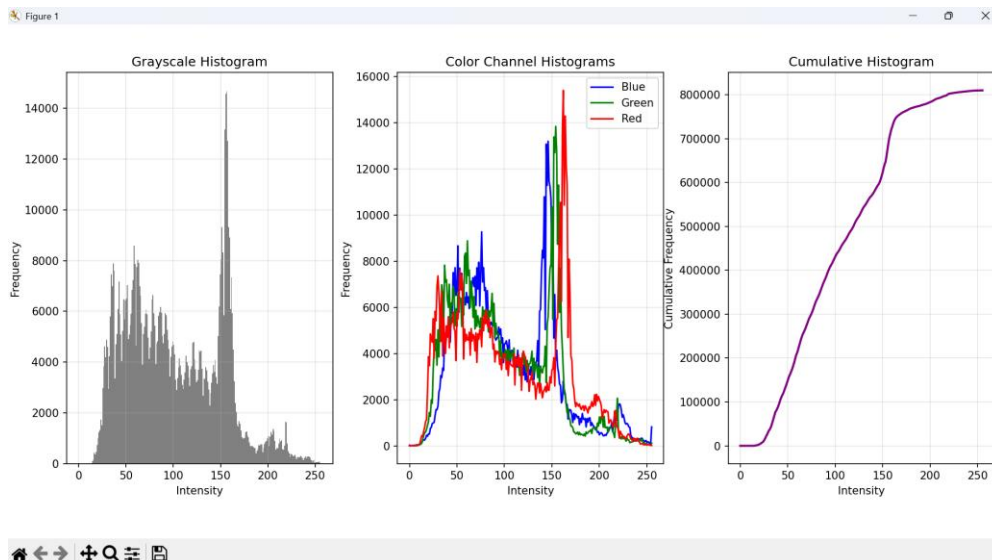
        print(f'{{w}}x{{h:<9}} {{fmt_name:<20}} {{memory:.,}} bytes ({{memory_mb:.1f}}
        MB)")

    print("="*60)

```

```
print("\n=== PRAKTIKUM SELESAI ===")
```





```
=== PRAKTIKUM 1: DASAR-DASAR CITRA DIGITAL ===
Materi: Representasi Citra, Resolusi, Depth, Aspect Ratio
```

1. LOADING SAMPLE IMAGE

```
=====
ANALYSIS: Original Color Image
=====
```

```
Dimensions: 1200 x 675
Channels: 3
Resolution: 810,000 pixels
Aspect Ratio: 1.78 (1200:675)
Bit Depth: 8-bit (uint8)
Memory Size: 2,430,000 bytes
              2373.05 KB
              2.32 MB
```

2. REPRESENTASI SEBAGAI MATRIKS

```
Mengakses nilai pixel pada posisi tertentu:
Pixel pada posisi (100, 100): BGR = [58 46 41]
```

```
Area 5x5 pixel dari posisi (100,100):
```

```
[[[58 46 41]
   [58 46 41]
   [59 47 40]
   [59 47 40]
   [59 47 42]]]
```

```
[[[58 46 41]
   [58 46 41]
   [60 49 41]
   [60 49 41]
   [60 48 43]]]
```

```
[[59 47 42]
 [59 47 42]
 [61 50 43]
 [61 50 43]
 [61 49 44]]
```

```
[[59 47 42]
 [59 47 42]
 [61 50 43]
 [60 49 41]
 [60 48 43]]
```

```
[[59 47 42]
 [59 47 42]
 [60 48 43]
 [60 48 43]
 [60 48 43]]
```

3. KONVERSI KE GRAYSCALE

```
=====
ANALYSIS: Grayscale Image
=====
```

```
Dimensions: 1200 x 675
Channels: 1
Resolution: 810,000 pixels
Aspect Ratio: 1.78 (1200:675)
Bit Depth: 8-bit (uint8)
Memory Size: 810,000 bytes
              791.02 KB
              0.77 MB
Intensity Range: [7, 255]
Mean Intensity: 101.73
Std Deviation: 49.91
```

4. PENGARUH BIT DEPTH

5. PENGARUH ASPECT RATIO

6. SEPARASI CHANNEL WARNA RGB

7. ANALISIS HISTOGRAM INTENSITAS

```
d:\SEMESTER 6\Pengolahan Citra Digital\Pertemuan 1\Praktikum1.py:155: MatplotlibDeprecationWarning: Passing the range parameter of hist() positionally is deprecated since Matplotlib 3.10; the parameter will become keyword-only in 3.12.
  axes[0].hist(gray_img.ravel(), 256, [0, 256], color='gray')
```

8. ANALISIS UKURAN MEMORI

Perbandingan ukuran memori untuk berbagai format:

```
=====
Resolution      Format      Memory
-----
640x480          Grayscale (8-bit)  307,200 bytes (0.3 MB)
640x480          RGB (24-bit)       921,600 bytes (0.9 MB)
640x480          RGBA (32-bit)      1,228,800 bytes (1.2 MB)
1920x1080        Grayscale (8-bit)  2,073,600 bytes (2.0 MB)
1920x1080        RGB (24-bit)       6,220,800 bytes (5.9 MB)
1920x1080        RGBA (32-bit)      8,294,400 bytes (7.9 MB)
3840x2160        Grayscale (8-bit)  8,294,400 bytes (7.9 MB)
3840x2160        RGB (24-bit)       24,883,200 bytes (23.7 MB)
3840x2160        RGBA (32-bit)      33,177,600 bytes (31.6 MB)
=====
```

```
=== PRAKTIKUM SELESAI ===
```