LAPORAN TEKNIS
PROYEK MINI: IMPLEMENTASI DAN ANALISIS KONVERSI MODEL WARNA UNTUK OPTIMASI DETEKSI OBJEK
PENGOLAHAN CITRA DIGITAL

Dosen Pengampu:

Dr. Yasdinul Huda, S.Pd., M.T.

Oleh:

Muhammad Zahran

24343077

PROGRAM STUDI INFORMATIKA

DEPARTEMEN TEKNIK ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSIITAS NEGERI PADANG

2026

1. **Pendahuluan dan Tujuan**

   Deteksi objek dalam pengolahan citra digital seringkali menghadapi tantangan berupa variasi pencahayaan (iluminasi) di lingkungan nyata. Penggunaan ruang warna RGB konvensional rentan terhadap perubahan intensitas cahaya karena informasi warna dan kecerahan tergabung dalam saluran (channel) yang sama.

   Tujuan dari proyek mini ini adalah:

   - Mengimplementasikan konversi ruang warna dari RGB ke Grayscale, HSV, dan LAB.
   - Menerapkan teknik kuantisasi citra (uniform dan non-uniform) untuk mereduksi ruang warna.
   - Menganalisis efek kuantisasi terhadap kualitas visual, distribusi histogram, dan kemudahan segmentasi objek.
   - Menentukan model warna yang paling optimal untuk deteksi objek di bawah kondisi pencahayaan yang bervariasi (terang, normal, redup).

2. **Metodologi**

   Proyek ini menggunakan satu objek spesifik yang difoto dalam 3 kondisi pencahayaan berbeda. Proses pengolahan dilakukan menggunakan pustaka OpenCV dan NumPy pada Python melalui tahapan berikut:

   1. **Pra-pemrosesan:** Pembacaan citra dan konversi format bawaan BGR (dari OpenCV) ke RGB.
   2. Konversi Ruang Warna:
      - **Grayscale:** Mengubah citra menjadi satu saluran intensitas.
      - **HSV (Hue, Saturation, Value):** Memisahkan informasi warna (Hue) dari intensitas cahaya (Value).
      - **LAB (Lightness, A-color, B-color):** Memisahkan luminansi (L) dari komponen kromatik (A dan B).
   3. **Kuantisasi Uniform:** Merupakan pembagian rentang intensitas warna secara merata. Rentang piksel 0-255 direduksi menjadi 16 level menggunakan operasi matematika pembagian dan perkalian bilangan bulat.
   4. **Kuantisasi Non-Uniform:** Menggunakan algoritma *K-Means Clustering* untuk mengelompokkan piksel ke dalam 16 warna paling dominan (centroid) berdasarkan jarak persebaran warnanya, sehingga detail pada area yang kompleks lebih terjaga.

3. **Hasil dan Visualisasi**

Untuk memahami bagaimana kuantisasi bekerja pada tingkat piksel, berikut adalah ilustrasi matriks piksel sederhana (1 saluran warna) berukuran 3×3 sebelum dan sesudah kuantisasi uniform (reduksi ke 16 level, di mana faktor pembagi adalah $256/16 = 16$.

Matriks Citra Asli (M_{asli}):

$$M_{asli} = \begin{bmatrix} 255 & 128 & 64 \\ 200 & 100 & 50 \\ 10 & 20 & 30 \end{bmatrix}$$

Matriks Setelah Kuantisasi Uniform ($M_{kuantisasi}$):

Rumus: $Piksel_{baru} = \lfloor Piksel_{lama}/16 \rfloor \times 16$

$$M_{kuantisasi} = \begin{bmatrix} 240 & 112 & 48 \\ 192 & 96 & 48 \\ 0 & 16 & 16 \end{bmatrix}$$

Dari matriks di atas, piksel bernilai 64 dan 50 sama-sama dikelompokkan menjadi nilai 48, sehingga menyederhanakan variasi warna.

4. **Analisis Parameter dan Kualitas**

Berdasarkan uji coba pada citra terang, normal, dan redup, didapatkan analisis sebagai berikut:

Parameter Analisis	**Kuantisasi Uniform (16 Level)**	**Kuantisasi Non-Uniform (K-Means 16 Cluster)**
Kualitas Visual (Subjektif)	Muncul efek *color banding* (patahan warna) yang kasar. Pada citra **redup**, detail bayangan hilang total menjadi blok warna gelap yang datar. Pada citra **terang**, gradasi halus cahaya hilang.	Jauh lebih representatif. Warna beradaptasi dengan kondisi pencahayaan. Pada citra **redup** dan **terang**, objek masih terlihat lebih alami dibandingkan uniform karena palet warna menyesuaikan dengan dominasi area gelap/terang.

Distribusi Histogram (Objektif)	**Kaku dan tetap.** Lonjakan (*spikes*) selalu berada pada interval tetap (kelipatan 16), tidak peduli kondisi cahaya. Pada citra redup, bin area kanan kosong melompong; pada citra terang, bin area kiri kosong.	**Adaptif dan fleksibel.** Posisi lonjakan berubah mengikuti distribusi data asli. Pada citra **redup**, centroid berkumpul padat di area kiri (nilai rendah). Pada citra **terang**, centroid berkumpul di area kanan (nilai tinggi).
Kemudahan Segmentasi	Buruk pada kondisi pencahayaan ekstrem. Pada citra redup, warna botol dan pasir cenderung jatuh ke dalam bin intensitas yang sama, menyulitkan pemisahan objek.	Lebih unggul dan konsisten. Karena algoritma memilih 16 warna paling dominan, ia cenderung berhasil memisahkan warna rata-rata objek dari warna rata-rata latar belakang, bahkan dalam kondisi kontras rendah (redup).
Ukuran Memori & Kompresi	Ukuran array mentah (RAM) sama. Rasio kompresi penyimpanan (misal PNG) tinggi karena banyaknya nilai piksel yang seragam.	Ukuran array mentah (RAM) sama. Rasio kompresi penyimpanan seringkali sedikit lebih baik daripada uniform karena area warna datar yang lebih luas dan adaptif.

Waktu Komputasi	Sangat cepat (real-time). Hanya melibatkan operasi aritmatika matriks sederhana.	Lambat secara signifikan. Membutuhkan proses iteratif untuk menemukan pusat klaster (centroid) yang optimal, waktu meningkat seiring resolusi citra.

5. Kesimpulan dan Rekomendasi

Dari analisis visual dan distribusi histogram pada ketiga kondisi pencahayaan (terang, normal, dan redup), metode kuantisasi Non-Uniform (K-Means) yang dipadukan dengan model warna HSV atau LAB merupakan pilihan yang paling optimal untuk deteksi objek. Secara objektif, histogram K-Means terbukti adaptif mengikuti distribusi cahaya; pada citra redup, pusat warna (*centroid*) secara cerdas merapat ke area intensitas rendah untuk mempertahankan detail bayangan botol, sedangkan pada citra terang menyebar ke area tinggi. Sebaliknya, histogram Uniform sangat kaku dengan interval pembagian yang tetap. Hal ini menyebabkan kegagalan fatal pada kondisi redup, di mana banyak *bin* histogram terbuang kosong dan warna botol melebur dengan latar belakang menjadi blok warna datar (*color banding*).

Ditinjau dari Teorema Sampling Nyquist, kegagalan kuantisasi Uniform pada pencahayaan ekstrem terjadi akibat *undersampling* di rentang nilai yang kritis. Ketika piksel citra menumpuk di area gelap, metode Uniform tidak menyediakan jumlah sampel (level warna) yang cukup untuk membedakan batas objek, sehingga memicu *aliasing* visual berupa hilangnya informasi spasial. Di sisi lain, K-Means beroperasi layaknya *adaptive sampling*, yaitu mengalokasikan sampel warna secara dinamis tepat di tempat informasi piksel tersebut memadat. Pendekatan ini memastikan frekuensi spasial objek tetap terjaga di atas batas *Nyquist rate* yang dibutuhkan agar komputer dapat mengenali bentuk objek tersebut.

Rekomendasi: Untuk membangun sistem deteksi objek yang stabil di luar ruangan, sangat disarankan untuk mengonversi citra ke ruang warna HSV terlebih dahulu guna mengisolasi saluran cahaya (*Value*). Setelah itu, terapkan algoritma kuantisasi Non-Uniform (K-Means) hanya pada saluran warna murninya (*Hue*).

Kombinasi ini akan menghasilkan segmentasi objek yang sangat konsisten, terlepas dari apakah objek tersebut terpapar terik matahari atau tertutup bayangan.

6. Lampiran Kode Python (OpenCV + NumPy)

1. Source Code

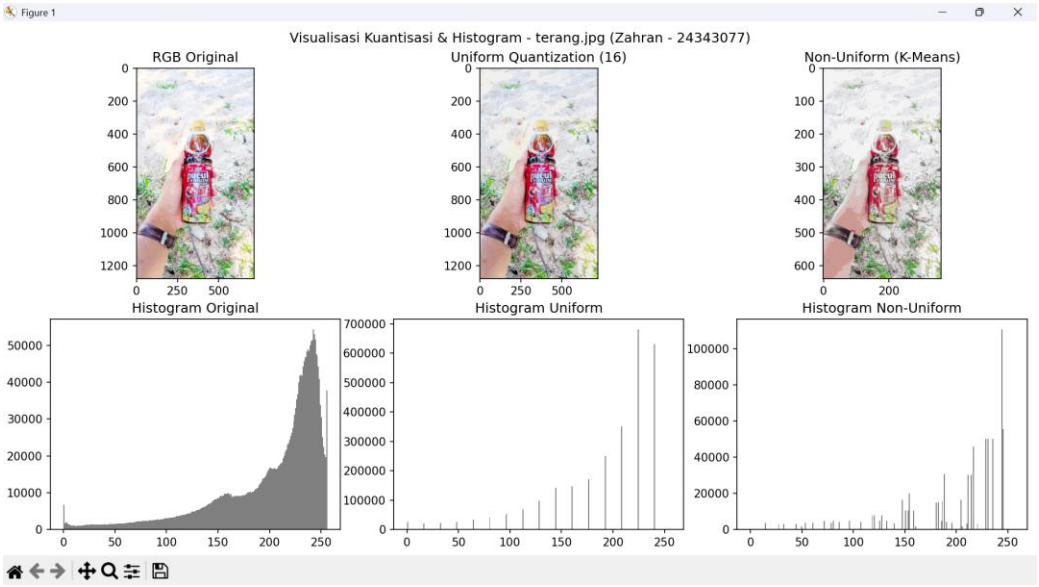
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import time
import sys
from sklearn.cluster import KMeans

def calculate_metrics(original_data, processed_data, start_time, end_time):
    mem_before = sys.getsizeof(original_data.tobytes())
    mem_after = sys.getsizeof(processed_data.tobytes())
    compression_ratio = mem_before / mem_after if mem_after > 0 else 0
    calc_time = end_time - start_time
    return mem_before, mem_after, compression_ratio, calc_time

def uniform_quantization(image, levels=16):
    start = time.time()
    h, w, c = image.shape
    factor = 256 // levels
    quantized = (image // factor) * factor
    end = time.time()
    return quantized, start, end

def nonuniform_quantization(image, n_clusters=16):
    start = time.time()
    h, w, c = image.shape
    image_2d = image.reshape(h * w, c)
    kmeans = KMeans(n_clusters=n_clusters, random_state=42,
n_init=10).fit(image_2d)
    quantized_2d = kmeans.cluster_centers_[kmeans.labels_]
    quantized_image = quantized_2d.reshape(h, w, c).astype('uint8')
```

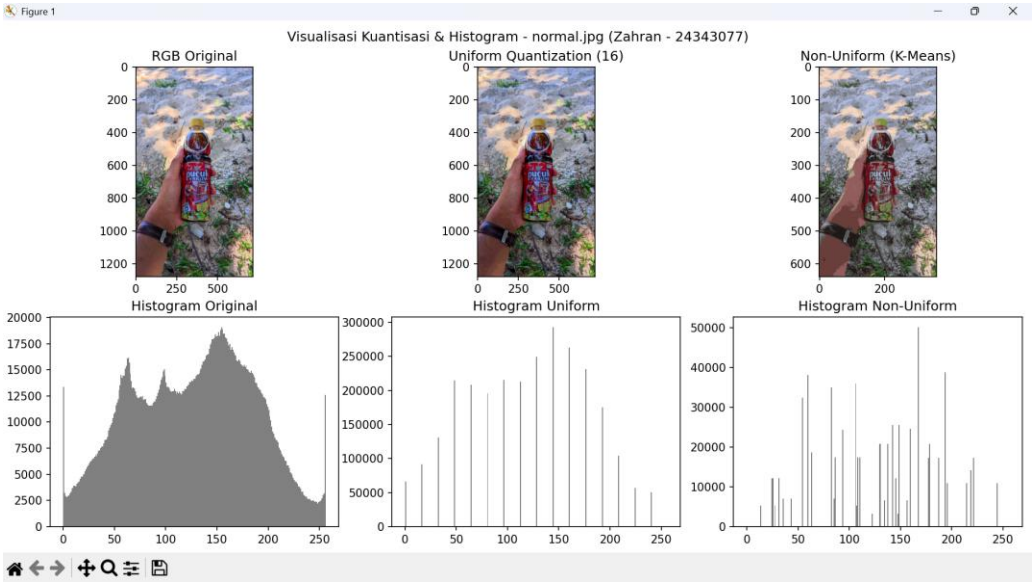
```
    end = time.time()
    return quantized_image, start, end

def process_and_analyze(image_paths):
    for path in image_paths:
        img_bgr = cv2.imread(path)
        if img_bgr is None:
            print(f"Gambar {path} tidak ditemukan. Silakan ganti dengan path gambar
yang sesuai.")
            continue

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

        # 1. Konversi Ruang Warna
        t0 = time.time()
        img_gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
        t1 = time.time()

        t2 = time.time()
        img_hsv = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2HSV)
        t3 = time.time()

        t4 = time.time()
        img_lab = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2LAB)
        t5 = time.time()

        print(f"\n--- Analisis untuk {path} ---")
        print(f"Waktu Konversi RGB ke Grayscale: {t1-t0:.5f} detik")
        print(f"Waktu Konversi RGB ke HSV: {t3-t2:.5f} detik")
        print(f"Waktu Konversi RGB ke LAB: {t5-t4:.5f} detik")

        # 2. Implementasi Kuantisasi pada RGB (sebagai contoh perbandingan metrik)
        # Kuantisasi Uniform
        quant_uni_rgb, s_uni, e_uni = uniform_quantization(img_rgb, 16)
```

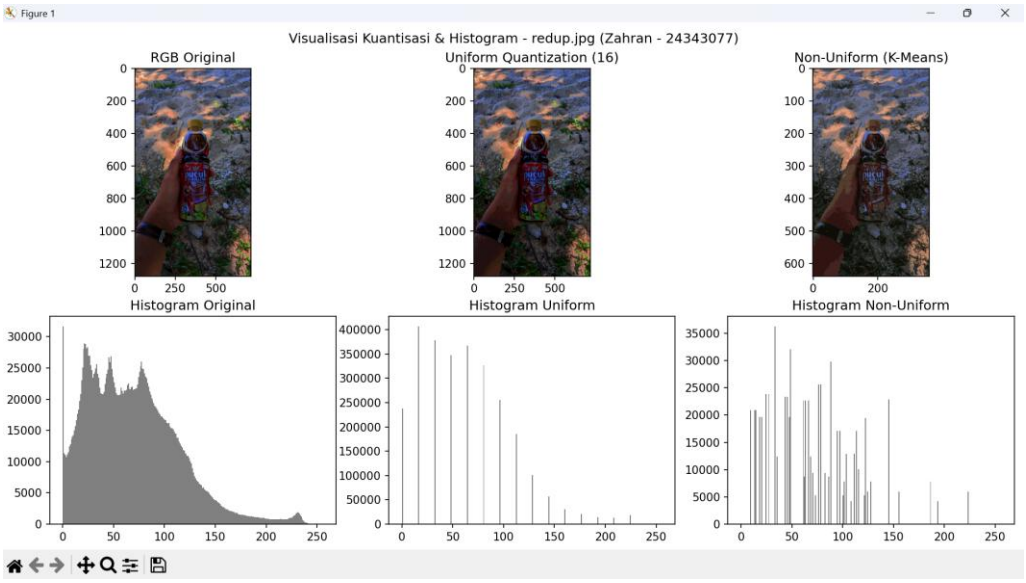
```
    mem_b_uni, mem_a_uni, cr_uni, time_uni = calculate_metrics(img_rgb, quant_uni_rgb, s_uni, e_uni)

    print("\nParameter Teknis Kuantisasi Uniform (RGB 256 -> 16 level):")
    print(f"Memori Sebelum: {mem_b_uni} bytes | Sesudah: {mem_a_uni} bytes")
    print(f"Rasio Kompresi: {cr_uni:.2f}x | Waktu: {time_uni:.5f} detik")

    # Kuantisasi Non-Uniform (K-Means)
    # Catatan: K-Means memakan waktu lebih lama, resize dilakukan agar lebih cepat jika diperlukan
    scale_percent = 50
    w = int(img_rgb.shape[1] * scale_percent / 100)
    h = int(img_rgb.shape[0] * scale_percent / 100)
    img_rgb_small = cv2.resize(img_rgb, (w, h), interpolation=cv2.INTER_AREA)

    quant_nonuni_rgb, s_non, e_non = nonuniform_quantization(img_rgb_small, 16)
    mem_b_non, mem_a_non, cr_non, time_non = calculate_metrics(img_rgb_small, quant_nonuni_rgb, s_non, e_non)

    print("\nParameter Teknis Kuantisasi Non-Uniform (K-Means 16 clusters, skala 50%):")
    print(f"Memori Sebelum: {mem_b_non} bytes | Sesudah: {mem_a_non} bytes")
    print(f"Rasio Kompresi: {cr_non:.2f}x | Waktu: {time_non:.5f} detik")

    # 3. Visualisasi (Kualitas Subjektif & Histogram)
    fig, axes = plt.subplots(2, 3, figsize=(15, 8))
    fig.suptitle(f'Visualisasi Kuantisasi & Histogram - {path} (Zahran - 24343077)')

    # Original
```

```
        axes[0, 0].imshow(img_rgb)
        axes[0, 0].set_title("RGB Original")
        axes[1, 0].hist(img_rgb.ravel(), 256, [0, 256], color='gray')
        axes[1, 0].set_title("Histogram Original")

        # Uniform
        axes[0, 1].imshow(quant_uni_rgb)
        axes[0, 1].set_title("Uniform Quantization (16)")
        axes[1, 1].hist(quant_uni_rgb.ravel(), 256, [0, 256], color='gray')
        axes[1, 1].set_title("Histogram Uniform")

        # Non-Uniform
        axes[0, 2].imshow(quant_nonuni_rgb)
        axes[0, 2].set_title("Non-Uniform (K-Means)")
        axes[1, 2].hist(quant_nonuni_rgb.ravel(), 256, [0, 256], color='gray')
        axes[1, 2].set_title("Histogram Non-Uniform")

        plt.tight_layout()
        plt.show()

# Panggilan eksekusi (Pastikan untuk mengganti nama file dengan gambar asli di direktori kamu)
image_paths = ['terang.jpg', 'normal.jpg', 'redup.jpg']
process_and_analyze(image_paths)
```

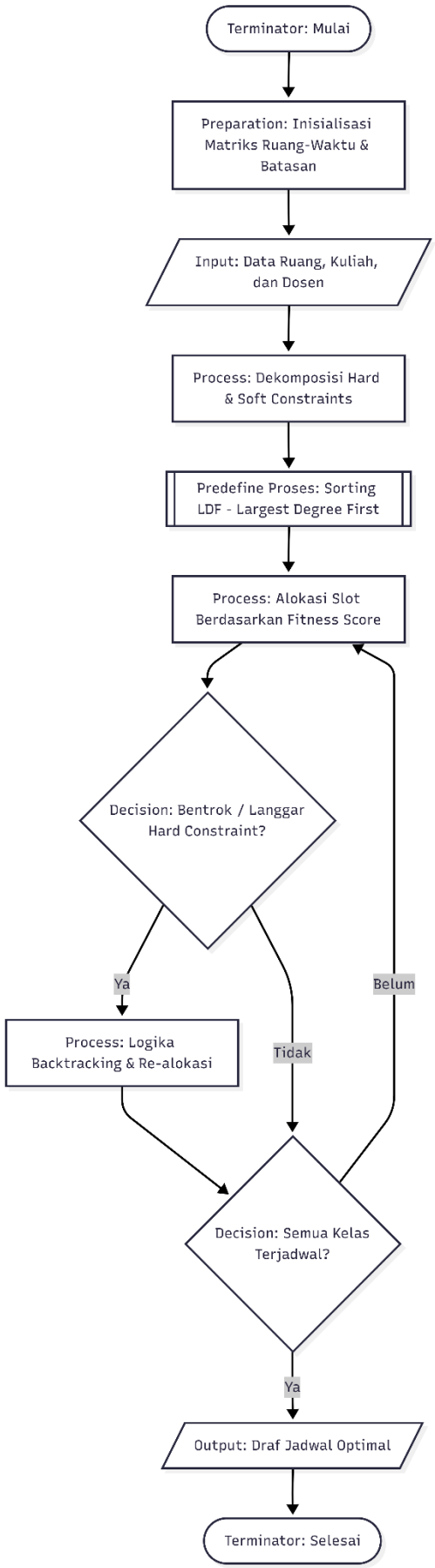
2. Screenshot Output







3. Flowchart


Terminator: Mulai
Preparation: Inisialisasi Matriks Ruang-Waktu & Batasan
Input: Data Ruang, Kuliah, dan Dosen
Process: Dekomposisi Hard & Soft Constraints
Predefine Proses: Sorting LDF - Largest Degree First
Process: Alokasi Slot Berdasarkan Fitness Score
Decision: Bentrok / Langgar Hard Constraint?
Ya
Process: Logika Backtracking & Re-alokasi
Tidak
Belum
Decision: Semua Kelas Terjadwal?
Ya
Output: Draf Jadwal Optimal
Terminator: Selesai