

LAPORAN TUGAS KECIL2

Kompresi Gambar Dengan Metode Quadtree

Disusun untuk Memenuhi Tugas Kecil 2 Mata Kuliah Strategi Algoritma IF2211

Dosen Pengampu:

Dr. Ir. Rinaldi Munir, M.T.

Dr. Nur Ulfa Maulidevi, S.T, M.Sc.



Felix Chandra 13523012

Ahmad Ibrahim 13523089

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2025

DAFTAR ISI

I. LANDASAN TEORI	3
1.1. Quadtree	3
1.2. Algoritma Divide and Conquer	3
1.3. Metode Pengukuran Error	4
II. IMPLEMENTASI ALGORITMA	5
2.1. Source Program	5
2.2. Penerapan Algoritma Divide and Conquer	14
III. PENGUJIAN	16
3.1. Variance	16
Percobaan Test Case 1 (Threshold = 0) :	16
Percobaan Test Case 2 (Threshold = 65025):	17
3.2. Mean Absolute Deviation (MAD)	18
Percobaan Test Case 3 (Threshold = 0) :	19
Percobaan Test Case 4 (Threshold = 255) :	20
3.3. Max Pixel Difference	21
Percobaan Test Case 5 (Threshold = 0) :	21
Percobaan Test Case 6 (Threshold = 255) :	22
3.4. Entropy	23
Percobaan Test Case 7 (Threshold = 0) :	24
Percobaan Test Case 8 (Threshold = 8) :	25
3.5. SSIM	26
Percobaan TestCase 9 (Threshold = 0) :	26
Percobaan TestCase 10 (Threshold = 2) :	27
IV. ANALISIS HASIL PENGUJIAN	29
4.1. Pengaruh Nilai Threshold	29
4.1.1. Ukuran File Output	29
4.1.2. Waktu Eksekusi	29
4.1.3. Kualitas Visual	29
4.2. Analisi Kompleksitas	29
4.2.1. Kompleksitas Fungsi Pembantu	29
4.2.2. Kompleksitas buildQuadtree	30
4.2.2.1. Analisis Waktu	30
4.2.2.2. Analisis Ruang:	30
4.2.3. Kompleksitas Keseluruhan	31

I. LANDASAN TEORI

1.1. Quadtree

Quadtree adalah sebuah struktur data berbentuk pohon (tree) yang dirancang khusus untuk mempartisi ruang dua dimensi secara rekursif. Setiap simpul (node) internal dalam Quadtree selalu memiliki tepat empat anak, yang masing-masing merepresentasikan satu dari empat kuadran (biasanya Barat Laut, Timur Laut, Barat Daya, Tenggara) dari ruang atau wilayah yang diwakili oleh simpul induknya. Proses pembagian ini dimulai dari simpul akar (root) yang mewakili keseluruhan ruang dua dimensi, dan berlanjut secara rekursif: jika suatu wilayah perlu dibagi lebih lanjut (misalnya karena terlalu kompleks, berisi terlalu banyak data, atau warnanya tidak homogen dalam konteks gambar), maka simpul yang mewakilinya akan dibagi menjadi empat simpul anak. Pembagian ini berhenti ketika suatu kuadran mencapai kondisi tertentu, seperti hanya berisi satu titik data, warnanya cukup seragam (menjadi simpul daun/leaf node), atau telah mencapai tingkat kedalaman/resolusi maksimum yang diinginkan. Struktur ini sangat efisien untuk aplikasi seperti kompresi gambar (di mana daun menyimpan warna rata-rata), pengindeksan spasial (untuk pencarian cepat objek dalam area), analisis data geografis (GIS), dan deteksi tabrakan dalam simulasi.

1.2. Algoritma Divide and Conquer

Algoritma Divide and Conquer adalah sebuah paradigma atau strategi pemecahan masalah yang kuat dalam ilmu komputer, yang bekerja dengan memecah masalah besar menjadi beberapa sub-masalah yang lebih kecil dan sejenis. Proses ini terdiri dari tiga langkah utama: Divide (Membagi), di mana masalah asli dipecah menjadi dua atau lebih sub-masalah yang ukurannya lebih kecil dan independen; Conquer (Menaklukkan/Menyehlesaikan), di mana setiap sub-masalah diselesaikan secara rekursif, dan jika sub-masalah sudah cukup kecil (mencapai kasus dasar atau base case), maka diselesaikan secara langsung; serta Combine (Menggabungkan), di mana solusi-solusi dari sub-masalah tersebut kemudian digabungkan kembali untuk membentuk solusi akhir dari masalah asli. Pendekatan ini seringkali menghasilkan

algoritma yang sangat efisien (seperti pada Merge Sort atau Quick Sort) karena mampu mengurangi kompleksitas komputasi secara signifikan, dan sifat independen dari sub-masalah membuatnya cocok untuk diproses secara paralel.

1.3. Metode Pengukuran Error

Dalam kode kompresi gambar Quadtree ini, beberapa metode error digunakan sebagai kriteria untuk menentukan apakah suatu wilayah (kuadran) gambar cukup homogen untuk direpresentasikan oleh satu warna rata-rata (menjadi simpul daun) atau masih terlalu kompleks sehingga perlu dibagi lagi menjadi empat sub-wilayah. Pilihan metode ini diberikan kepada pengguna dan berfungsi sebagai pengukur tingkat variasi atau detail dalam kuadran tersebut. Metode yang tersedia meliputi Variance (mengukur sebaran nilai warna piksel dari rata-ratanya), Mean Absolute Deviation (MAD) (mengukur rata-rata simpangan absolut warna piksel dari rata-ratanya), Max Pixel Difference (mengukur rentang maksimum antara nilai piksel terang dan gelap per channel warna), Entropy (mengukur tingkat ketidakpastian atau keragaman informasi warna), dan Structural Similarity Index Measure (SSIM) (mencoba mengukur kesamaan struktural antara wilayah asli dengan representasi warna rata-ratanya). Nilai error yang dihitung dari metode terpilih kemudian dibandingkan dengan ambang batas (threshold) yang ditetapkan pengguna; jika error di bawah ambang batas, pembagian berhenti, jika di atas, pembagian berlanjut, sehingga secara langsung memengaruhi tingkat kompresi dan detail visual gambar hasil.

II. IMPLEMENTASI ALGORITMA

2.1. Source Program

Source program

```
#include <iostream>
#include <vector>
#include <string>
#include <cmath>
#include <numeric>
#include <limits>
#include <chrono>
#include <fstream>
#include <filesystem>

#define cimg_display 0
#include "CImg-3.5.4_pre04072515/CImg.h"

using namespace cimg_library;
using namespace std;

struct RGB {
    double r = 0.0, g = 0.0, b = 0.0;
};

class QuadtreeNode {
public:
    RGB avgColor;
    int x, y, width, height;
    bool isLeaf = false;

    QuadtreeNode *children[4] = {nullptr, nullptr, nullptr, nullptr}; // NW, NE, SW, SE
    int depth = 0;

    QuadtreeNode(int _x, int _y, int _w, int _h, int _d) : x(_x), y(_y), width(_w),
    height(_h), depth(_d) {}

    ~QuadtreeNode() {
        for (int i = 0; i < 4; ++i) {
            delete children[i];
        }
    }
};

void printWarning(string message) {
    cout << "\033[1;31m  !! " << message << "\033[0m\n";
}

void printCommand(string message) {
    cout << "\n\033[2m——\033[0m" << message << "\n\033[2m└\033[0;32m» \033[0m";
}
```

```

void printLine(string message) {
    cout << "\033[1;32m \u25b6 " << message << "\033[0m\n";
}

void clearCinBuffer() {
    cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
}

long long getFileSizeStream(const std::string& filepath) {
    std::ifstream file(filepath, std::ios::binary | std::ios::ate);
    std::streampos size = file.tellg();
    file.close();
    return static_cast<long long>(size);
}

RGB calculateAverageColor(const CImg<unsigned char>& image, int x, int y, int width,
int height) {
    RGB avg = {};
    double numPixels = static_cast<double>(width) * height;
    if (numPixels == 0) return avg;

    double r_sum = 0.0, g_sum = 0.0, b_sum = 0.0;

    for (int j = y; j < y + height; ++j) {
        for (int i = x; i < x + width; ++i) {
            r_sum += image(i, j, 0, 0);
            g_sum += image(i, j, 0, 1);
            b_sum += image(i, j, 0, 2);
        }
    }
    avg.r = r_sum / numPixels;
    avg.g = g_sum / numPixels;
    avg.b = b_sum / numPixels;
    return avg;
}

double calculateErrorVariance(const CImg<unsigned char>& image, int x, int y, int
width, int height, const RGB& avgColor) {
    double numPixels = static_cast<double>(width) * height;
    if (numPixels <= 1) return 0.0;

    double r_var_sum = 0.0, g_var_sum = 0.0, b_var_sum = 0.0;

    for (int j = y; j < y + height; ++j) {
        for (int i = x; i < x + width; ++i) {
            double r_diff = image(i, j, 0, 0) - avgColor.r;
            double g_diff = image(i, j, 0, 1) - avgColor.g;
            double b_diff = image(i, j, 0, 2) - avgColor.b;
            r_var_sum += r_diff * r_diff;
            g_var_sum += g_diff * g_diff;
            b_var_sum += b_diff * b_diff;
        }
    }

    double r_variance = r_var_sum / numPixels;
    double g_variance = g_var_sum / numPixels;
    double b_variance = b_var_sum / numPixels;

    return (r_variance + g_variance + b_variance) / 3.0;
}

```

```

double calculateErrorMAD(const CIImg<unsigned char>& image, int x, int y, int width, int
height, const RGB& avgColor) {
    double numPixels = static_cast<double>(width) * height;
    if (numPixels == 0) return 0.0;

    double r_mad_sum = 0.0, g_mad_sum = 0.0, b_mad_sum = 0.0;

    int x_end = min(x + width, image.width());
    int y_end = min(y + height, image.height());

    for (int j = y; j < y_end; ++j) {
        for (int i = x; i < x_end; ++i) {
            r_mad_sum += abs(image(i, j, 0, 0) - avgColor.r);
            g_mad_sum += abs(image(i, j, 0, 1) - avgColor.g);
            b_mad_sum += abs(image(i, j, 0, 2) - avgColor.b);
        }
    }

    numPixels = static_cast<double>(x_end - x) * (y_end - y);
    if (numPixels <= 0) return 0.0;

    double r_mad = r_mad_sum / numPixels;
    double g_mad = g_mad_sum / numPixels;
    double b_mad = b_mad_sum / numPixels;

    return (r_mad + g_mad + b_mad) / 3.0;
}

double calculateErrorMaxDiff(const CIImg<unsigned char>& image, int x, int y, int width,
int height) {
    if (width <= 0 || height <= 0) return 0.0;
    unsigned char minR = 255, maxR = 0;
    unsigned char minG = 255, maxG = 0;
    unsigned char minB = 255, maxB = 0;
    for (int j = y; j < y + height; ++j) {
        for (int i = x; i < x + width; ++i) {
            unsigned char r = image(i, j, 0, 0);
            unsigned char g = image(i, j, 0, 1);
            unsigned char b = image(i, j, 0, 2);
            if (r < minR) minR = r;
            if (r > maxR) maxR = r;
            if (g < minG) minG = g;
            if (g > maxG) maxG = g;
            if (b < minB) minB = b;
            if (b > maxB) maxB = b;
        }
    }

    double DR = static_cast<double>(maxR - minR);
    double DG = static_cast<double>(maxG - minG);
    double DB = static_cast<double>(maxB - minB);
    double DRGB = (DR + DG + DB) / 3.0;
    return DRGB;
}

double calculateEntropy(const CIImg<unsigned char>& image, int x, int y, int width, int
height) {
    if (width <= 0 || height <= 0) return 0.0;
    std::vector<int> freqR(256, 0), freqG(256, 0), freqB(256, 0);
    int totalPixels = width * height;
    for (int j = y; j < y + height; ++j) {

```

```

        for (int i = x; i < x + width; ++i) {
            unsigned char r = image(i, j, 0, 0);
            unsigned char g = image(i, j, 0, 1);
            unsigned char b = image(i, j, 0, 2);
            freqR[r]++;
            freqG[g]++;
            freqB[b]++;
        }
    }

    auto calculateChannelEntropy = [&](const std::vector<int>& freq) -> double {
        double entropy = 0.0;
        for (int i = 0; i < 256; ++i) {
            if (freq[i] > 0) {
                double probability = static_cast<double>(freq[i]) / totalPixels;
                entropy -= probability * std::log2(probability);
            }
        }
        return entropy;
    };

    double entropyR = calculateChannelEntropy(freqR);
    double entropyG = calculateChannelEntropy(freqG);
    double entropyB = calculateChannelEntropy(freqB);
    double entropyRGB = (entropyR + entropyG + entropyB) / 3.0;
    return entropyRGB;
}

double calculateSSIM(const CImg<unsigned char>& originalImage, int x, int y, int width,
int height,
    const RGB& avgColor) {
    const double L = 255.0;
    const double k1 = 0.01;
    const double k2 = 0.03;
    const double C1 = (k1 * L) * (k1 * L); // (k1*L)^2
    const double C2 = (k2 * L) * (k2 * L); // (k2*L)^2

    const double wR = 1.0/3.0;
    const double wG = 1.0/3.0;
    const double wB = 1.0/3.0;

    double meanR_orig = 0.0, meanG_orig = 0.0, meanB_orig = 0.0;
    double varR_orig = 0.0, varG_orig = 0.0, varB_orig = 0.0;

    for (int j = y; j < y + height; ++j) {
        for (int i = x; i < x + width; ++i) {
            meanR_orig += originalImage(i, j, 0, 0);
            meanG_orig += originalImage(i, j, 0, 1);
            meanB_orig += originalImage(i, j, 0, 2);
        }
    }

    double numPixels = width * height;
    meanR_orig /= numPixels;
    meanG_orig /= numPixels;
    meanB_orig /= numPixels;

    for (int j = y; j < y + height; ++j) {
        for (int i = x; i < x + width; ++i) {
            double diffR = originalImage(i, j, 0, 0) - meanR_orig;
            double diffG = originalImage(i, j, 0, 1) - meanG_orig;
            double diffB = originalImage(i, j, 0, 2) - meanB_orig;
        }
    }
}

```

```

        varR_orig += diffR * diffR;
        varG_orig += diffG * diffG;
        varB_orig += diffB * diffB;
    }
}

varR_orig /= numPixels;
varG_orig /= numPixels;
varB_orig /= numPixels;

double meanR_comp = avgColor.r;
double meanG_comp = avgColor.g;
double meanB_comp = avgColor.b;

double covR = 0.0, covG = 0.0, covB = 0.0;

for (int j = y; j < y + height; ++j) {
    for (int i = x; i < x + width; ++i) {
        double diffR_orig = originalImage(i, j, 0, 0) - meanR_orig;
        double diffG_orig = originalImage(i, j, 0, 1) - meanG_orig;
        double diffB_orig = originalImage(i, j, 0, 2) - meanB_orig;

        double diffR_comp = originalImage(i, j, 0, 0) - meanR_comp;
        double diffG_comp = originalImage(i, j, 0, 1) - meanG_comp;
        double diffB_comp = originalImage(i, j, 0, 2) - meanB_comp;

        covR += diffR_orig * diffR_comp;
        covG += diffG_orig * diffG_comp;
        covB += diffB_orig * diffB_comp;
    }
}

covR /= numPixels;
covG /= numPixels;
covB /= numPixels;

double ssimR = ((2 * meanR_orig * meanR_comp + C1) * (2 * covR + C2)) /
    ((meanR_orig * meanR_orig + meanR_comp * meanR_comp + C1) * (varR_orig + 0 + C2));

double ssimG = ((2 * meanG_orig * meanG_comp + C1) * (2 * covG + C2)) /
    ((meanG_orig * meanG_orig + meanG_comp * meanG_comp + C1) * (varG_orig + 0 + C2));

double ssimB = ((2 * meanB_orig * meanB_comp + C1) * (2 * covB + C2)) /
    ((meanB_orig * meanB_orig + meanB_comp * meanB_comp + C1) * (varB_orig + 0 + C2));

double ssimRGB = wR * ssimR + wG * ssimG + wB * ssimB;
return ssimRGB;
}

long long nodeCount = 0;
int maxDepth = 0;

QuadtreeNode* buildQuadtree(const CImg<unsigned char>& image, int x, int y, int width,
int height,
                           double threshold, int minBlockSize, int currentDepth, int
errorMethod) {

    nodeCount++;
    if (currentDepth > maxDepth) {
        maxDepth = currentDepth;
    }
}

```

```

QuadtreeNode* node = new QuadtreeNode(x, y, width, height, currentDepth);

node->avgColor = calculateAverageColor(image, x, y, width, height);

double error = 0.0;
if (errorMethod == 1) {
    error = calculateErrorVariance(image, x, y, width, height, node->avgColor);
} else if (errorMethod == 2) {
    error = calculateErrorMAD(image, x, y, width, height, node->avgColor);
} else if (errorMethod == 3) {
    error = calculateErrorMaxDiff(image, x, y, width, height);
} else if (errorMethod == 4) {
    error = calculateEntropy(image, x, y, width, height);
} else if (errorMethod == 5) {
    error = calculateSSIM(image, x, y, width, height, node->avgColor);
} else {
    printWarning("Metode error tidak valid!");
    error = 0;
}

int nextWidth = width / 2;
int nextHeight = height / 2;
bool subBlocksTooSmall = (nextWidth == 0 || nextHeight == 0);

if (error < threshold || (width * height) <= minBlockSize || subBlocksTooSmall ) {
    node->isLeaf = true;
} else {
    node->isLeaf = false;

    int w1 = nextWidth;
    int h1 = nextHeight;
    int w2 = width - w1;
    int h2 = height - h1;

    if (w1 > 0 && h1 > 0)
        node->children[0] = buildQuadtree(image, x, y, w1, h1, threshold,
minBlockSize, currentDepth + 1, errorMethod); // NW
    if (w2 > 0 && h1 > 0)
        node->children[1] = buildQuadtree(image, x + w1, y, w2, h1, threshold,
minBlockSize, currentDepth + 1, errorMethod); // NE
    if (w1 > 0 && h2 > 0)
        node->children[2] = buildQuadtree(image, x, y + h1, w1, h2, threshold,
minBlockSize, currentDepth + 1, errorMethod); // SW
    if (w2 > 0 && h2 > 0)
        node->children[3] = buildQuadtree(image, x + w1, y + h1, w2, h2, threshold,
minBlockSize, currentDepth + 1, errorMethod); // SE
}

return node;
}

void reconstructImage(CImg<unsigned char>& outputImage, const QuadtreeNode* node) {
    if (node == nullptr) {
        return;
    }

    if (node->isLeaf) {
        unsigned char color[3];

```

```

color[0] = static_cast<unsigned char>(max(0.0, min(255.0, node->avgColor.r)));
color[1] = static_cast<unsigned char>(max(0.0, min(255.0, node->avgColor.g)));
color[2] = static_cast<unsigned char>(max(0.0, min(255.0, node->avgColor.b)));

// CImg draw_rectangle(x0, y0, x1, y1, color, opacity)
outputImage.draw_rectangle(node->x, node->y,
                           node->x + node->width - 1, node->y + node->height -
1,
                           color, 1.0f);
} else {
    for (int i = 0; i < 4; ++i) {
        reconstructImage(outputImage, node->children[i]);
    }
}
}

int main() {
    string inputFile, outputFile = "test/default.png";
    string errorStr = "";
    int errorMethodChoice;
    double threshold, maxThreshold;
    int minBlockSize;
    CImg<unsigned char> inputImage;

    cout << endl;
    printLine("===== Kompresi Gambar Quadtree =====");
    cout << "\033[1;32m \u25b6 Current Working Directory: " <<
std::filesystem::current_path() << "\033[0m\n";

    while (true) {
        printCommand("Masukkan path gambar input");
        getline(cin, inputFile);

        try {
            CImg<unsigned char> testImg(inputFile.c_str());
            inputImage = testImg;
            if (inputImage.spectrum() != 3) {
                printWarning("Gambar input tidak memiliki 3 channel warna (RGB).
Program mungkin tidak berfungsi benar.");
            }
            break;
        } catch (const cimg_library::CImgException& e) {
            printWarning("Error: Gambar tidak dikenali.");
            continue;
        }
    }
    long long inputSize = getFileSizeStream(inputFile);

    while (true) {
        printCommand("Pilih metode error (1=Variance, 2=MAD, 3=MaxDiff, 4=Entropy,
5=SSIM)");
        cin >> errorMethodChoice;

        if (cin.fail()) {
            printWarning("Error: Masukan harus berupa angka.");
            cin.clear();
            clearCinBuffer();
            continue;
        }
    }
}

```

```

        if (errorMethodChoice >= 1 && errorMethodChoice <= 5) {
            break;
        } else {
            printWarning("Error: Pilihan metoda harus antara 1 hingga 5 (inklusif).");
        }
    }

    switch (errorMethodChoice) {
        case 1:
            maxThreshold = 65025.0;
            errorStr = "Variance";
            break;
        case 2:
            maxThreshold = 255.0;
            errorStr = "Mean Absolute Deviation (MAD)";
            break;
        case 3:
            maxThreshold = 255.0;
            errorStr = "Max Pixel Difference";
            break;
        case 4:
            maxThreshold = 8.0;
            errorStr = "Entropy";
            break;
        case 5:
            maxThreshold = 2;
            errorStr = "SSIM";
            break;
    }

    while (true) {
        printCommand("Masukkan nilai threshold untuk " + errorStr +
                    " (rentang efektif: 0.0 - " + to_string(maxThreshold) + ")");
        cin >> threshold;

        if (cin.fail()) {
            printWarning("Error: Masukan harus berupa angka.");
            cin.clear();
            clearCinBuffer();
            continue;
        }

        if (threshold >= 0.0) {
            if (threshold > maxThreshold) {
                printWarning("Nilai di atas " + to_string(maxThreshold) + " kemungkinan
tidak akan berpengaruh pada kompresi");
            }
            break;
        } else {
            printWarning("Error: Threshold tidak boleh negatif.");
        }
    }

    while (true) {
        printCommand("Masukkan ukuran blok minimum (>= 1)");
        cin >> minBlockSize;

        if (cin.fail()) {
            printWarning("Error: Masukan harus berupa angka integer.");
            cin.clear();
            clearCinBuffer();
        }
    }
}

```

```

        continue;
    }

    if (minBlockSize >= 1) {
        break;
    } else {
        printWarning("Error: Ukuran blok minimum harus 1 atau lebih besar.");
    }
}

cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
printCommand("Masukkan path gambar output");
getline(cin, outputFile);

// --- Process ---
cout << "\n";
try {
    printLine("Memulai Kompresi...");
    CImg<unsigned char> outputImage(inputImage.width(), inputImage.height(), 1, 3,
0);

    auto startTime = chrono::high_resolution_clock::now();

    nodeCount = 0;
    maxDepth = 0;

    QuadtreeNode* root = buildQuadtree(inputImage, 0, 0, inputImage.width(),
inputImage.height(),
threshold, minBlockSize, 0, errorMethodChoice);

    reconstructImage(outputImage, root);

    auto endTime = chrono::high_resolution_clock::now();
    chrono::duration<double, milli> duration = endTime - startTime;
    printLine("Kompresi Selesai.");

    printLine("Menyimpan file gambar...");
    size_t dotPos = outputFile.find_last_of(".");
    if (dotPos == string::npos) {
        outputFile += ".png";
        printWarning("Ekstensi file output tidak dispesifikasi. Menyimpan sebagai
PNG.");
    }
    outputImage.save(outputFile.c_str());
    printLine("Gambar tersimpan.");

    printLine("");

    // --- Statistik ---
    printLine("--- Statistik ---");
    printLine("Path gambar output      : " + outputFile);
    printLine("Waktu eksekusi          : " + to_string(duration.count()) + " ms");

    printLine("Resolusi Gambar         : " + to_string(inputImage.width()) + "x" +
to_string(inputImage.height()));

    printLine("Metode error            : " + errorStr);
    printLine("Threshold               : " + to_string(threshold));
    printLine("Blok Minimum             : " + to_string(minBlockSize));
    printLine("Ukuran sebelum           : " + to_string(inputSize / (1024.0 *
1024.0)) + " MiB (" +

```

```

        to_string(inputSize) + " bytes");

    long long outputSize = getFileSizeStream(outputFile);
    if (outputSize >= 0) {
        printLine("Ukuran sesudah           : " + to_string(outputSize / (1024.0 *
1024.0)) + " MiB (" +
                           to_string(outputSize) + " bytes");

        if (inputSize > 0) {
            double compressionRatio = static_cast<double>(outputSize) / inputSize;
            printLine("Rasio kompresi         : " + to_string(compressionRatio));

            double compressionPercentage = (1.0 - compressionRatio) * 100.0;
            printLine("Persentase kompresi     : " +
to_string(compressionPercentage) + " %");
        } else {
            printLine("Rasio kompresi         : N/A (Ukuran input 0 atau error)");
            printLine("Persentase kompresi     : N/A");
        }
    } else {
        printWarning("Tidak dapat membaca ukuran file output.");
    }

    printLine("Kedalaman pohon maks   : " + to_string(maxDepth));
    printLine("Jumlah simpul total    : " + to_string(nodeCount));
    cout << endl;

    delete root;

} catch (CImgException &e) {
    printWarning("Error CImg: " + std::string(e.what()));
    return 1;
} catch (exception &e) {
    printWarning("Error: " + std::string(e.what()));
    return 1;
}

return 0;
}

```

2.2. Penerapan Algoritma Divide and Conquer

Algoritma Divide and Conquer diterapkan secara fundamental dalam fungsi buildQuadtree untuk membangun struktur Quadtree yang merepresentasikan gambar. Prosesnya dimulai dengan Divide, di mana wilayah gambar saat ini (awalnya seluruh gambar) secara konseptual dibagi menjadi empat sub-wilayah atau kuadran (Barat Laut, Timur Laut, Barat Daya, Tenggara). Langkah Conquer kemudian dijalankan dengan memanggil fungsi buildQuadtree secara rekursif untuk masing-masing kuadran tersebut, secara efektif menyelesaikan sub-masalah yang lebih kecil (membangun Quadtree untuk sub-wilayah). Rekursi ini memiliki Base Case atau

kondisi berhenti: jika suatu wilayah dinilai cukup homogen berdasarkan metrik error yang dipilih (misalnya, varians warnanya di bawah ambang batas yang ditentukan) atau jika ukurannya sudah mencapai batas minimum yang ditetapkan atau tidak dapat dibagi lagi, maka wilayah tersebut tidak dibagi lebih lanjut dan menjadi simpul daun (leaf node) yang menyimpan warna rata-rata wilayah itu. Langkah Combine dalam konteks ini terjadi secara implisit saat hasil dari pemanggilan rekursif (pointer ke simpul anak yang mewakili Quadtree untuk sub-wilayah) disimpan dalam array children milik simpul induknya, secara bertahap membangun keseluruhan struktur hierarkis Quadtree dari bawah ke atas hingga representasi seluruh gambar selesai.

III. PENGUJIAN

3.1. Variance



Gambar 1. Gambar Original 1

Percobaan Test Case 1 (Threshold = 0) :



Gambar 2. Hasil percobaan variance dengan threshold = 0

```

● felix@Felixs-MacBook-Air Tucil2_13523012_13523089 % make run
./bin/main

===== Kompresi Gambar Quadtree =====
Current Working Directory: "/Users/felix/Documents/GitHub/Tucil2_13523012_13523089"

Masukkan path gambar input
-> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/original.png
!! Gambar input tidak memiliki 3 channel warna (RGB). Program mungkin tidak berfungsi benar.

Pilih metode error (1=Variance, 2=MAD, 3=MaxDiff, 4=Entropy, 5=SSIM)
-> 1

Masukkan nilai threshold untuk Variance (rentang efektif: 0.0 - 65025.000000)
-> 0

Masukkan ukuran blok minimum (>= 1)
-> 1

Masukkan path gambar output
-> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc1.png

|| Memulai Kompresi...
|| Kompresi Selesai.
|| Menyimpan file gambar...
|| Gambar tersimpan.

--- Statistik ---
Path gambar output      : /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc1.png
Waktu eksekusi          : 196.610500 ms
Resolusi Gambar         : 1288x1564
Metode error             : Variance
Threshold                : 0.000000
Blok Minimum             : 1
Ukuran sebelum            : 1.389266 MiB (1456751 bytes)
Ukuran sesudah           : 0.846340 MiB (887452 bytes)
Rasio kompresi            : 0.609200
Persentase kompresi       : 39.080049 %
Kedalaman pohon maks     : 11
Jumlah simpul total      : 1968341

❖ felix@Felixs-MacBook-Air Tucil2_13523012_13523089 % 

```

Gambar 3. Tangkapan layar terminal percobaan 1 (threshold = 0)

Percobaan Test Case 2 (Threshold = 65025):



Gambar 4. Hasil percobaan variance dengan threshold = 65025

```

Masukkan path gambar input
>>> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/original.png
!! Gambar input tidak memiliki 3 channel warna (RGB). Program mungkin tidak berfungsi benar.

Pilih metode error (1=Variance, 2=MAD, 3=MaxDiff, 4=Entropy, 5=SSIM)
>>> 1

Masukkan nilai threshold untuk Variance (rentang efektif: 0.0 - 65025.000000)
>>> 65025

Masukkan ukuran blok minimum (>= 1)
>>> 1

Masukkan path gambar output
>>> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc2.png

Memulai Kompresi...
Kompresi Selesai.
Menyimpan file gambar...
Gambar tersimpan.

--- Statistik ---
Path gambar output      : /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc2.png
Waktu eksekusi          : 14.168042 ms
Resolusi Gambar         : 1288x1564
Metode error             : Variance
Threshold                : 65025.000000
Blok Minimum             : 1
Ukuran sebelum            : 1.389266 MiB (1456751 bytes)
Ukuran sesudah           : 0.008816 MiB (9244 bytes)
Rasio kompresi            : 0.006346
Percentase kompresi       : 99.365437 %
Kedalaman pohon maks     : 0
Jumlah simpul total      : 1

felix@Felixs-MacBook-Air Tucil2_13523012_13523089 %

```

Gambar 5. Tangkapan layar terminal percobaan 2 (threshold = 65025)

3.2. Mean Absolute Deviation (MAD)



Gambar 6. Gambar Original 2

Percobaan Test Case 3 (Threshold = 0) :



Gambar 7. Hasil percobaan MAD dengan threshold = 0

```
felix@Felixs-MacBook-Air Tucil2_13523012_13523089 % make run
./bin/main

[[= Kompresi Gambar Quadtree =====
Current Working Directory: "/Users/felix/Documents/GitHub/Tucil2_13523012_13523089"

Masukkan path gambar input
>>> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/original2.png

Pilih metode error (1=Variance, 2=MAD, 3=MaxDiff, 4=Entropy, 5=SSIM)
>>> 2

Masukkan nilai threshold untuk Mean Absolute Deviation (MAD) (rentang efektif: 0.0 - 255.000000)
>>> 0

Masukkan ukuran blok minimum (>= 1)
>>> 1

Masukkan path gambar output
>>> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc3.png

[[= Memulai Kompresi...
[Kompresi Selesai.
[Menyimpan file gambar...
[Gambar tersimpan.

--- Statistik ---
Path gambar output      : /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc3.png
Waktu eksekusi          : 33858.112625 ms
Resolusi Gambar          : 11648x8736
Metode error             : Mean Absolute Deviation (MAD)
Threshold                : 0.000000
Blok Minimum             : 1
Ukuran sebelum            : 199.891693 MiB (209601632 bytes)
Ukuran sesudah           : 137.898736 MiB (144597305 bytes)
Rasio kompresi            : 0.689867
Persentase kompresi       : 31.013273 %
Kedalaman pohon maks     : 14
Jumlah simpul total      : 96998741
```

Gambar 8. Tangkapan layar terminal percobaan 3 (threshold = 0)

Percobaan Test Case 4 (Threshold = 255) :



Gambar 9. Hasil percobaan MAD dengan threshold = 255

```
felix@Felixs-MacBook-Air Tucil2_13523012_13523089 % make run
./bin/main

===== Kompresi Gambar Quadtree =====
Current Working Directory: "/Users/felix/Documents/GitHub/Tucil2_13523012_13523089"

Masukkan path gambar input
>>> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/original2.png

Pilih metode error (1=Variance, 2=MAD, 3=MaxDiff, 4=Entropy, 5=SSIM)
->>> 2

Masukkan nilai threshold untuk Mean Absolute Deviation (MAD) (rentang efektif: 0.0 - 255.000000)
->>> 255

Masukkan ukuran blok minimum (>= 1)
->>> 1

Masukkan path gambar output
->>> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc4.png

Memulai Kompresi...
Kompresi Selesai.
Menyimpan file gambar...
Gambar tersimpan.

--- Statistik ---
Path gambar output      : /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc4.png
Waktu eksekusi          : 359.905541 ms
Resolusi Gambar         : 11648x8736
Metode error             : Mean Absolute Deviation (MAD)
Threshold                : 255.000000
Blok Minimum             : 1
Ukuran sebelum            : 199.891693 MiB (209601632 bytes)
Ukuran sesudah           : 0.307888 MiB (322844 bytes)
Rasio kompresi            : 0.001540
Percentase kompresi       : 99.845973 %
Kedalaman pohon maks     : 0
Jumlah simpul total       : 1
```

Gambar 10. Tangkapan layar terminal percobaan 4 (threshold = 255)

3.3. Max Pixel Difference



Gambar 11. Gambar Original 3

Percobaan Test Case 5 (Threshold = 0) :



Gambar 12. Hasil percobaan Max Pixel Difference dengan threshold = 0

```
● felix@Felixs-MacBook-Air Tucil2_13523012_13523089 % make run
./bin/main
===== Kompresi Gambar Quadtree ======
Current Working Directory: "/Users/felix/Documents/GitHub/Tucil2_13523012_13523089"
Masukkan path gambar input
-> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/original3.png
!! Gambar input tidak memiliki 3 channel warna (RGB). Program mungkin tidak berfungsi benar.

Pilih metode error (1=Variance, 2=MAD, 3=MaxDiff, 4=Entropy, 5=SSIM)
-> 3

Masukkan nilai threshold untuk Max Pixel Difference (rentang efektif: 0.0 - 255.000000)
-> 0

Masukkan ukuran blok minimum (>= 1)
-> 1

Masukkan path gambar output
-> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc5.png

Memulai Kompresi...
Kompresi Selesai.
Menyimpan file gambar...
Gambar tersimpan.

--- Statistik ---
Path gambar output      : /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc5.png
Waktu eksekusi          : 41.066291 ms
Resolusi Gambar         : 662x522
Metode error             : Max Pixel Difference
Threshold                : 0.000000
Blok Minimum             : 1
Ukuran sebelum            : 0.657385 MiB (689318 bytes)
Ukuran sesudah           : 0.500274 MiB (524575 bytes)
Rasio kompresi            : 0.761006
Persentase kompresi       : 23.899419 %
Kedalaman pohon maks     : 10
Jumlah simpul total      : 355525
```

Gambar 13. Tangkapan layar terminal percobaan 5 (threshold = 0)

Percobaan Test Case 6 (Threshold = 255) :



Gambar 14. Hasil percobaan Max Pixel Difference dengan threshold = 255

```

felix@Felixs-MacBook-Air Tucil2_13523012_13523089 % make run
./bin/main

===== Kompresi Gambar Quadtree =====
Current Working Directory: "/Users/felix/Documents/GitHub/Tucil2_13523012_13523089"

Masukkan path gambar input
>> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/original3.png
!! Gambar input tidak memiliki 3 channel warna (RGB). Program mungkin tidak berfungsi benar.

Pilih metode error (1=Variance, 2=MAD, 3=MaxDiff, 4=Entropy, 5=SSIM)
>> 3

Masukkan nilai threshold untuk Max Pixel Difference (rentang efektif: 0.0 - 255.000000)
>> 255

Masukkan ukuran blok minimum (>= 1)
>> 1

Masukkan path gambar output
>> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc6.png

Memulai Kompresi...
Kompresi Selesai.
Menyimpan file gambar...
Gambar tersimpan.

--- Statistik ---
Path gambar output      : /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc6.png
Waktu eksekusi          : 3.046958 ms
Resolusi Gambar         : 662x522
Metode error             : Max Pixel Difference
Threshold                : 255.000000
Blok Minimum             : 1
Ukuran sebelum           : 0.657385 MiB (689318 bytes)
Ukuran sesudah           : 0.002078 MiB (2179 bytes)
Rasio kompresi            : 0.003161
Percentase kompresi       : 99.683890 %
Kedalaman pohon maks     : 0
Jumlah simpul total      : 1

```

Gambar 15. Tangkapan layar terminal percobaan 6 (threshold = 255)

3.4. Entropy



Gambar 16. Gambar Original 4

Percobaan Test Case 7 (Threshold = 0) :



Gambar 17. Hasil percobaan entropy dengan threshold = 0

```
felix@Felixs-MacBook-Air Tucil2_13523012_13523089 % make run
./bin/main
=====
===== Kompresi Gambar Quadtree =====
Current Working Directory: "/Users/felix/Documents/GitHub/Tucil2_13523012_13523089"

Masukkan path gambar input
>> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/original4.png
!! Gambar input tidak memiliki 3 channel warna (RGB). Program mungkin tidak berfungsi benar.

Pilih metode error (1=Variance, 2=MAD, 3=MaxDiff, 4=Entropy, 5=SSIM)
>> 4

Masukkan nilai threshold untuk Entropy (rentang efektif: 0.0 - 8.000000)
>> 0

Masukkan ukuran blok minimum (>= 1)
>> 1

Masukkan path gambar output
>> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc7.png

===== Memulai Kompresi...
Kompresi Selesai.
Menyimpan file gambar...
Gambar tersimpan.

--- Statistik ---
Path gambar output      : /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc7.png
Waktu eksekusi          : 159.510292 ms
Resolusi Gambar          : 368x794
Metode error              : Entropy
Threshold                 : 0.000000
Blok Minimum              : 1
Ukuran sebelum             : 0.508694 MiB (533404 bytes)
Ukuran sesudah             : 0.260168 MiB (272806 bytes)
Rasio kompresi            : 0.511443
Persentase kompresi        : 48.855652 %
Kedalaman pohon maks       : 9
Jumlah simpul total        : 202069
```

Gambar 18. Tangkapan layar terminal percobaan 7 (threshold = 0)

Percobaan Test Case 8 (Threshold = 8) :



Gambar 19. Hasil percobaan entropy dengan threshold = 8

```
● felix@Felixs-MacBook-Air Tucil2_13523012_13523089 % make run
./bin/main

[[= Kompresi Gambar Quadtree =====
Current Working Directory: "/Users/felix/Documents/GitHub/Tucil2_13523012_13523089"

Masukkan path gambar input
-> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/original4.png
!! Gambar input tidak memiliki 3 channel warna (RGB). Program mungkin tidak berfungsi benar.

Pilih metode error (1=Variance, 2=MAD, 3=MaxDiff, 4=Entropy, 5=SSIM)
-> 4

Masukkan nilai threshold untuk Entropy (rentang efektif: 0.0 - 8.000000)
-> 8

Masukkan ukuran blok minimum (>= 1)
-> 1

Masukkan path gambar output
-> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc8.png

[[= Memulai Kompresi...
Kompresi Selesai.
Menyimpan file gambar...
Gambar tersimpan.

--- Statistik ---
Path gambar output      : /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc8.png
Waktu eksekusi          : 2.795833 ms
Resolusi Gambar         : 368x794
Metode error             : Entropy
Threshold                : 8.000000
Blok Minimum              : 1
Ukuran sebelum            : 0.508694 MiB (533404 bytes)
Ukuran sesudah           : 0.002454 MiB (2573 bytes)
Rasio kompresi            : 0.004824
Percentase kompresi       : 99.517626 %
Kedalaman pohon maks     : 0
Jumlah simpul total       : 1
```

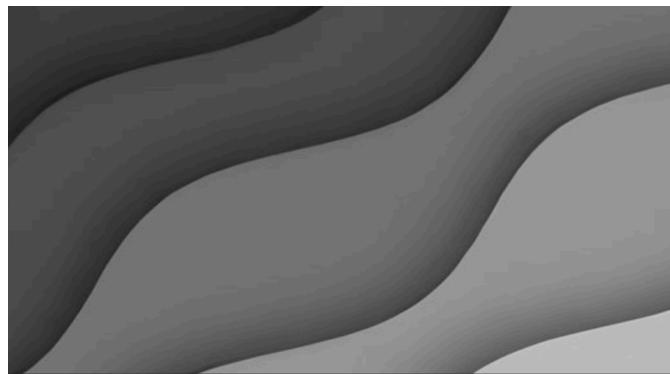
Gambar 20. Tangkapan layar terminal percobaan 8 (threshold = 8)

3.5. SSIM



Gambar 21. Gambar Original 5

Percobaan TestCase 9 (Threshold = 0) :



Gambar 22. Hasil percobaan SSIM dengan threshold = 0

```
Masukkan path gambar input
>> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/original5.png
!! Gambar input tidak memiliki 3 channel warna (RGB). Program mungkin tidak berfungsi benar.

Pilih metode error (1=Variance, 2=MAD, 3=MaxDiff, 4=Entropy, 5=SSIM)
>> 5

Masukkan nilai threshold untuk SSIM (rentang efektif: 0.0 - 2.000000)
>> 0

Masukkan ukuran blok minimum (>= 1)
>> 1

Masukkan path gambar output
>> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc9.png

||| Memulai Kompresi...
||| Kompresi Selesai.
||| Menyimpan file gambar...
||| Gambar tersimpan.

--- Statistik ---
Path gambar output      : /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc9.png
Waktu eksekusi          : 91.656250 ms
Resolusi Gambar         : 1030x566
Metode error             : SSIM
Threshold                : 0.000000
Blok Minimum             : 1
Ukuran sebelum           : 0.179434 MiB (188150 bytes)
Ukuran sesudah           : 0.084555 MiB (88662 bytes)
Rasio kompresi            : 0.471230
Persentase kompresi       : 52.876960 %
Kedalaman pohon maks     : 10
Jumlah simpul total      : 460117
```

Gambar 23. Tangkapan layar terminal percobaan 9 (threshold = 0)

Percobaan TestCase 10 (Threshold = 2) :



Gambar 24. Hasil percobaan SSIM dengan threshold = 2

```
    Masukkan path gambar input
>>> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/original5.png
!! Gambar input tidak memiliki 3 channel warna (RGB). Program mungkin tidak berfungsi benar.

    Pilih metode error (1=Variance, 2=MAD, 3=MaxDiff, 4=Entropy, 5=SSIM)
>>> 5

    Masukkan nilai threshold untuk SSIM (rentang efektif: 0.0 - 2.000000)
>>> 2

    Masukkan ukuran blok minimum (>= 1)
>>> 1

    Masukkan path gambar output
>>> /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc10.png

||| Memulai Kompresi...
||| Kompresi Selesai.
||| Menyimpan file gambar...
||| Gambar tersimpan.

--- Statistik ---
Path gambar output      : /Users/felix/Documents/GitHub/Tucil2_13523012_13523089/test/tc10.png
Waktu eksekusi          : 12.474375 ms
Resolusi Gambar         : 1030x566
Metode error             : SSIM
Threshold                : 2.000000
Blok Minimum             : 1
Ukuran sebelum            : 0.179434 MiB (188150 bytes)
Ukuran sesudah           : 0.002849 MiB (2987 bytes)
Rasio kompresi            : 0.015876
Persentase kompresi       : 98.412437 %
Kedalaman pohon maks     : 0
Jumlah simpul total      : 1
```

Gambar 25. Tangkapan layar terminal percobaan 10 (threshold = 2)

IV. ANALISIS HASIL PENGUJIAN

4.1. Pengaruh Nilai Threshold

4.1.1. Ukuran File Output

Threshold Tinggi: Algoritma akan lebih mudah "puas" dan menganggap suatu wilayah cukup homogen. Ini akan menghasilkan lebih sedikit pembagian (*subdivision*), pohon Quadtree yang lebih dangkal, dan blok-blok warna rata-rata yang lebih besar. Blok-blok besar yang seragam ini cenderung lebih mudah dikompresi oleh algoritma kompresi standar (seperti PNG), sehingga ukuran file output cenderung lebih kecil.

Threshold Rendah: Algoritma memerlukan tingkat keseragaman yang sangat tinggi. Ini akan menyebabkan lebih banyak pembagian, pohon yang lebih dalam, dan blok-blok warna yang lebih kecil untuk merepresentasikan detail. Struktur yang lebih kompleks ini menghasilkan gambar yang lebih detail tetapi lebih sulit dikompresi, sehingga ukuran file output cenderung lebih besar.

4.1.2. Waktu Eksekusi

Threshold Tinggi: Lebih sedikit pembagian berarti lebih sedikit pemanggilan fungsi rekursif, lebih sedikit perhitungan error, dan lebih sedikit operasi penggambaran blok. Waktu eksekusi cenderung lebih cepat.

Threshold Rendah: Lebih banyak pembagian berarti lebih banyak pemanggilan rekursif, perhitungan error, dan operasi menggambar. Waktu eksekusi cenderung lebih lama.

4.1.3. Kualitas Visual

Threshold Tinggi: Blok warna rata-rata yang besar akan menghilangkan detail halus dan tekstur, menghasilkan gambar yang terlihat "kotak-kotak" (*blocky*) atau kabur. Kualitas visual lebih rendah.

Threshold Rendah: Blok warna yang lebih kecil lebih mampu mereplikasi detail dan gradasi warna dari gambar asli. Kualitas visual lebih tinggi, lebih mendekati gambar asli.

4.2. Analisi Kompleksitas

4.2.1. Kompleksitas Fungsi Pembantu

`calculateAverageColor(..., k)`: Melakukan iterasi tunggal pada k piksel untuk menjumlahkan nilai R, G, B. Kompleksitas Waktu: $O(k)$. Kompleksitas Ruang: $O(1)$.

`calculateErrorVariance(..., k, ...)`: Setelah warna rata-rata diketahui, fungsi ini melakukan iterasi tunggal lagi pada k piksel untuk menghitung jumlah kuadrat selisih. Kompleksitas Waktu: $O(k)$. Kompleksitas Ruang: $O(1)$.

`calculateErrorMAD(..., k, ...)`: Mirip dengan variance, iterasi tunggal pada k piksel untuk menghitung jumlah selisih absolut. Kompleksitas Waktu: $O(k)$. Kompleksitas Ruang: $O(1)$.

`calculateErrorMaxDiff(..., k)`: Iterasi tunggal pada k piksel untuk mencari nilai min/max R, G, B. Kompleksitas Waktu: $O(k)$. Kompleksitas Ruang: $O(1)$.

`calculateEntropy(..., k)`: Iterasi tunggal pada k piksel untuk membangun histogram frekuensi (ukuran tetap 3×256). Kemudian iterasi pada histogram (ukuran tetap) untuk menghitung entropy. Didominasi oleh iterasi piksel. Kompleksitas Waktu: $O(k)$. Kompleksitas Ruang: $O(1)$ (karena ukuran histogram konstan).

`calculateSSIM(..., k, ...)`: Melakukan beberapa iterasi (sekitar 3 kali) pada k piksel untuk menghitung mean, variance, dan covariance. Kompleksitas Waktu: $O(k)$. Kompleksitas Ruang: $O(1)$.

4.2.2. Kompleksitas buildQuadtree

4.2.2.1. Analisis Waktu

Fungsi ini bersifat rekursif. Pekerjaan utama di setiap pemanggilan (untuk blok k piksel) adalah memanggil `calculateAverageColor` ($O(k)$) dan salah satu fungsi error ($O(k)$). Total pekerjaan per node adalah $O(k)$. Setiap piksel dalam gambar akan diproses di node tempat ia berada dan juga di semua node leluhurnya (ancestor).

Pada level 0 (akar), N piksel diproses. Pada level 1, total N piksel diproses ($4 * N/4$). Pada level d, total N piksel diproses di semua node pada level tersebut. Kedalaman maksimum pohon (D_{max}) bergantung pada dimensi gambar dan `minBlockSize`, tetapi umumnya sebanding dengan $O(\log N)$ (karena setiap pembagian mengurangi area sekitar 4 kali). Total waktu adalah jumlah dari $O(k)$ untuk semua node yang dibuat. Karena total piksel yang diproses di setiap level adalah N, dan ada sekitar $O(\log N)$ level dalam kasus umum/buruk, kompleksitas waktunya adalah $O(N \log N)$.

Kasus Terbaik: Gambar benar-benar seragam. Hanya node akar yang diproses secara signifikan ($O(N)$), lalu berhenti. Kompleksitas waktu $O(N)$.

Kasus Buruk: Gambar memaksa pembagian hingga `minBlockSize = 1` untuk hampir semua piksel. Jumlah node bisa mendekati $O(N)$. Kompleksitas waktu tetap cenderung $O(N \log N)$.

4.2.2.2. Analisis Ruang:

Ruang utama digunakan untuk menyimpan objek `QuadtreeNode` di heap. Jumlah node (`nodeCount`) bervariasi.

Kasus Terbaik: Hanya 1 node (akar). $O(1)$.

Kasus Buruk: Jumlah node bisa mencapai $O(N)$ jika `minBlockSize = 1` dan pembagian maksimal terjadi.

Selain itu, ada ruang tumpukan (stack) karena rekursi. Kedalaman tumpukan adalah kedalaman maksimum pohon, yaitu $O(\log N)$.

Kompleksitas ruang didominasi oleh penyimpanan node: $O(\text{Jumlah Node})$, yang bisa mencapai $O(N)$ dalam kasus terburuk.

4.2.3. Kompleksitas Keseluruhan

Secara keseluruhan, kompleksitas waktu eksekusi algoritma kompresi Quadtree dalam kode ini umumnya adalah $O(N \log N)$, di mana N adalah jumlah total piksel dalam gambar input. Batasan ini didominasi oleh fungsi `buildQuadtree`, yang dalam kasus tipikal atau terburuk perlu memproses piksel pada beberapa tingkatan kedalaman pohon yang secara logaritmik berkaitan dengan N . Meskipun dalam kasus terbaik (gambar sangat homogen) kompleksitasnya bisa $O(N)$, $O(N \log N)$ lebih mewakili kinerja pada gambar yang bervariasi. Sementara itu, kompleksitas ruang algoritma ini adalah $O(N)$, terutama karena kebutuhan untuk menyimpan data gambar input, gambar output, dan struktur data Quadtree itu sendiri, yang dalam kasus terburuk jumlah simpulnya bisa mendekati N . Kinerja aktual dalam hal waktu dan memori juga akan dipengaruhi oleh karakteristik gambar dan parameter yang dipilih oleh pengguna.

V. LAMPIRAN

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa kesalahan	✓	
2	Program berhasil dijalankan	✓	
3	Program berhasil melakukan kompresi gambar sesuai parameter yang ditentukan	✓	
4	Mengimplementasi seluruh metode perhitungan error wajib	✓	
5	[Bonus] Implementasi Structural Similarity Index (SSIM) sebagai metode pengukuran error	✓	
6	Program dan laporan dibuat (kelompok) sendiri	✓	

Tautan Repository Github:

https://github.com/aibrahim185/Tucil2_13523012_13523089

VI. DAFTAR PUSTAKA

- Munir, Rinaldi. (2025). “Algoritma Algoritma Divide and Conquer (Bagian 1)”.
Homepage Rinaldi Munir. Diakses pada 9 April 2025 melalui
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/07-Algoritma-Divide-and-Conquer-\(2025\)-Bagian1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/07-Algoritma-Divide-and-Conquer-(2025)-Bagian1.pdf)
- Munir, Rinaldi. (2025). “Algoritma Algoritma Divide and Conquer (Bagian 2)”.
Homepage Rinaldi Munir. Diakses pada 9 April 2025 melalui
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/08-Algoritma-Divide-and-Conquer-\(2025\)-Bagian2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/08-Algoritma-Divide-and-Conquer-(2025)-Bagian2.pdf)
- Munir, Rinaldi. (2025). “Algoritma Algoritma Divide and Conquer (Bagian 3)”.
Homepage Rinaldi Munir. Diakses pada 9 April 2025 melalui
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/09-Algoritma-Divide-and-Conquer-\(2025\)-Bagian3.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/09-Algoritma-Divide-and-Conquer-(2025)-Bagian3.pdf)
- Munir, Rinaldi. (2025). “Algoritma Algoritma Divide and Conquer (Bagian 4)”.
Homepage Rinaldi Munir. Diakses pada 9 April 2025 melalui
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/10-Algoritma-Divide-and-Conquer-\(2025\)-Bagian4.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/10-Algoritma-Divide-and-Conquer-(2025)-Bagian4.pdf)
- The CImg Library - C++ Template Image Processing Toolkit - Reference Documentation.
<https://cimg.eu/reference/>