

# **Laporan Tugas Kecil 2**

## **IF2211 Strategi Algoritma**

### **Kompresi Gambar dengan Metode**

### **QuadTree**



**Disusun oleh:**

Rafizan Muhammad Syawalazmi 13523034  
Aryo Wisanggeni 13523100

**Sekolah Teknik Elektro dan Informatika**  
**Institut Teknologi Bandung**  
**2025**

# Daftar Isi

<b>I. Pendahuluan</b>	<b>4</b>
<b>II. Penjelasan Algoritma</b>	<b>5</b>
A. Algoritma Divide and Conquer	5
B. Penerapan Divide and Conquer dalam Solusi	5
<b>III. Source Code</b>	<b>6</b>
A. Struktur File Program	6
B. Program Utama	7
1. Bagian Input	7
2. Bagian Processing	7
3. Bagian Output	9
C. Program Input dan Output	10
1. Bagian Input	10
2. Bagian Output	17
D. Program QuadtreeNode	18
1. Atribut	18
2. Metode	19
E. Program QuadtreeTree	23
1. Atribut	23
2. Metode	24
F. Program QuadtreeArray	28
1. Atribut	28
2. Metode	29
<b>IV. Pengujian (Input dan Output)</b>	<b>31</b>
A. Uji Normal Setiap Threshold Method	31
1. Variance PNG	31
2. Mean Absolute Deviation PNG	32
3. Max Pixel Difference PNG	33
4. Entropy PNG	34
5. Structural Similarity Index PNG	35
6. Variance JPG	36
7. Mean Absolute Deviation JPG	36
8. Max Pixel Difference JPG	38
9. Entropy JPG	39
10. Structural Similarity Index JPG	40
B. Uji Bonus Kompresi Target	41

1. Variance	41
2. Mean Absolute Deviation	42
3. Max Pixel Difference	43
4. Entropy	44
5. Structural Similarity Index	45
<b>V. Hasil Analisis Percobaan</b>	<b>46</b>
<b>VI. Implementasi Bonus</b>	<b>48</b>
A. Target Persentase Kompresi	48
B. Structural Similarity Index (SSIM)	48
C. GIF	50
<b>VI. Lampiran</b>	<b>51</b>

## I. Pendahuluan

Tugas kecil Strategi Algoritma 2025 berupa proyek pembuatan perangkat lunak berbasis CLI yang bertujuan untuk menerapkan dan juga menampilkan algoritma Divide and Conquer dalam menyelesaikan masalah. Khususnya, masalah yang ditinjau adalah mengompresi gambar. Untuk mencapai hasil pengompresian gambar dan juga penerapan Divide and Conquer, salah satu teknik yang dapat memenuhi itu adalah dengan Quadtree.

Quadtree adalah struktur data hierarkis yang digunakan untuk membagi ruang atau data menjadi bagian yang lebih kecil, yang sering digunakan dalam pengolahan gambar. Dalam konteks kompresi gambar, Quadtree membagi gambar menjadi blok-blok kecil berdasarkan keseragaman warna atau intensitas piksel. Prosesnya dimulai dengan membagi gambar menjadi empat bagian, lalu memeriksa apakah setiap bagian memiliki nilai yang seragam berdasarkan analisis sistem warna RGB, yaitu dengan membandingkan komposisi nilai merah (R), hijau (G), dan biru (B) pada piksel-piksel di dalamnya. Jika bagian tersebut tidak seragam, maka bagian tersebut akan terus dibagi hingga mencapai tingkat keseragaman tertentu atau ukuran minimum yang ditentukan.

Dalam implementasi teknis, sebuah Quadtree direpresentasikan sebagai simpul (node) dengan maksimal empat anak (children). Simpul daun (leaf) merepresentasikan area gambar yang seragam dan yang akan muncul pada produk akhir kompresi, sementara simpul internal menunjukkan area yang masih membutuhkan pembagian lebih lanjut. Setiap simpul menyimpan informasi seperti posisi (x, y), ukuran (width, height), dan nilai rata-rata warna atau intensitas piksel dalam area tersebut. Struktur ini memungkinkan pengkodean data gambar yang lebih efisien dengan menghilangkan redundansi pada area yang seragam. QuadTree sering digunakan dalam algoritma kompresi lossy karena mampu mengurangi ukuran file secara signifikan tanpa mengorbankan detail penting pada gambar.

## II. Penjelasan Algoritma

### A. Algoritma Divide and Conquer

Algoritma divide and conquer adalah paradigma algoritma yang menyelesaikan masalah dengan cara membaginya menjadi sub-masalah yang lebih kecil, menyelesaikan sub-masalah tersebut secara rekursif, lalu menggabungkan hasilnya untuk mendapatkan solusi akhir.

Tahap-tahap umum dalam algoritma divide and conquer:

1. **Divide**, yaitu membagi persoalan menjadi beberapa upapersoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil. Setiap upa-persoalan idealnya memiliki ukuran yang sama.
2. **Conquer**, yaitu menyelesaikan upa-persoalan tersebut. Jika ukurannya sudah cukup kecil, bisa diselesaikan dengan langsung. Jika tidak, bisa diselesaikan secara rekursif dengan cara yang sama
3. **Combine**, yaitu menggabungkan upa-persoalan yang sudah diselesaikan dengan upa-persoalan lainnya agar membentuk solusi akhir dari persoalan awal.

### B. Penerapan Divide and Conquer dalam Solusi

Algoritma divide and conquer untuk **proses kompresi** yang digunakan:

1. Kalkulasikan error dari blok sesuai dengan metode perhitungan variansi,
2. Untuk kasus error **lebih kecil dari threshold** variansi **atau** ukuran blok sesudah dibagi **kurang dari minimum block size**,  
**SOLVE:** normalisasi warna blok sesuai dengan rata-rata nilai RGB blok.
3. Untuk kasus lainnya,
  - (A) **DIVIDE:** Bagi blok menjadi empat sub-blok sama besar,
  - (B) **CONQUER:** Lakukan algoritma yang sama (balik ke langkah 1) untuk masing masing sub-blok.

Setelah semua blok di-SOLVE,

**Combine:** Gabungkan semua blok hasil normalisasi untuk menghasilkan gambar hasil kompresi.

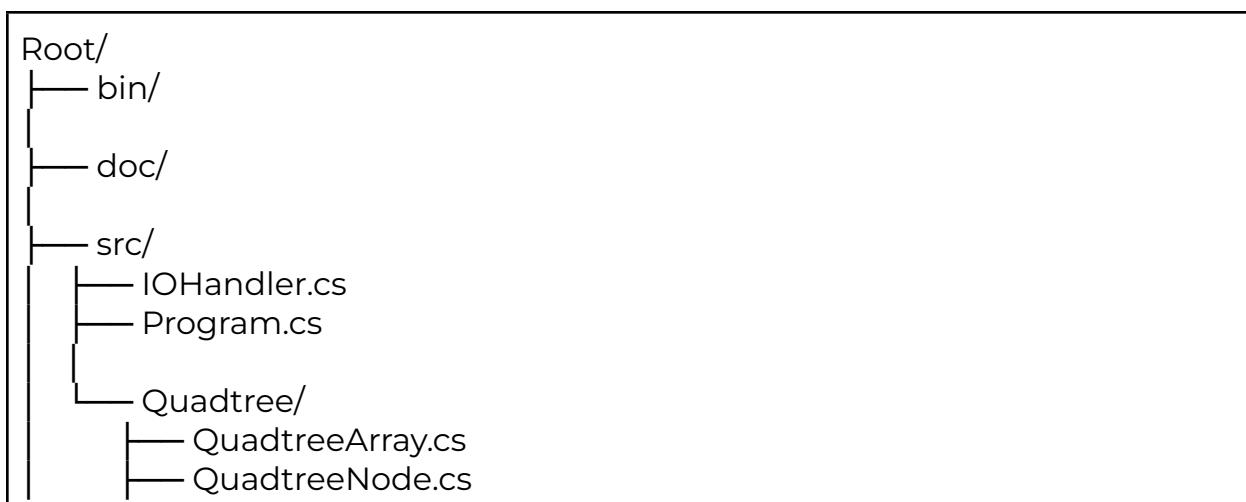
## **III. Source Code**

### **A. Struktur File Program**

Pada tugas kecil ini, program cukup sederhana sehingga tidak perlu banyak modularitas untuk file-file implementasi. Secara garis besar, terdapat empat direktori: bin, doc, src, dan test. Bin merupakan direktori yang mengandung file executable untuk menjalankan program. Doc merupakan direktori untuk menyimpan laporan tugas ini. Src merupakan direktori yang menyimpan kode sumber program. Terakhir, test merupakan direktori yang menyimpan gambar-gambar test case yang digunakan pada bab IV dan berperan dalam perkembangan program.

Pada program ini, Program.cs berperan sebagai main driver program yang memanggil modul-modul lain. IOHandler.cs, sesuai dengan namanya, adalah file yang berisi semua fungsi yang diperlukan untuk input dan output user pada program.

Satu bagian terakhir adalah namespace Quadtree yang mengandung tiga file. File pertama adalah QuadtreeNode.cs yang berisi implementasi simpul-simpul yang ada di struktur Quadtree. QuadtreeTree.cs merupakan file utama yang mengandung pohon yang dibuat dan terdiri dari simpul-simpul yang dideklarasi pada QuadtreeNode. Terakhir adalah QuadtreeArray yang merupakan struktur tambahan yang membantu dalam pembuatan bonus GIF.





## B. Program Utama

### 1. Bagian Input

```
Tcui2_13523034_13523100 - Program.cs

20 #region inputs
21     (Rgba32[,] image, long oriFileSize) = InputHandler.GetImage();
22     int errorMethod = InputHandler.GetErrorMethod();
23     float targetCompression = InputHandler.GetTargetCompression();
24
25     int minimumBlock = 1;
26     double threshold = 10;
27     if (targetCompression == 0){
28         threshold = InputHandler.GetThreshold();
29         minimumBlock = InputHandler.GetMinimumBlock();
30     }
31     string imageOutputPath = InputHandler.GetOutputPath("Masukkan alamat absolut gambar hasil kompresi: ", InputHandler.ImageExtensionType);
32     string gifOutputPath = InputHandler.GetOutputPath("Masukkan alamat absolut gif hasil (.gif): ", ".gif");
33
34     Console.Clear();
35     InputHandler.ShowInputStatus();
36
37 #endregion
```

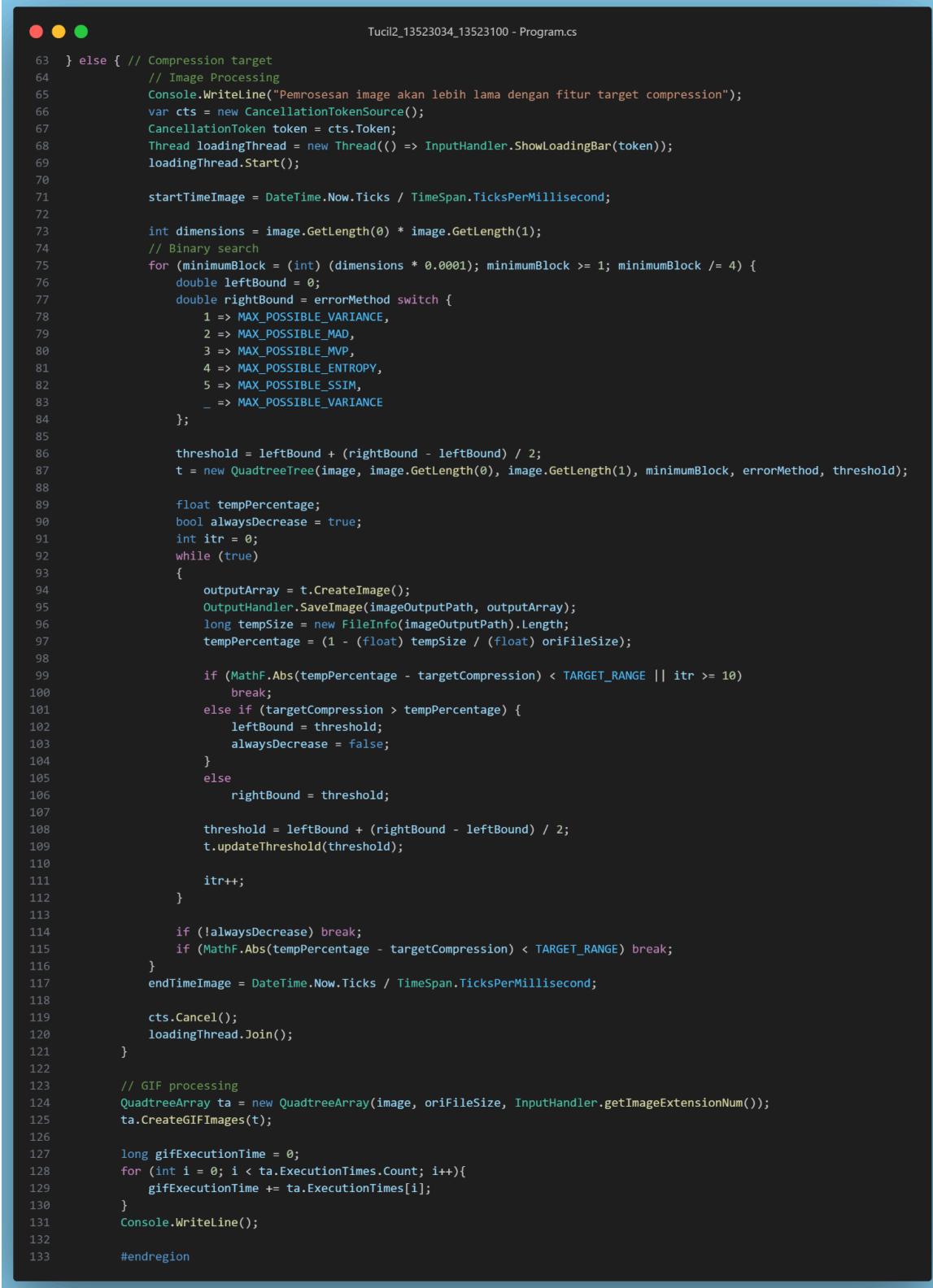
Gambar 3.B.1. Input program utama

### 2. Bagian Processing

```
Tcui2_13523034_13523100 - Program.cs

39 #region processing
40     ConsoleColor foregroundColor = ConsoleColor.Cyan;
41     Console.WriteLine("    PROCESSING");
42     Console.ResetColor();
43
44     long startTimeImage;
45     long endTimeImage;
46     QuadtreeTree t = null!;
47     Rgba32[,] outputArray = null!;
48
49     if (targetCompression == 0){ // NO compression target
50         // Image processing
51         var cts = new CancellationTokenSource();
52         CancellationToken token = cts.Token;
53         Thread loadingThread = new Thread(() => InputHandler.ShowLoadingBar(token));
54         loadingThread.Start();
55
56         startTimeImage = DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond;
57         t = new QuadtreeTree(image, image.GetLength(0), image.GetLength(1), minimumBlock, errorMethod, threshold);
58         outputArray = t.CreateImage();
59         endTimeImage = DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond;
60
61         cts.Cancel();
62         loadingThread.Join();
```

Gambar 3.B.2. Processing program utama tanpa targetting

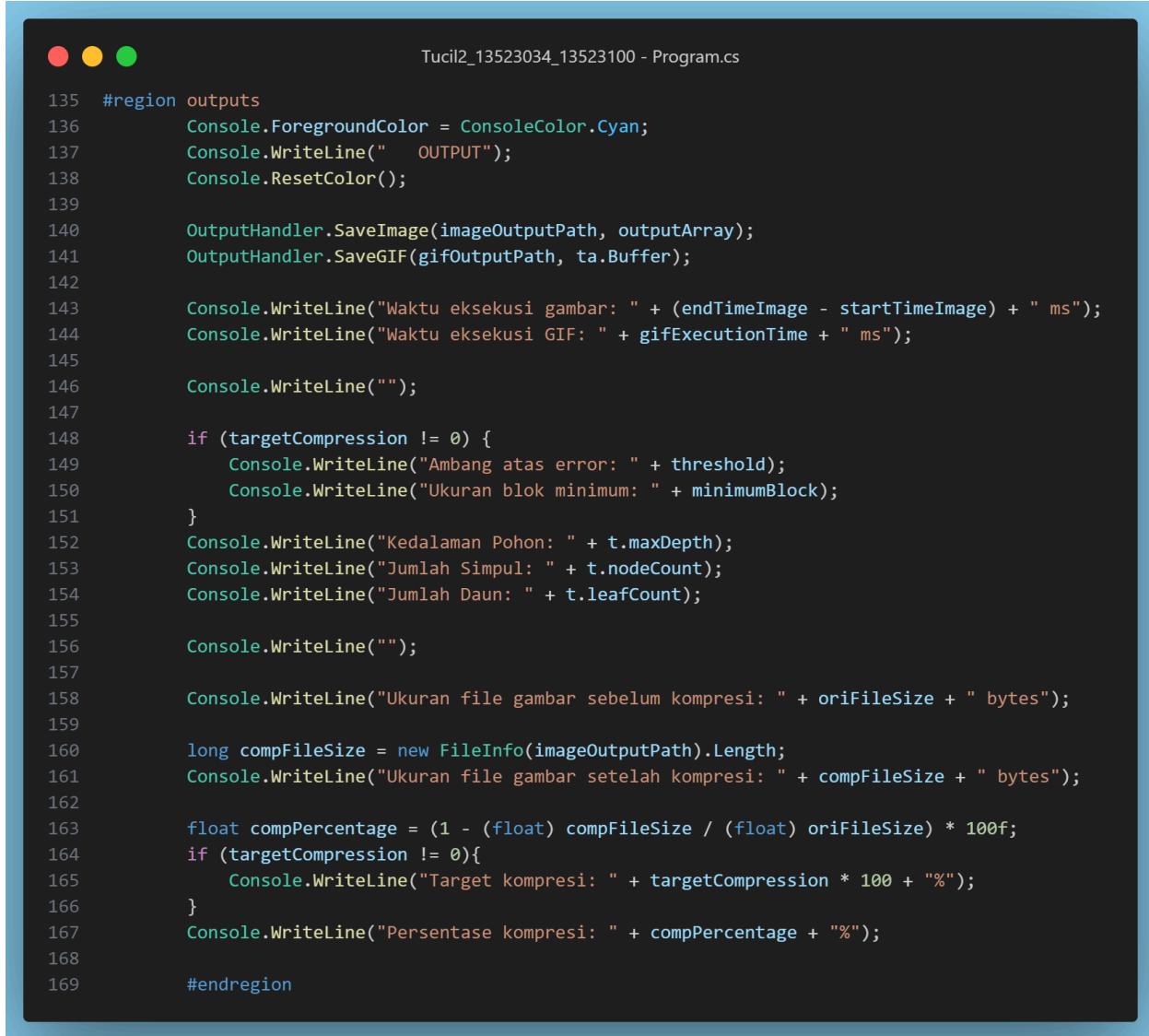


```
Tucil2_13523034_13523100 - Program.cs

63 } else { // Compression target
64     // Image Processing
65     Console.WriteLine("Pemrosesan image akan lebih lama dengan fitur target compression");
66     var cts = new CancellationTokenSource();
67     CancellationToken token = cts.Token;
68     Thread loadingThread = new Thread(() => InputHandler.ShowLoadingBar(token));
69     loadingThread.Start();
70
71     startTimeImage = DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond;
72
73     int dimensions = image.GetLength(0) * image.GetLength(1);
74     // Binary search
75     for (minimumBlock = (int) (dimensions * 0.0001); minimumBlock >= 1; minimumBlock /= 4) {
76         double leftBound = 0;
77         double rightBound = errorMethod switch {
78             1 => MAX_POSSIBLE_VARIANCE,
79             2 => MAX_POSSIBLE_MAD,
80             3 => MAX_POSSIBLE_MVP,
81             4 => MAX_POSSIBLE_ENTROPY,
82             5 => MAX_POSSIBLE_SSIM,
83             _ => MAX_POSSIBLE_VARIANCE
84         };
85
86         threshold = leftBound + (rightBound - leftBound) / 2;
87         t = new QuadtreeTree(image, image.GetLength(0), image.GetLength(1), minimumBlock, errorMethod, threshold);
88
89         float tempPercentage;
90         bool alwaysDecrease = true;
91         int itr = 0;
92         while (true)
93         {
94             outputArray = t.CreateImage();
95             OutputHandler.SaveImage(imageOutputPath, outputArray);
96             long tempSize = new FileInfo(imageOutputPath).Length;
97             tempPercentage = (1 - (float) tempSize / (float) oriFileSize);
98
99             if (MathF.Abs(tempPercentage - targetCompression) < TARGET_RANGE || itr >= 10)
100                 break;
101             else if (targetCompression > tempPercentage) {
102                 leftBound = threshold;
103                 alwaysDecrease = false;
104             }
105             else
106                 rightBound = threshold;
107
108             threshold = leftBound + (rightBound - leftBound) / 2;
109             t.updateThreshold(threshold);
110
111             itr++;
112         }
113
114         if (!alwaysDecrease) break;
115         if (MathF.Abs(tempPercentage - targetCompression) < TARGET_RANGE) break;
116     }
117     endTimeImage = DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond;
118
119     cts.Cancel();
120     loadingThread.Join();
121 }
122
123 // GIF processing
124 QuadtreeArray ta = new QuadtreeArray(image, oriFileSize, InputHandler.getImageExtensionNum());
125 ta.CreateGIFImages(t);
126
127 long gifExecutionTime = 0;
128 for (int i = 0; i < ta.ExecutionTimes.Count; i++){
129     gifExecutionTime += ta.ExecutionTimes[i];
130 }
131 Console.WriteLine();
132
133 #endregion
```

Gambar 3.B.3. Processing program utama compression target

### 3. Bagian Output



The screenshot shows a terminal window titled "Tucil2\_13523034\_13523100 - Program.cs". The code is a region named "outputs" containing various console output statements. It includes saving images and GIFs, printing execution times, and displaying tree statistics like depth, node count, and leaf count. It also compares file sizes before and after compression and calculates the compression percentage.

```
135 #region outputs
136     Console.ForegroundColor = ConsoleColor.Cyan;
137     Console.WriteLine("    OUTPUT");
138     Console.ResetColor();
139
140     OutputHandler.SaveImage(imageOutputPath, outputArray);
141     OutputHandler.SaveGIF(gifOutputPath, ta.Buffer);
142
143     Console.WriteLine("Waktu eksekusi gambar: " + (endTimeImage - startTimeImage) + " ms");
144     Console.WriteLine("Waktu eksekusi GIF: " + gifExecutionTime + " ms");
145
146     Console.WriteLine("");
147
148     if (targetCompression != 0) {
149         Console.WriteLine("Ambang atas error: " + threshold);
150         Console.WriteLine("Ukuran blok minimum: " + minimumBlock);
151     }
152     Console.WriteLine("Kedalaman Pohon: " + t.maxDepth);
153     Console.WriteLine("Jumlah Simpul: " + t.nodeCount);
154     Console.WriteLine("Jumlah Daun: " + t.leafCount);
155
156     Console.WriteLine("");
157
158     Console.WriteLine("Ukuran file gambar sebelum kompresi: " + oriFileSize + " bytes");
159
160     long compFileSize = new FileInfo(imageOutputPath).Length;
161     Console.WriteLine("Ukuran file gambar setelah kompresi: " + compFileSize + " bytes");
162
163     float compPercentage = (1 - (float) compFileSize / (float) oriFileSize) * 100f;
164     if (targetCompression != 0){
165         Console.WriteLine("Target kompresi: " + targetCompression * 100 + "%");
166     }
167     Console.WriteLine("Persentase kompresi: " + compPercentage + "%");
168
169 #endregion
```

Gambar 3.B.4. Output program utama

## C. Program Input dan Output

### 1. Bagian Input

```
12 references | Windsurf: Refactor | Explain
public static class InputHandler
{
    33 references
    public static string InputError = "";
    4 references
    public static string InputAddress = "";
    7 references
    public static string ErrorThresholdMethod = "";
    5 references
    public static string ImageExtensionType = "";
    3 references
    public static double ErrorThreshold = -1;
    3 references
    public static int MinimumBlock = -1;
    5 references
    public static float CompressionRateTarget = -1;
    3 references
    public static string ImageOutputAddress = "";
    3 references
    public static string GifOutputAddress = "";
    0 references
    public static float MinTargetThreshold = -1;
    0 references
    public static float MaxTargetThreshold = -1;
    1 reference | Windsurf: Refactor | Explain | Generate Documentation | X
    public static (Rgba32[,], long) GetImage()
    {
        string? absolutePath;
        while (true)
        {
            Console.Clear();
            ShowInputStatus();
            if (InputError != ""){
                ShowInputError();
            }

            Console.WriteLine("Masukkan alamat absolut gambar yang akan dikompresi: ");
        }
    }
}
```

Gambar 3.C.1. Input source code (1)

```

        absolutePath = Console.ReadLine()?.Trim();

        if (absolutePath == null)
        {
            InputError = "Input tidak valid, harap masukkan alamat yang valid.";
            continue;
        }

        try
        {
            if (!File.Exists(absolutePath))
            {
                InputError = "Gambar tidak ditemukan!";
                continue;
            }

            string extension = Path.GetExtension(absolutePath).ToLower();
            if (extension != ".png" && extension == ".jpg" && extension == ".jpeg")
            {
                InputError = "Format file tidak didukung! Harap gunakan .png, .jpg, atau .jpeg.";
                continue;
            }
            ImageExtensionType = extension;
        }

        using (Image<Rgba32> image = Image.Load<Rgba32>(absolutePath))
        {
            int width = image.Width;
            int height = image.Height;
            Rgba32[,] pixelMatrix = new Rgba32[width, height];

            for (int x = 0; x < width; x++)
            {
                for (int y = 0; y < height; y++)
                {

```

Gambar 3.C.2. Input source code (2)

```

                    pixelMatrix[x, y] = image[x,y];
                }
            }

            InputError = "";
            InputAddress = absolutePath;

            return (pixelMatrix, new FileInfo(absolutePath).Length);
        }
    }
    catch
    {
        InputError = "Input tidak valid, harap masukkan alamat yang valid.";
        continue;
    }
}

1 reference | Windsurf: Refactor | Explain | Generate Documentation | X
public static int GetErrorMethod()
{
    int value;
    while (true)
    {
        Console.Clear();
        ShowInputStatus();

        Console.ForegroundColor = ConsoleColor.Cyan;
        Console.WriteLine("  ERROR METHODS");
        Console.ResetColor();
        Console.WriteLine("1. Variance");
        Console.WriteLine("2. Mean Absolute Deviation (MAD)");
        Console.WriteLine("3. Max Pixel Difference");
        Console.WriteLine("4. Entropy");
        Console.WriteLine("5. Structural Similarity Index (SSIM)");

```

Gambar 3.C.3. Input source code (3)

```

        Console.WriteLine("");
        if (InputError != ""){
            ShowInputError();
        }
        Console.Write("Pilih metode pengukuran error yang diinginkan [1-5]: ");
        string? input = Console.ReadLine()?.Trim();

        if (input == null)
        {
            InputError = "Input tidak valid, harap masukkan angka yang valid.";
            continue;
        }

        try
        {
            value = Convert.ToInt32(input);

            switch (value)
            {
                case 1: ErrorThresholdMethod = "Variance"; break;
                case 2: ErrorThresholdMethod = "Mean Absolute Deviation (MAD)"; break;
                case 3: ErrorThresholdMethod = "Max Pixel Difference"; break;
                case 4: ErrorThresholdMethod = "Entropy"; break;
                case 5: ErrorThresholdMethod = "Structural Similarity Index (SSIM)"; break;
                default:
                    InputError = "Input tidak valid, harap masukkan angka bulat 1-5.";
                    continue;
            }

            InputError = "";
            return value;
        }
        catch
    }

```

Gambar 3.C.4. Input source code (4)

```

        InputError = "Input tidak valid, harap masukkan angka yang valid dalam range 1-5.";
        continue;
    }
}

1 reference | Windsurf: Refactor | Explain | Generate Documentation | X
public static double GetThreshold()
{
    double value;

    while (true)
    {
        Console.Clear();
        ShowInputStatus();
        if (InputError != ""){
            ShowInputError();
        }

        Console.Write("Masukkan ambang atas error: ");
        string? input = Console.ReadLine()?.Trim();

        if (input == null)
        {
            InputError = "Input tidak valid, harap masukkan angka yang valid.";
            continue;
        }

        try
        {
            value = Convert.ToDouble(input);
            if (value < 0)

```

Gambar 3.C.5. Input source code (5)

```

        {
            InputError = "Input tidak valid, harap masukkan angka >= 0.";
            continue;
        }

        ErrorThreshold = value;
        InputError = "";

        return value;
    }
    catch
    {
        InputError = "Input tidak valid, harap masukkan angka yang valid.";
        continue;
    }
}

1 reference | Windsurf: Refactor | Explain | Generate Documentation | X
public static int GetMinimumBlock()
{
    int value;

    while (true)
    {
        Console.Clear();
        ShowInputStatus();
        if (InputError != ""){
            ShowInputError();
        }

        Console.Write("Masukkan ukuran blok minimum: ");
        string? input = Console.ReadLine()?.Trim();

        if (input == null)

```

Gambar 3.C.6. Input source code (6)

```

        {
            InputError = "Input tidak valid, harap masukkan angka yang valid.";
            continue;
        }

        try
        {
            value = Convert.ToInt32(input);

            MinimumBlock = value;
            InputError = "";

            return value;
        }
        catch
        {
            InputError = "Input tidak valid, harap masukkan angka yang valid.";
            continue;
        }
    }

1 reference | Windsurf: Refactor | Explain | Generate Documentation | X
public static float GetTargetCompression()
{
    float value;

    while (true)
    {
        Console.Clear();
        ShowInputStatus();

        Console.ForegroundColor = ConsoleColor.Cyan;
        Console.WriteLine("    NILAI PERSENTASI KOMPRESI YANG VALID");

```

Gambar 3.C.7. Input source code (7)

```

Console.ResetColor();
Console.WriteLine("Inputlah dalam range 0-1 dengan 0 berarti mematikan target kompresi.");
Console.WriteLine("Walau pun range 0 hingga 1, terdapat range yang tidak dapat dicapai oleh kompresi.");
Console.WriteLine("Dalam kasus tersebut, program akan menampilkan nilai minimal/maximal yang mendekati nilai tersebut.");
Console.ForegroundColor = ConsoleColor.Red;
Console.WriteLine("Pakai koma (,) jika titik (.) tidak bisa dan sebaliknya");
Console.ResetColor();
Console.WriteLine("");

if (InputError != ""){
    ShowInputError();
}
Console.Write("Masukkan target persentase kompresi: ");
string? input = Console.ReadLine()?.Trim();

if (input == null)
{
    InputError = "Input tidak valid, harap masukkan angka yang valid.";
    continue;
}

try
{
    value = Convert.ToSingle(input);

    if (value < 0 || value > 1){
        InputError = "Input tidak valid, tidak dalam range 0-1";
        continue;
    }

    InputError = "";
    CompressionRateTarget = value;
}

return value;

```

Gambar 3.C.8. Input source code (8)

```

}
catch
{
    InputError = "Input tidak valid, harap angka yang valid dalam range 0-1.";
    continue;
}
}

2 references | Windsurf: Refactor | Explain | Generate Documentation | X
public static string GetOutputPath(string message, string extension)
{
    string? absolutePath;
    while (true)
    {
        Console.Clear();
        ShowInputstatus();
        if (InputError != ""){
            ShowInputError();
        }

        Console.Write(message);
        absolutePath = Console.ReadLine();

        try
        {
            if (absolutePath == null)
            {
                InputError = "Input tidak valid, harap masukkan alamat yang valid.";
                continue;
            }
        }
    }
}

```

Gambar 3.C.9. Input source code (9)

```

        if (!Path.HasExtension(absolutePath))
        {
            absolutePath += extension;
        }
        else if (!absolutePath.EndsWith(extension, StringComparison.OrdinalIgnoreCase))
        {
            InputError = "Ekstensi file tidak sesuai! Pastikan berekstensi " + extension;
            continue;
        }

        string? directory = Path.GetDirectoryNameAbsolutePath);

        if (AbsolutePath == InputAddress){
            InputError = "Alamat output tidak boleh sama dengan alamat input, silakan input lagi!";
            continue;
        }

        if (directory == null || !Directory.Exists(directory))
        {
            InputError = "Direktori tidak ditemukan!";
            continue;
        }

        if (File.ExistsAbsolutePath))
        {
            Console.WriteLine("File sudah ada! Apakah ingin mengganti file yang sudah ada (y/n)? ");
            string? response = Console.ReadLine()?.Trim().ToLower();

            if (response != "y")
            {
                continue;
            }
        }
    }
}

```

Gambar 3.C.10. Input source code (10)

```

InputError = "";
if (extension == ".jpg" || extension == ".jpeg" || extension == ".png"){
    ImageOutputAddress = absolutePath;
} else if (extension == ".gif"){
    GIFOutputAddress = absolutePath;
}

return absolutePath;
}
catch
{
    InputError = "Input tidak valid, harap masukkan alamat yang valid.";
    continue;
}
}

7 references | Windsurf: Refactor | Explain | Generate Documentation | X
public static void ShowInputStatus()
{
    Console.ForegroundColor = ConsoleColor.Cyan;
    Console.WriteLine(" STATUS INPUT");
    Console.ResetColor();

    Console.WriteLine($"1. Alamat gambar : {((string.IsNullOrEmpty(InputAddress) ? "[belum diisi]" : InputAddress))}");
    Console.WriteLine($"2. Metode error : {((string.IsNullOrEmpty(ErrorThresholdMethod) ? "[belum diisi]" : ErrorThresholdMethod))}");
    Console.WriteLine($"3. Target kompresi : {((CompressionRateTarget < 0 ? "[belum diisi]" : CompressionRateTarget.ToString()))}");
    Console.WriteLine($"4. Threshold error : {((CompressionRateTarget > 0 ? "XXX" : (ErrorThreshold < 0 ? "[belum diisi]" : ErrorThreshold))}");
    Console.WriteLine($"5. Minimum block : {((CompressionRateTarget > 0 ? "XXX" : (MinimumBlock < 0 ? "[belum diisi]" : MinimumBlock))}");
    Console.WriteLine($"6. Output image path : {((string.IsNullOrEmpty(ImageOutputAddress) ? "[belum diisi]" : ImageOutputAddress))}");
    Console.WriteLine($"7. Output GIF path : {((string.IsNullOrEmpty(GIFOutputAddress) ? "[belum diisi]" : GIFOutputAddress))}");
    Console.WriteLine();
}

```

Gambar 3.C.11. Input source code (11)

```
6 references | Windsurf: Refactor | Explain | Generate Documentation | X
public static void ShowInputError()
{
    Console.WriteLine(InputError);
}
2 references | Windsurf: Refactor | Explain | Generate Documentation | X
public static void ShowLoadingBar(CancellationToken token)
{
    int current = 1;
    int total = 50;

    while (!token.IsCancellationRequested)
    {
        string bar = new string('■', current);
        Console.WriteLine($"\\rCurrently processing image: [{bar.PadRight(total)}]");

        current++;
        if (current > total)
            current = 1;

        Thread.Sleep(100);
    }

    Console.Write("\r" + new string(' ', 100) + "\r");
    Console.ForegroundColor = ConsoleColor.Green;
    Console.WriteLine("Image processing finished!");
    Console.ResetColor();
    Console.WriteLine("");
}
1 reference | Windsurf: Refactor | Explain | Generate Documentation | X
public static int getImageExtensionNum(){
    if (ImageExtensionType == ".jpeg" || ImageExtensionType == ".jpg"){
        return 2;
    }
    else if (ImageExtensionType == ".png"){
        return 1;
    } else {
        return -1;
    }
}
```

Gambar 3.C.12. Input source code (12)

```
} else if (ImageExtensionType == ".png"){
    return 1;
} else {
    return -1;
}
```

Gambar 3.C.13. Input source code (13)

## 2. Bagian Output

```
3 references | Windsurf: Refactor | Explain
public static class OutputHandler
{
    2 references | Windsurf: Refactor | Explain | Generate Documentation | X
    public static void SaveImage(string outputPath, Rgba32[,] pixelMatrix)
    {
        using (Image<Rgba32> image = new Image<Rgba32>(pixelMatrix.GetLength(0), pixelMatrix.GetLength(1)))
        {
            for (int x = 0; x < pixelMatrix.GetLength(0); x++)
            {
                for (int y = 0; y < pixelMatrix.GetLength(1); y++)
                {
                    image[x, y] = pixelMatrix[x, y];
                }
            }

            image.Save(outputPath);
        }
    }

    1 reference | Windsurf: Refactor | Explain | Generate Documentation | X
    public static void SaveGIF(string outputPath, List<Rgba32[,]> frameMatrices, int frameDelay = 50, int repeatCount = 0)
    {
        if (frameMatrices == null || frameMatrices.Count == 0)
            throw new ArgumentException("Frame list is empty.");

        using var gif = MatrixToImage(frameMatrices[0]);

        for (int i = 1; i < frameMatrices.Count; i++)
        {
            using var frameImage = MatrixToImage(frameMatrices[i]);
            gif.Frames.AddFrame(frameImage.Frames.RootFrame);
        }
    }
}
```

Gambar 3.C.14. Output source code (1)

```
foreach (var frame in gif.Frames)
{
    frame.Metadata.GetGifMetadata().FrameDelay = frameDelay;
}

gif.Metadata.GetGifMetadata().RepeatCount = (ushort)repeatCount;
gif.Save(outputPath);
}

2 references | Windsurf: Refactor | Explain | Generate Documentation | X
private static Image<Rgba32> MatrixToImage(Rgba32[,] matrix)
{
    int width = matrix.GetLength(0);
    int height = matrix.GetLength(1);
    var image = new Image<Rgba32>(width, height);

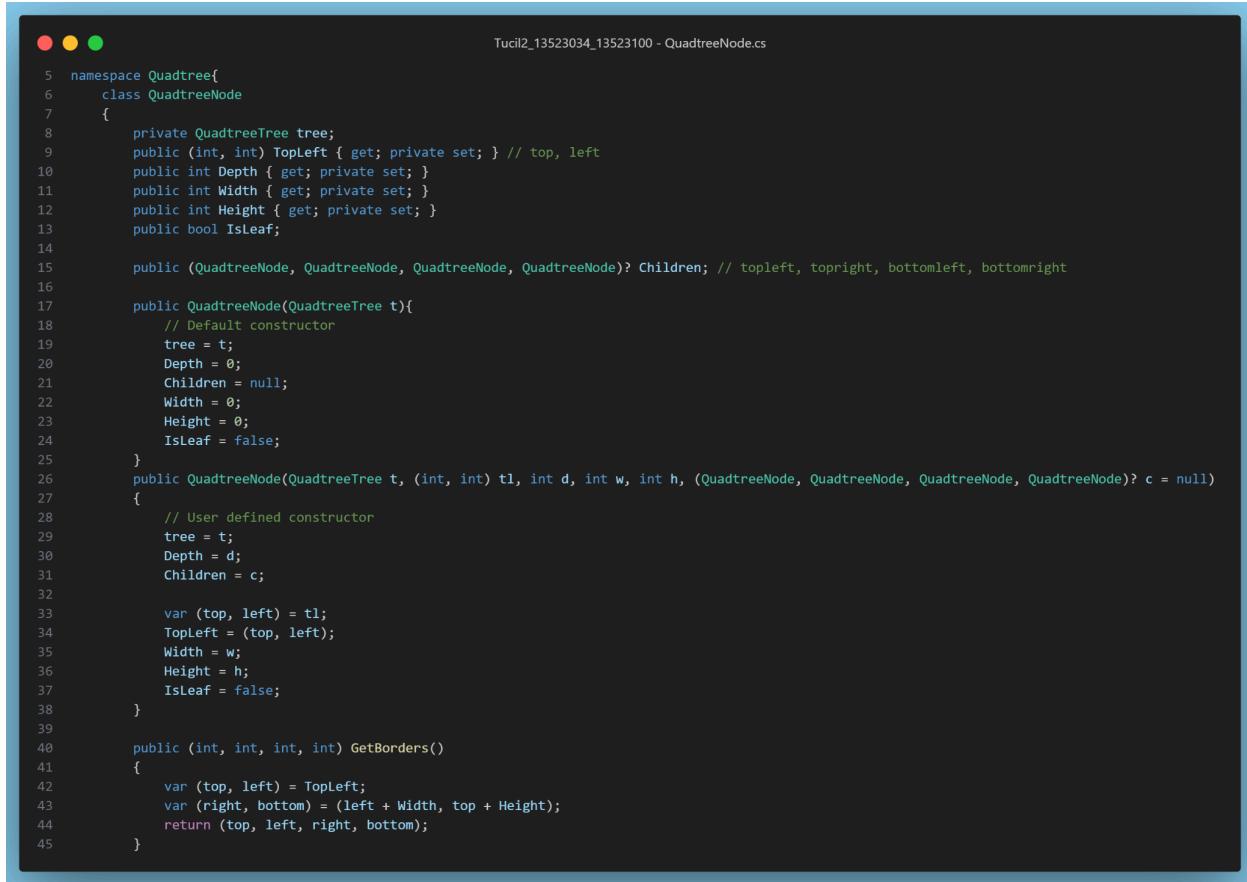
    for (int x = 0; x < width; x++)
    {
        for (int y = 0; y < height; y++)
        {
            image[x, y] = matrix[x, y];
        }
    }

    return image;
}
```

Gambar 3.C.15. Output source code (2)

## D. Program QuadtreeNode

### 1. Atribut

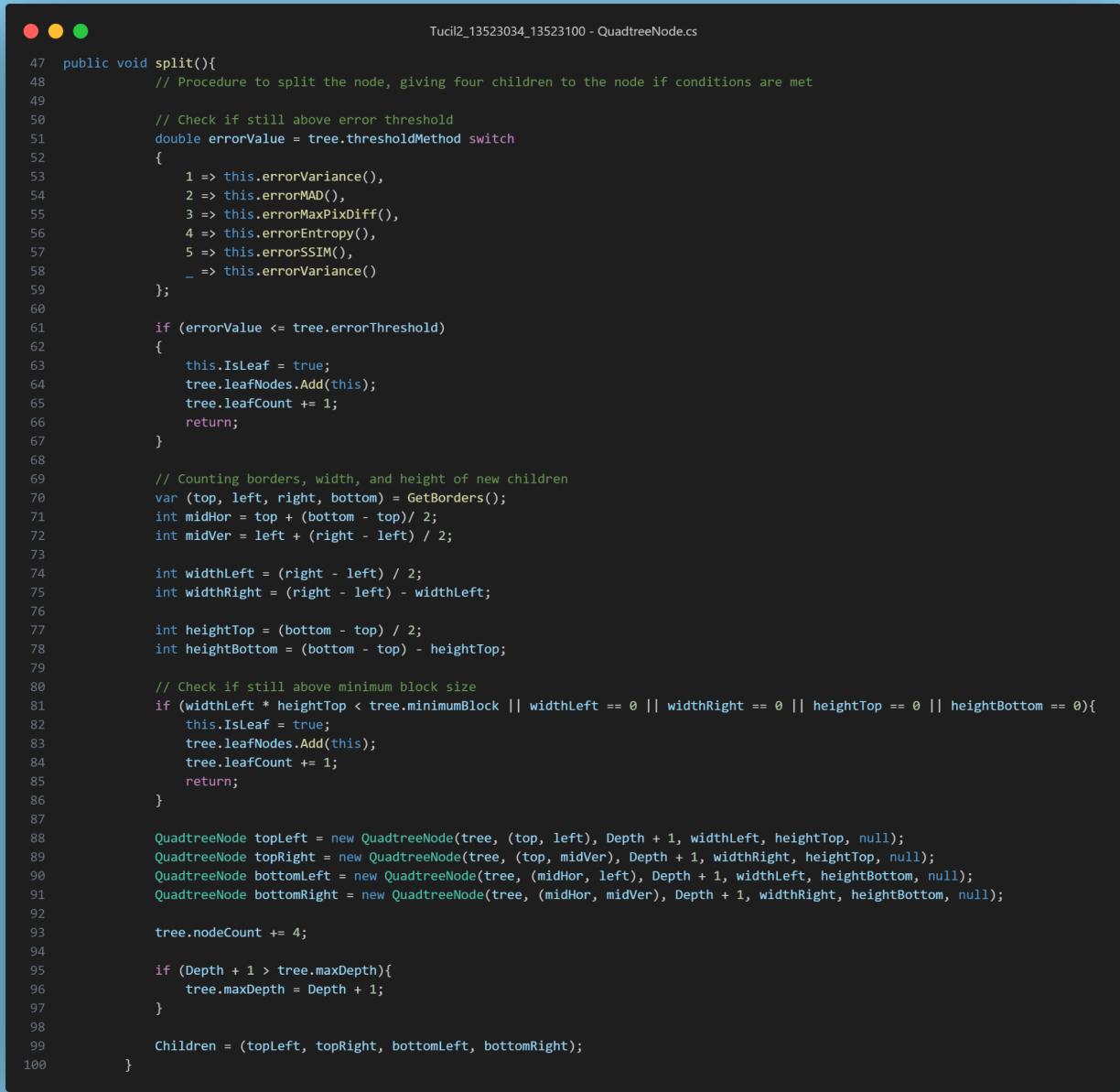


The screenshot shows a code editor window with a dark theme. The title bar reads "TcuiI2\_13523034\_13523100 - QuadtreeNode.cs". The code is written in C# and defines a class named QuadtreeNode. The class has several properties and methods. Properties include TopLeft, Depth, Width, Height, and IsLeaf. Methods include a constructor that takes a QuadtreeTree object, a user-defined constructor that takes coordinates (t1, d), width (w), height (h), and children (c), and a GetBorders() method that returns the top-left and bottom-right coordinates of the node's bounding box.

```
5  namespace Quadtree{
6      class QuadtreeNode{
7      {
8          private QuadtreeTree tree;
9          public (int, int) TopLeft { get; private set; } // top, left
10         public int Depth { get; private set; }
11         public int Width { get; private set; }
12         public int Height { get; private set; }
13         public bool IsLeaf;
14
15         public (QuadtreeNode, QuadtreeNode, QuadtreeNode, QuadtreeNode)? Children; // topleft, topright, bottomleft, bottomright
16
17         public QuadtreeNode(QuadtreeTree t){
18             // Default constructor
19             tree = t;
20             Depth = 0;
21             Children = null;
22             Width = 0;
23             Height = 0;
24             IsLeaf = false;
25         }
26         public QuadtreeNode(QuadtreeTree t, (int, int) tl, int d, int w, int h, (QuadtreeNode, QuadtreeNode, QuadtreeNode, QuadtreeNode)? c = null)
27         {
28             // User defined constructor
29             tree = t;
30             Depth = d;
31             Children = c;
32
33             var (top, left) = tl;
34             TopLeft = (top, left);
35             Width = w;
36             Height = h;
37             IsLeaf = false;
38         }
39
40         public (int, int, int, int) GetBorders()
41         {
42             var (top, left) = TopLeft;
43             var (right, bottom) = (left + Width, top + Height);
44             return (top, left, right, bottom);
45         }
46     }
```

Gambar 3.D.1. Atribut QuadtreeNode

## 2. Metode



```
47 public void split(){
48     // Procedure to split the node, giving four children to the node if conditions are met
49
50     // Check if still above error threshold
51     double errorValue = tree.thresholdMethod switch
52     {
53         1 => this.errorVariance(),
54         2 => this.errorMAD(),
55         3 => this.errorMaxPixDiff(),
56         4 => this.errorEntropy(),
57         5 => this.errorSSIM(),
58         _ => this.errorVariance()
59     };
60
61     if (errorValue <= tree.errorThreshold)
62     {
63         this.IsLeaf = true;
64         tree.leafNodes.Add(this);
65         tree.leafCount += 1;
66         return;
67     }
68
69     // Counting borders, width, and height of new children
70     var (top, left, right, bottom) = GetBorders();
71     int midHor = top + (bottom - top)/ 2;
72     int midVer = left + (right - left) / 2;
73
74     int widthLeft = (right - left) / 2;
75     int widthRight = (right - left) - widthLeft;
76
77     int heightTop = (bottom - top) / 2;
78     int heightBottom = (bottom - top) - heightTop;
79
80     // Check if still above minimum block size
81     if (widthLeft * heightTop < tree.minimumBlock || widthLeft == 0 || widthRight == 0 || heightTop == 0 || heightBottom == 0){
82         this.IsLeaf = true;
83         tree.leafNodes.Add(this);
84         tree.leafCount += 1;
85         return;
86     }
87
88     QuadtreeNode topLeft = new QuadtreeNode(tree, (top, left), Depth + 1, widthLeft, heightTop, null);
89     QuadtreeNode topRight = new QuadtreeNode(tree, (top, midVer), Depth + 1, widthRight, heightTop, null);
90     QuadtreeNode bottomLeft = new QuadtreeNode(tree, (midHor, left), Depth + 1, widthLeft, heightBottom, null);
91     QuadtreeNode bottomRight = new QuadtreeNode(tree, (midHor, midVer), Depth + 1, widthRight, heightBottom, null);
92
93     tree.nodeCount += 4;
94
95     if (Depth + 1 > tree.maxDepth){
96         tree.maxDepth = Depth + 1;
97     }
98
99     Children = (topLeft, topRight, bottomLeft, bottomRight);
100 }
```

Gambar 3.D.2. Metode QuadtreeNode (1)

```

102    public void merge(){
103        // Check if still above error threshold
104        double errorValue = tree.thresholdMethod switch
105        {
106            1 => this.errorVariance(),
107            2 => this.errorMAD(),
108            3 => this.errorMaxpixDiff(),
109            4 => this.errorEntropy(),
110            5 => this.errorSSIM(),
111            _ => this.errorVariance()
112        };
113
114        if (errorValue <= tree.errorThreshold)
115        {
116            this.removeChildren();
117            this.IsLeaf = true;
118            tree.leafNodes.Add(this);
119            tree.leafCount += 1;
120            return;
121        }
122    }
123
124    public void removeChildren(){
125        // Procedure to remove children of a node
126        if (this.IsLeaf || this.Children == null) {
127            tree.leafNodes.Remove(this);
128            tree.leafCount -= 1;
129            return;
130        }
131
132        (QuadtreeNode topLeft, QuadtreeNode topRight, QuadtreeNode bottomLeft, QuadtreeNode bottomRight) = this.Children.Value;
133        topLeft.removeChildren();
134        topRight.removeChildren();
135        bottomLeft.removeChildren();
136        bottomRight.removeChildren();
137        this.Children = null;
138    }
139
140    public double errorVariance(){
141        // Error measurement using Variance of the three color channels
142        double N = Width * Height;
143        var (meanR, meanG, meanB) = colorMean();
144
145        double varianceR = 0;
146        double varianceG = 0;
147        double varianceB = 0;
148
149        for (int i = 0; i < Width; i++){
150            for (int j = 0; j < Height; j++){
151                varianceR += (tree.GetPixel(i, j, TopLeft).R - meanR) * (tree.GetPixel(i, j, TopLeft).R - meanR);
152                varianceG += (tree.GetPixel(i, j, TopLeft).G - meanG) * (tree.GetPixel(i, j, TopLeft).G - meanG);
153                varianceB += (tree.GetPixel(i, j, TopLeft).B - meanB) * (tree.GetPixel(i, j, TopLeft).B - meanB);
154            }
155        }
156
157        varianceR = varianceR / N;
158        varianceG = varianceG / N;
159        varianceB = varianceB / N;
160
161        double result = (varianceR + varianceG + varianceB) / 3;
162        return result;
163    }

```

Gambar 3.D.2. Metode QuadtreeNode (2)

```
165 public double errorMAD(){
166     // Error measurement using Mean Absolute Deviation of the three color channels
167     double N = Width * Height;
168     var (meanR, meanG, meanB) = colorMean();
169
170     double madR = 0;
171     double madG = 0;
172     double madB = 0;
173
174     for (int i = 0; i < Width; i++){
175         for (int j = 0; j < Height; j++){
176             madR += Math.Abs(tree.GetPixel(i, j, TopLeft).R - meanR);
177             madG += Math.Abs(tree.GetPixel(i, j, TopLeft).G - meanG);
178             madB += Math.Abs(tree.GetPixel(i, j, TopLeft).B - meanB);
179         }
180     }
181
182     madR = madR / N;
183     madG = madG / N;
184     madB = madB / N;
185
186     double result = (madR + madG + madB) / 3;
187     return result;
188 }
189
190 public double errorMaxPixDiff(){
191     // Error measurement using Max Pixel Difference of the three color channels
192     double maxR = tree.GetPixel(0, 0, TopLeft).R;
193     double maxG = tree.GetPixel(0, 0, TopLeft).G;
194     double maxB = tree.GetPixel(0, 0, TopLeft).B;
195
196     double minR = tree.GetPixel(0, 0, TopLeft).R;
197     double minG = tree.GetPixel(0, 0, TopLeft).G;
198     double minB = tree.GetPixel(0, 0, TopLeft).B;
199
200     for (int i = 0; i < Width; i++){
201         for (int j = 0; j < Height; j++){
202             var pixel = tree.GetPixel(i, j, TopLeft);
203
204             if (maxR < pixel.R) maxR = pixel.R;
205             if (maxG < pixel.G) maxG = pixel.G;
206             if (maxB < pixel.B) maxB = pixel.B;
207
208             if (minR > pixel.R) minR = pixel.R;
209             if (minG > pixel.G) minG = pixel.G;
210             if (minB > pixel.B) minB = pixel.B;
211         }
212     }
213
214     double diffR = maxR - minR;
215     double diffG = maxG - minG;
216     double diffB = maxB - minB;
217
218     double result = (diffR + diffG + diffB) / 3;
219     return result;
220 }
221
```

Gambar 3.D.2. Metode QuadtreeNode (2)

```
222 public double errorEntropy(){
223     // Error measurement using entropy of the three color channels
224     double N = Width * Height;
225
226     int[] redCount = new int[256];
227     int[] greenCount = new int[256];
228     int[] blueCount = new int[256];
229
230     for (int i = 0; i < Width; i++){
231         for (int j = 0; j < Height; j++){
232             redCount[tree.GetPixel(i, j, TopLeft).R] += 1;
233             greenCount[tree.GetPixel(i, j, TopLeft).G] += 1;
234             blueCount[tree.GetPixel(i, j, TopLeft).B] += 1;
235         }
236     }
237
238     double redEntropy = 0;
239     double greenEntropy = 0;
240     double blueEntropy = 0;
241
242     for (int i = 0; i < 256; i++){
243         if (redCount[i] != 0){
244             double probI = redCount[i] / N;
245             redEntropy += (probI) * (Math.Log(probI) / Math.Log(2));
246         }
247         if (greenCount[i] != 0){
248             double probI = greenCount[i] / N;
249             greenEntropy += (probI) * (Math.Log(probI) / Math.Log(2));
250         }
251         if (blueCount[i] != 0){
252             double probI = blueCount[i] / N;
253             blueEntropy += (probI) * (Math.Log(probI) / Math.Log(2));
254         }
255     }
256
257     redEntropy = -redEntropy;
258     greenEntropy = -greenEntropy;
259     blueEntropy = -blueEntropy;
260
261     double result = (redEntropy + greenEntropy + blueEntropy) / 3;
262     return result;
263 }
264
265 public double errorSSIM(){
266     // Error measurement using ...
267     const double C2 = 58.5225;
268     const double WR = 0.2989;
269     const double WG = 0.5870;
270     const double WB = 0.1140;
271
272
273     double N = Width * Height;
274     var (meanR, meanG, meanB) = colorMean();
275
276     double varianceR = 0;
277     double varianceG = 0;
278     double varianceB = 0;
279
280     for (int i = 0; i < Width; i++){
281         for (int j = 0; j < Height; j++){
282             varianceR += (tree.GetPixel(i, j, TopLeft).R - meanR) * (tree.GetPixel(i, j, TopLeft).R - meanR);
283             varianceG += (tree.GetPixel(i, j, TopLeft).G - meanG) * (tree.GetPixel(i, j, TopLeft).G - meanG);
284             varianceB += (tree.GetPixel(i, j, TopLeft).B - meanB) * (tree.GetPixel(i, j, TopLeft).B - meanB);
285         }
286     }
287     varianceR = varianceR / N;
288     varianceG = varianceG / N;
289     varianceB = varianceB / N;
290
291     double ssimR = (C2 / (C2 + varianceR));
292     double ssimG = (C2 / (C2 + varianceG));
293     double ssimB = (C2 / (C2 + varianceB));
294
295     double result = WR * ssimR + WG * ssimG + WB * ssimB;
```

Gambar 3.D.3. Metode QuadtreeNode (3)

```

299     public (double, double, double) colorMean(){
300         // Function to return the mean of each color channel in a node
301         double N = Width * Height;
302         double sumR = 0;
303         double sumG = 0;
304         double sumB = 0;
305
306         // Console.WriteLine($"Image Dimensions in mean: {Width}x{Height}, TopLeft: {TopLeft.Item2}x{TopLeft.Item1}, Max iteration: {TopLeft.Item2 + Width}x{TopLeft.Item1 + Height}");
307         for (int i = 0; i < Width; i++){
308             for (int j = 0; j < Height; j++){
309                 sumR += tree.GetPixel(i, j, TopLeft).R;
310                 sumG += tree.GetPixel(i, j, TopLeft).G;
311                 sumB += tree.GetPixel(i, j, TopLeft).B;
312             }
313         }
314
315         double meanR = sumR / N;
316         double meanG = sumG / N;
317         double meanB = sumB / N;
318
319         return (meanR, meanG, meanB);
320     }

```

Gambar 3.D.4. Metode QuadtreeNode (4)

## E. Program QuadtreeTree

### 1. Atribut

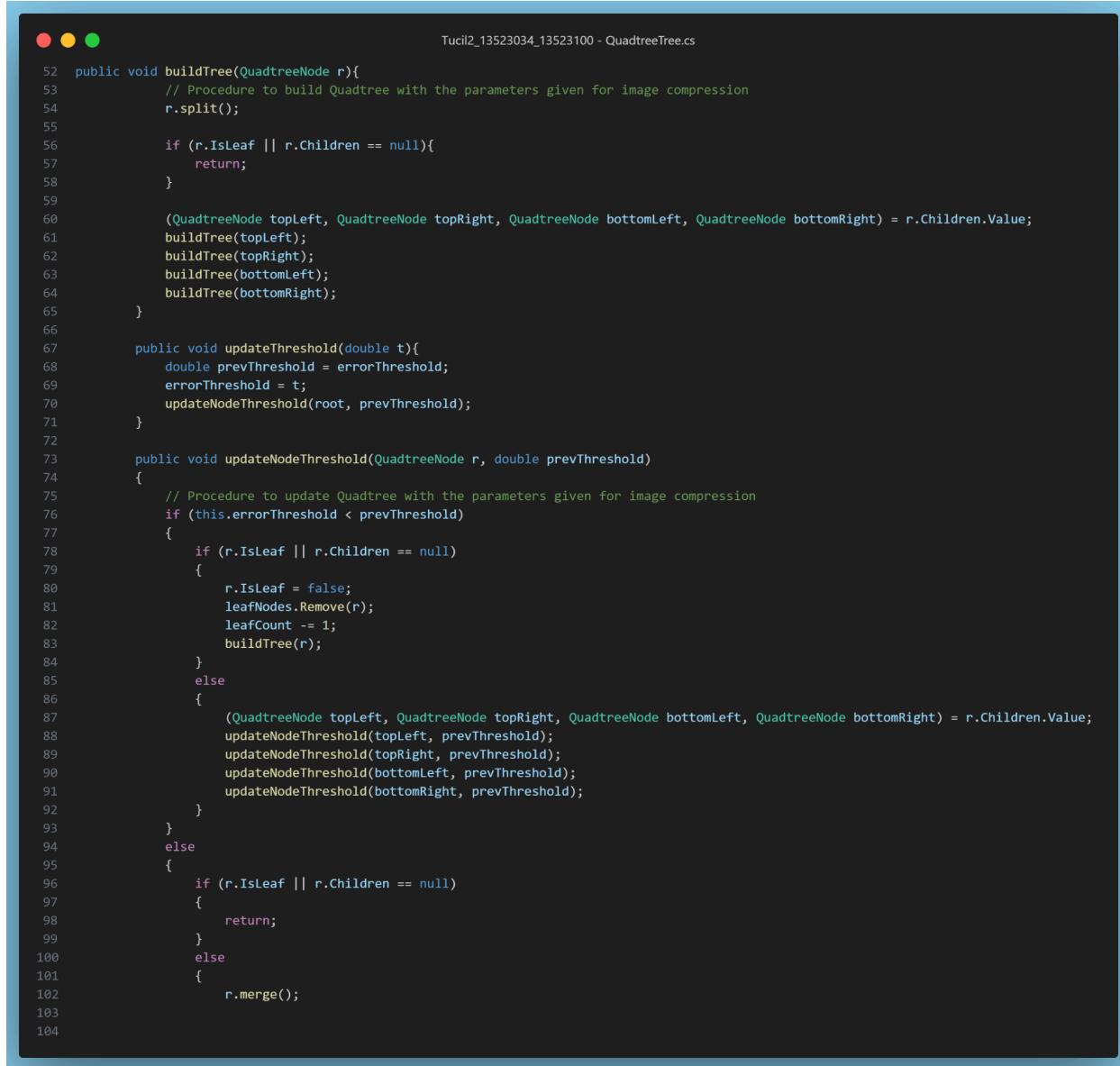
```

5  namespace Quadtree{
6      class QuadtreeTree
7      {
8          public Rgba32[,] Image { get; private set; }
9          private QuadtreeNode root;
10         public int minimumBlock { get; private set; }
11         public int thresholdMethod { get; private set; }    // 1 = Variance
12                                         // 2 = Mean Absolute Deviation (MAD)
13                                         // 3 = Max Pixel Difference
14                                         // 4 = Entropy
15                                         // 5 = Structural Similarity Index (SSIM)
16         public double errorThreshold { get; private set; }
17         public int nodeCount { get; set; }
18         public int leafCount { get; set; }
19         public int maxDepth { get; set; }
20         public List<QuadtreeNode> leafNodes { get; set; }
21
22         public QuadtreeTree(Rgba32[,] i, int width, int height, int mb, int tm, double t){
23             // User-defined Constructor
24             this.Image = i;
25             this.root = new QuadtreeNode(this, (0, 0), 0, width, height, null);
26             this.minimumBlock = mb;
27             this.thresholdMethod = tm;
28             this.errorThreshold = t;
29             this.nodeCount = 1;
30             this.leafCount = 0;
31             this.maxDepth = 0;
32             this.leafNodes = new List<QuadtreeNode>();
33
34             buildTree(root);
35         }
36
37         // Helper functions to get certain pixels from the original image
38         public Rgba32 GetPixel(int x, int y) => Image[x, y];
39         public Rgba32 GetPixel(int x, int y, (int top, int left) offset)
40         {
41             int newX = x + offset.left;
42             int newY = y + offset.top;
43
44             if (newX < 0 || newX >= Image.GetLength(0) || newY < 0 || newY >= Image.GetLength(1))
45             {
46                 throw new IndexOutOfRangeException($"Attempted to access ({newX}, {newY}), but image is {Image.GetLength(0)}x{Image.GetLength(1)}");
47             }
48
49             return Image[newX, newY];
50         }

```

Gambar 3.E.1. QuadtreeTree source code (1)

## 2. Metode



```
52 public void buildTree(QuadtreeNode r){
53     // Procedure to build Quadtree with the parameters given for image compression
54     r.split();
55
56     if (r.IsLeaf || r.Children == null){
57         return;
58     }
59
60     (QuadtreeNode topLeft, QuadtreeNode topRight, QuadtreeNode bottomLeft, QuadtreeNode bottomRight) = r.Children.Value;
61     buildTree(topLeft);
62     buildTree(topRight);
63     buildTree(bottomLeft);
64     buildTree(bottomRight);
65 }
66
67 public void updateThreshold(double t){
68     double prevThreshold = errorThreshold;
69     errorThreshold = t;
70     updateNodeThreshold(root, prevThreshold);
71 }
72
73 public void updateNodeThreshold(QuadtreeNode r, double prevThreshold)
74 {
75     // Procedure to update Quadtree with the parameters given for image compression
76     if (this.errorThreshold < prevThreshold)
77     {
78         if (r.IsLeaf || r.Children == null)
79         {
80             r.IsLeaf = false;
81             leafNodes.Remove(r);
82             leafCount -= 1;
83             buildTree(r);
84         }
85         else
86         {
87             (QuadtreeNode topLeft, QuadtreeNode topRight, QuadtreeNode bottomLeft, QuadtreeNode bottomRight) = r.Children.Value;
88             updateNodeThreshold(topLeft, prevThreshold);
89             updateNodeThreshold(topRight, prevThreshold);
90             updateNodeThreshold(bottomLeft, prevThreshold);
91             updateNodeThreshold(bottomRight, prevThreshold);
92         }
93     }
94     else
95     {
96         if (r.IsLeaf || r.Children == null)
97         {
98             return;
99         }
100        else
101        {
102            r.merge();
103        }
104    }
}

```

Gambar 3.E.2. QuadtreeTree source code (2)



The screenshot shows a code editor window with a dark theme. At the top left are three circular icons: red, yellow, and green. The title bar reads "Tucil2\_13523034\_13523100 - QuadtreeTree.cs". The code itself is a C# method named "CreateImage". It starts by defining variables for scaling (m), padding (dx, dy), and image dimensions (imageWidth, imageHeight). It then creates a new Rgba32 image of the specified size. The main loop iterates over leaf nodes. For each node, it retrieves its borders (t, l, r, b) and color mean (meanR, meanG, meanB). It calculates an average color (avgColor) and then loops through all pixels within the node's boundaries. If a pixel's x and y coordinates fall within the node's boundaries, it sets the image's color at that position to the average color.

```
117 public Rgba32[,] CreateImage()
118     {
119         // Function that creates the compressed image based on the Quadtree that is built
120         float m = 1; // scaling
121         int dx = 0, dy = 0; // padding
122
123         int imageWidth = (int)(root.Width * m + dx);
124         int imageHeight = (int)(root.Height * m + dy);
125
126         var image = new Rgba32[imageWidth, imageHeight];
127
128         // var leafNodes2 = GetLeafNodesAtDepth(this.maxDepth);
129
130         foreach (var node in leafNodes)
131         {
132             var (t, l, r, b) = node.GetBorders();
133             var (meanR, meanG, meanB) = node.colorMean();
134             Rgba32 avgColor = new Rgba32((byte)meanR, (byte)meanG, (byte)meanB, 255);
135
136             for (int y = t; y < b; y++)
137             {
138                 for (int x = l; x < r; x++)
139                 {
140                     if (x >= 0 && x < imageWidth && y >= 0 && y < imageHeight)
141                     {
142                         image[x, y] = avgColor;
143                     }
144                 }
145             }
146         }
147
148         return image;
149     }
150 }
```

Gambar 3.E.3. QuadtreeTree source code (3)

```
Tucil2_13523034_13523100 - QuadtreeTree.cs

151 public Rgba32[,] CreateImageAtDepth(int n)
152     {
153         // Function that creates the compressed image based on the Quadtree that is built
154         float m = 1; // scaling
155         int dx = 0, dy = 0; // padding
156
157         int imageWidth = (int)(root.Width * m + dx);
158         int imageHeight = (int)(root.Height * m + dy);
159
160         var image = new Rgba32[imageWidth, imageHeight];
161
162         var leafNodesDepth = GetLeafNodesAtDepth(n);
163
164         foreach (var node in leafNodesDepth)
165         {
166             var (t, l, r, b) = node.GetBorders();
167             var (meanR, meanG, meanB) = node.colorMean();
168             Rgba32 avgColor = new Rgba32((byte)meanR, (byte)meanG, (byte)meanB, 255);
169
170             for (int y = t; y < b; y++)
171             {
172                 for (int x = l; x < r; x++)
173                 {
174                     if (x >= 0 && x < imageWidth && y >= 0 && y < imageHeight)
175                     {
176                         image[x, y] = avgColor;
177                     }
178                 }
179             }
180         }
181
182         return image;
183     }
184 }
```

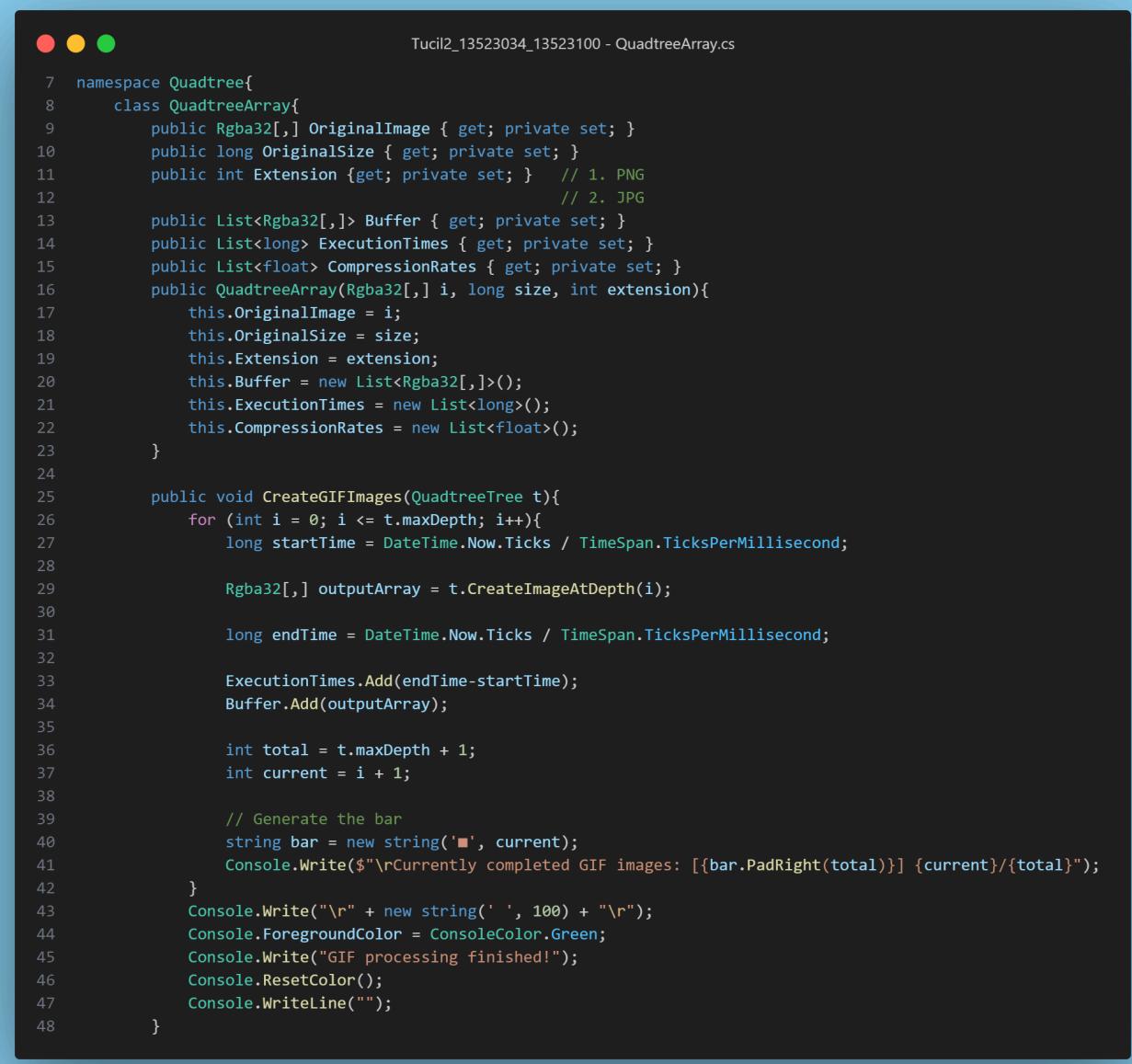
Gambar 3.E.4. QuadtreeTree source code (4)

```
185 private List<QuadtreeNode> GetLeafNodesAtDepth(int depth)
186     {
187         // MIGHT BECOME LEGACY
188         // Function that returns the leaves of the tree up to a certain depth
189         List<QuadtreeNode> leafNodes = new List<QuadtreeNode>();
190         CollectLeafNodesAtDepth(root, depth, leafNodes);
191         return leafNodes;
192     }
193
194     private void CollectLeafNodesAtDepth(QuadtreeNode node, int depth, List<QuadtreeNode> leafNodes)
195     {
196         // Procedure to collect leaf nodes up to the specified depth
197         if (node.Depth == depth || node.IsLeaf)
198         {
199             leafNodes.Add(node);
200         }
201         else if (node.Children != null)
202         {
203             (QuadtreeNode topLeft, QuadtreeNode topRight, QuadtreeNode bottomLeft, QuadtreeNode bottomRight) = node.Children.Value;
204             CollectLeafNodesAtDepth(topLeft, depth, leafNodes);
205             CollectLeafNodesAtDepth(topRight, depth, leafNodes);
206             CollectLeafNodesAtDepth(bottomLeft, depth, leafNodes);
207             CollectLeafNodesAtDepth(bottomRight, depth, leafNodes);
208         }
209     }
```

Gambar 3.E.5. QuadtreeTree source code (5)

## F. Program QuadtreeArray

### 1. Atribut

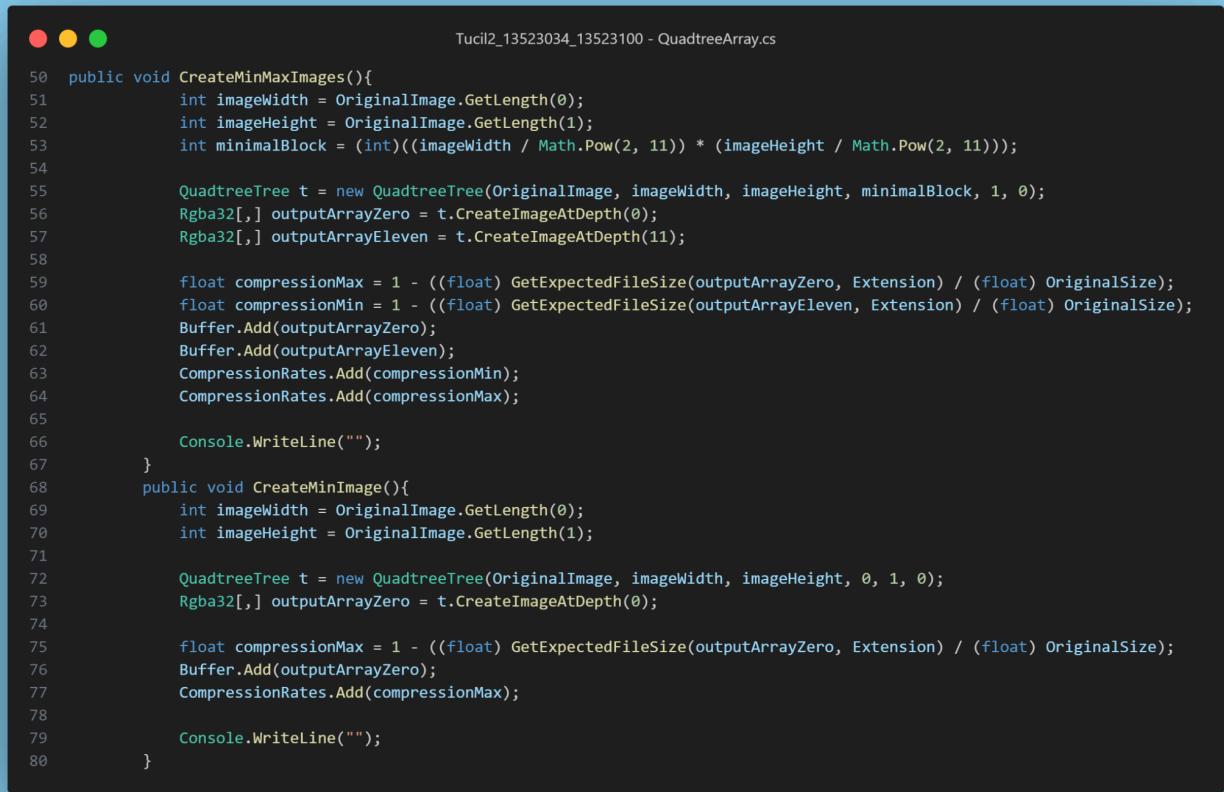


The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are three colored window control buttons (red, yellow, green). Below them, the title bar displays "Tucil2\_13523034\_13523100 - QuadtreeArray.cs". The main area of the window contains the C# source code for the `QuadtreeArray` class. The code includes properties for `OriginalImage`, `OriginalSize`, `Extension`, and lists for `Buffer`, `ExecutionTimes`, and `CompressionRates`. It also includes a constructor and a method `CreateGIFImages` that generates a progress bar and prints completion messages to the console.

```
7  namespace Quadtree{
8      class QuadtreeArray{
9          public Rgba32[,] OriginalImage { get; private set; }
10         public long OriginalSize { get; private set; }
11         public int Extension {get; private set; } // 1. PNG
12                                         // 2. JPG
13         public List<Rgba32[,]> Buffer { get; private set; }
14         public List<long> ExecutionTimes { get; private set; }
15         public List<float> CompressionRates { get; private set; }
16         public QuadtreeArray(Rgba32[,] i, long size, int extension){
17             this.OriginalImage = i;
18             this.OriginalSize = size;
19             this.Extension = extension;
20             this.Buffer = new List<Rgba32[,]>();
21             this.ExecutionTimes = new List<long>();
22             this.CompressionRates = new List<float>();
23         }
24
25         public void CreateGIFImages(QuadtreeTree t){
26             for (int i = 0; i <= t.maxDepth; i++){
27                 long startTime = DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond;
28
29                 Rgba32[,] outputArray = t.CreateImageAtDepth(i);
30
31                 long endTime = DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond;
32
33                 ExecutionTimes.Add(endTime-startTime);
34                 Buffer.Add(outputArray);
35
36                 int total = t.maxDepth + 1;
37                 int current = i + 1;
38
39                 // Generate the bar
40                 string bar = new string('■', current);
41                 Console.WriteLine($"\\rCurrently completed GIF images: [{bar.PadRight(total)}] {current}/{total}");
42             }
43             Console.Write("\\r" + new string(' ', 100) + "\\r");
44             Console.ForegroundColor = ConsoleColor.Green;
45             Console.Write("GIF processing finished!");
46             Console.ResetColor();
47             Console.WriteLine("");
48         }
49     }
```

Gambar 3.F.1. QuadtreeArray source code (1)

## 2. Metode



```
Tucil2_13523034_13523100 - QuadtreeArray.cs

50 public void CreateMinMaxImages(){
51     int imageWidth = OriginalImage.GetLength(0);
52     int imageHeight = OriginalImage.GetLength(1);
53     int minimalBlock = (int)((imageWidth / Math.Pow(2, 11)) * (imageHeight / Math.Pow(2, 11)));
54
55     QuadtreeTree t = new QuadtreeTree(OriginalImage, imageWidth, imageHeight, minimalBlock, 1, 0);
56     Rgba32[,] outputArrayZero = t.CreateImageAtDepth(0);
57     Rgba32[,] outputArrayEleven = t.CreateImageAtDepth(11);
58
59     float compressionMax = 1 - ((float) GetExpectedFileSize(outputArrayZero, Extension) / (float) OriginalSize);
60     float compressionMin = 1 - ((float) GetExpectedFileSize(outputArrayEleven, Extension) / (float) OriginalSize);
61     Buffer.Add(outputArrayZero);
62     Buffer.Add(outputArrayEleven);
63     CompressionRates.Add(compressionMin);
64     CompressionRates.Add(compressionMax);
65
66     Console.WriteLine("");
67 }
68 public void CreateMinImage(){
69     int imageWidth = OriginalImage.GetLength(0);
70     int imageHeight = OriginalImage.GetLength(1);
71
72     QuadtreeTree t = new QuadtreeTree(OriginalImage, imageWidth, imageHeight, 0, 1, 0);
73     Rgba32[,] outputArrayZero = t.CreateImageAtDepth(0);
74
75     float compressionMax = 1 - ((float) GetExpectedFileSize(outputArrayZero, Extension) / (float) OriginalSize);
76     Buffer.Add(outputArrayZero);
77     CompressionRates.Add(compressionMax);
78
79     Console.WriteLine("");
80 }
```

Gambar 3.F.2. QuadtreeArray source code (2)

```
● ● ● Tucil2_13523034_13523100 - QuadtreeArray.cs

81  public static long GetExpectedFileSize(Rgba32[,] pixelMatrix, int extension)
82  {
83      using var image = MatrixToImage(pixelMatrix);
84      using var ms = new MemoryStream();
85
86      switch (extension)
87      {
88          case 1: // PNG
89              image.Save(ms, new PngEncoder());
90              break;
91          case 2: // JPG
92              image.Save(ms, new JpegEncoder());
93              break;
94          default:
95              throw new ArgumentException("Unsupported extension value.");
96      }
97
98      return ms.Length;
99  }
100
101     private static Image<Rgba32> MatrixToImage(Rgba32[,] matrix)
102  {
103      int width = matrix.GetLength(0);
104      int height = matrix.GetLength(1);
105      var image = new Image<Rgba32>(width, height);
106
107      for (int x = 0; x < width; x++)
108      {
109          for (int y = 0; y < height; y++)
110          {
111              image[x, y] = matrix[x, y];
112          }
113      }
114
115      return image;
116  }
```

Gambar 3.F.3. QuadtreeArray source code (3)

## IV. Pengujian (Input dan Output)

### A. Uji Normal Setiap Threshold Method

#### 1. Variance PNG

```
STATUS INPUT
1. Alamat gambar : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\CasPrice.png
2. Metode error : Variance
3. Target kompresi : 0
4. Threshold error : 800
5. Minimum block : 100
6. Output image path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.png
7. Output GIF path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.gif
```

Gambar 4.A.1. Input Castorice.png dengan variansi

```
OUTPUT
Waktu eksekusi gambar: 13118 ms
Waktu eksekusi GIF: 5677 ms

Kedalaman Pohon: 7
Jumlah Simpul: 6941
Jumlah Daun: 5206

Ukuran file gambar sebelum kompresi: 304700 bytes
Ukuran file gambar setelah kompresi: 65386 bytes
Persentase kompresi: 78,54086%
```

Gambar 4.A.2. Hasil output test case Castorice.png dengan variansi

Original	Kompresi
 A chibi-style illustration of a purple-haired girl with a crown and flower wreath, sitting on the ground. Butterflies are flying around her. The image is in full color.	 The same chibi-style illustration, but significantly more pixelated and compressed. The colors are less vibrant, and the details are lost due to the compression.

Tabel 4.A.1. Perbandingan gambar Castorice.png sebelum dan setelah kompresi dengan variansi

## 2. Mean Absolute Deviation PNG

```
STATUS INPUT
1. Alamat gambar : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\sylphieeeeeee.png
2. Metode error : Mean Absolute Deviation (MAD)
3. Target kompresi : 0
4. Threshold error : 10
5. Minimum block : 20
6. Output image path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.png
7. Output GIF path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.gif
```

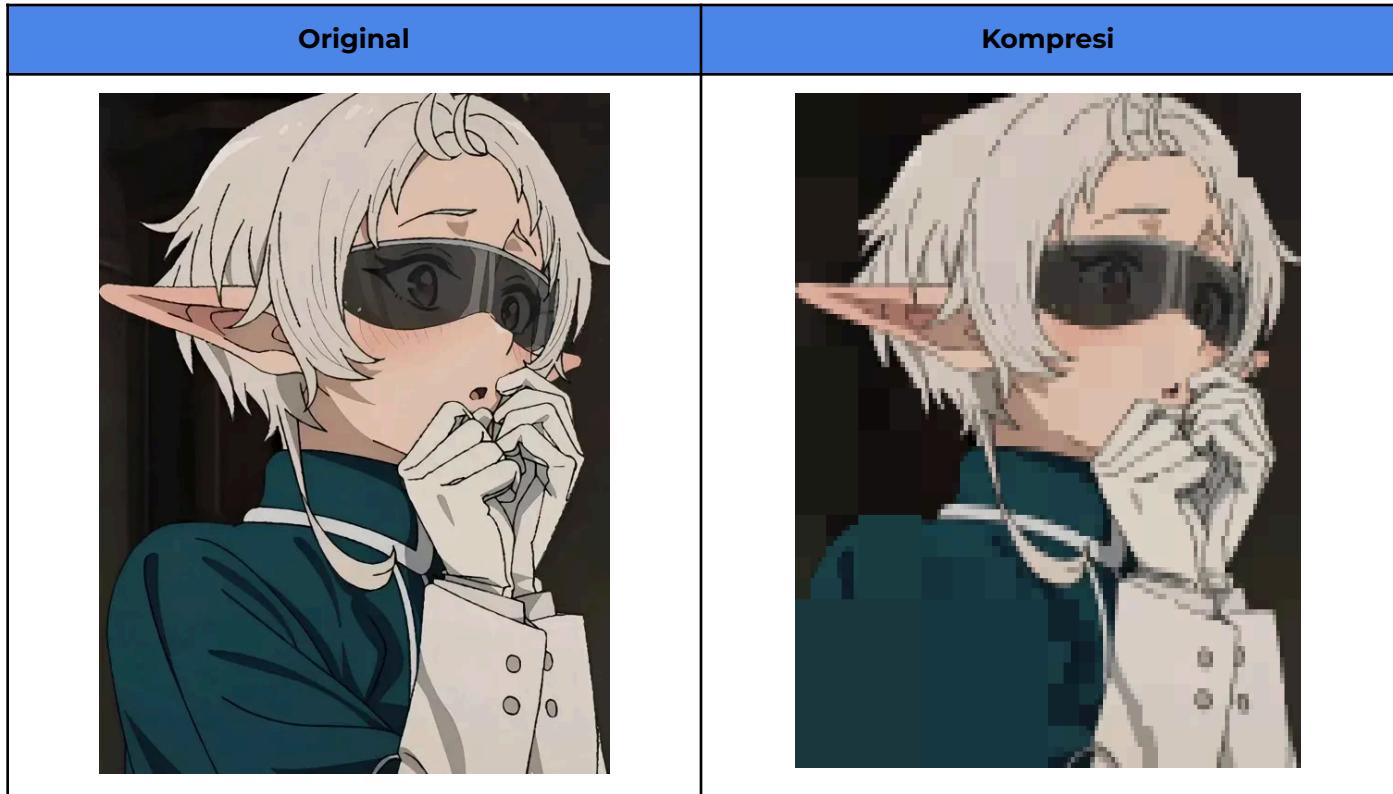
Gambar 4.A.3. Input sylphieeeeeee.png dengan MAD

```
OUTPUT
Waktu eksekusi gambar: 615 ms
Waktu eksekusi GIF: 361 ms

Kedalaman Pohon: 7
Jumlah Simpul: 7725
Jumlah Daun: 5794

Ukuran file gambar sebelum kompresi: 378415 bytes
Ukuran file gambar setelah kompresi: 32806 bytes
Persentase kompresi: 91,33068%
```

Gambar 4.A.4. Hasil output test case sylphieeeeeee.png dengan MAD



Tabel 4.A.2. Perbandingan gambar sylphieeeeeee.png sebelum dan setelah kompresi dengan MAD

### 3. Max Pixel Difference PNG

```
STATUS INPUT
1. Alamat gambar : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\JeanDarc.png
2. Metode error : Max Pixel Difference
3. Target kompresi : 0
4. Threshold error : 20
5. Minimum block : 20
6. Output image path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.png
7. Output GIF path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.gif
```

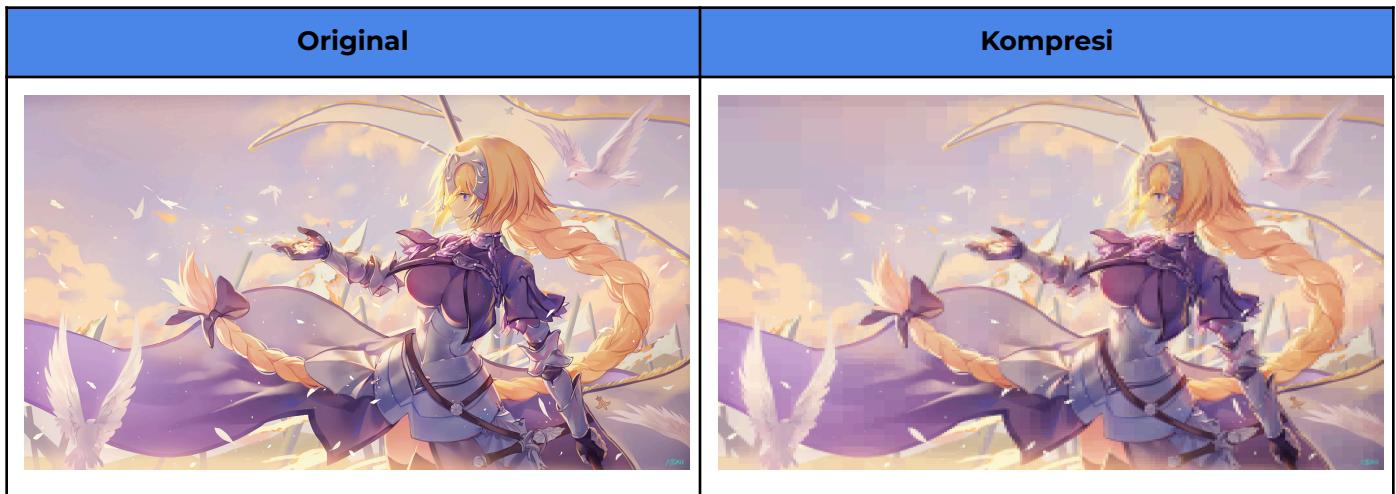
Gambar 4.A.5. Input JeanDarc.png dengan MPD

```
OUTPUT
Waktu eksekusi gambar: 2897 ms
Waktu eksekusi GIF: 6704 ms

Kedalaman Pohon: 9
Jumlah Simpul: 44825
Jumlah Daun: 33619

Ukuran file gambar sebelum kompresi: 4454596 bytes
Ukuran file gambar setelah kompresi: 185339 bytes
Persentase kompresi: 95,83938%
```

Gambar 4.A.6. Hasil output test case JeanDarc.png dengan MPD



Tabel 4.A.3. Perbandingan gambar JeanDarc.png sebelum dan setelah kompresi dengan MPD

## 4. Entropy PNG

```
STATUS INPUT
1. Alamat gambar : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\tiamat.png
2. Metode error : Entropy
3. Target kompresi : 0
4. Threshold error : 1
5. Minimum block : 10
6. Output image path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.png
7. Output GIF path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\tiamat.gif
```

Gambar 4.A.7. Input tiamat.png dengan entropy

**OUTPUT**

Waktu eksekusi gambar: 3097 ms  
Waktu eksekusi GIF: 2866 ms

Kedalaman Pohon: 9  
Jumlah Simpul: 89981  
Jumlah Daun: 67486

Ukuran file gambar sebelum kompresi: 1807799 bytes  
Ukuran file gambar setelah kompresi: 199798 bytes  
Persentase kompresi: 88,948%

Gambar 4.A.8. Hasil output test case tiamat.png dengan entropy



Tabel 4.A.4. Perbandingan gambar tiamat.png sebelum dan setelah kompresi dengan entropy

## 5. Structural Similarity Index PNG

```
STATUS INPUT
1. Alamat gambar : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\JeanDarc.png
2. Metode error : Structural Similarity Index (SSIM)
3. Target kompresi : 0
4. Threshold error : 0,001
5. Minimum block : 10
6. Output image path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.png
7. Output GIF path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.gif
```

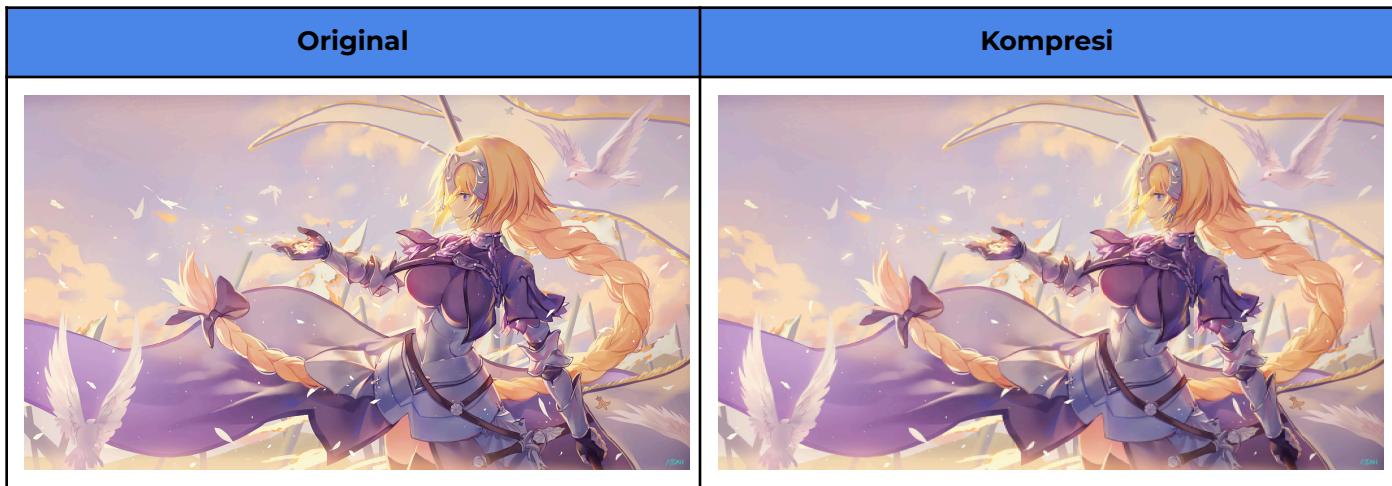
Gambar 4.A.9. Input JeanDarc.png dengan SSIM

```
OUTPUT
Waktu eksekusi gambar: 18387 ms
Waktu eksekusi GIF: 6828 ms

Kedalaman Pohon: 9
Jumlah Simpul: 349525
Jumlah Daun: 262144

Ukuran file gambar sebelum kompresi: 4454596 bytes
Ukuran file gambar setelah kompresi: 623529 bytes
Persentase kompresi: 86,00257%
```

Gambar 4.A.10. Hasil output test case JeanDarc.png dengan SSIM



Tabel 4.A.5. Perbandingan gambar JeanDarc.png sebelum dan setelah kompresi dengan SSIM

## 6. Variance JPG

```
STATUS INPUT
1. Alamat gambar : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\SnakeRiver.jpg
2. Metode error : Variance
3. Target kompresi : 0
4. Threshold error : 500
5. Minimum block : 200
6. Output image path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.jpg
7. Output GIF path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\SnakeRiver.gif
```

Gambar 4.A.11. Input SnakeRiver.jpg dengan variansi

```
OUTPUT
Waktu eksekusi gambar: 45403 ms
Waktu eksekusi GIF: 18625 ms

Kedalaman Pohon: 8
Jumlah Simpul: 40665
Jumlah Daun: 30499

Ukuran file gambar sebelum kompresi: 5245329 bytes
Ukuran file gambar setelah kompresi: 1049149 bytes
Persentase kompresi: 79,99841%
```

Gambar 4.A.12. Hasil output test case SnakeRiver.jpg dengan variansi



Tabel 4.A.6. Perbandingan gambar SnakeRiver.jpg sebelum dan setelah kompresi dengan variansi

## 7. Mean Absolute Deviation JPG

```
STATUS INPUT
1. Alamat gambar : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Madam.jpg
2. Metode error : Mean Absolute Deviation (MAD)
3. Target kompresi : 0
4. Threshold error : 25
5. Minimum block : 20
6. Output image path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.jpg
7. Output GIF path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.gif
```

Gambar 4.A.13. Input Madam.jpg dengan MAD

### OUTPUT

Waktu eksekusi gambar: 1427 ms

Waktu eksekusi GIF: 907 ms

Kedalaman Pohon: 7

Jumlah Simpul: 4913

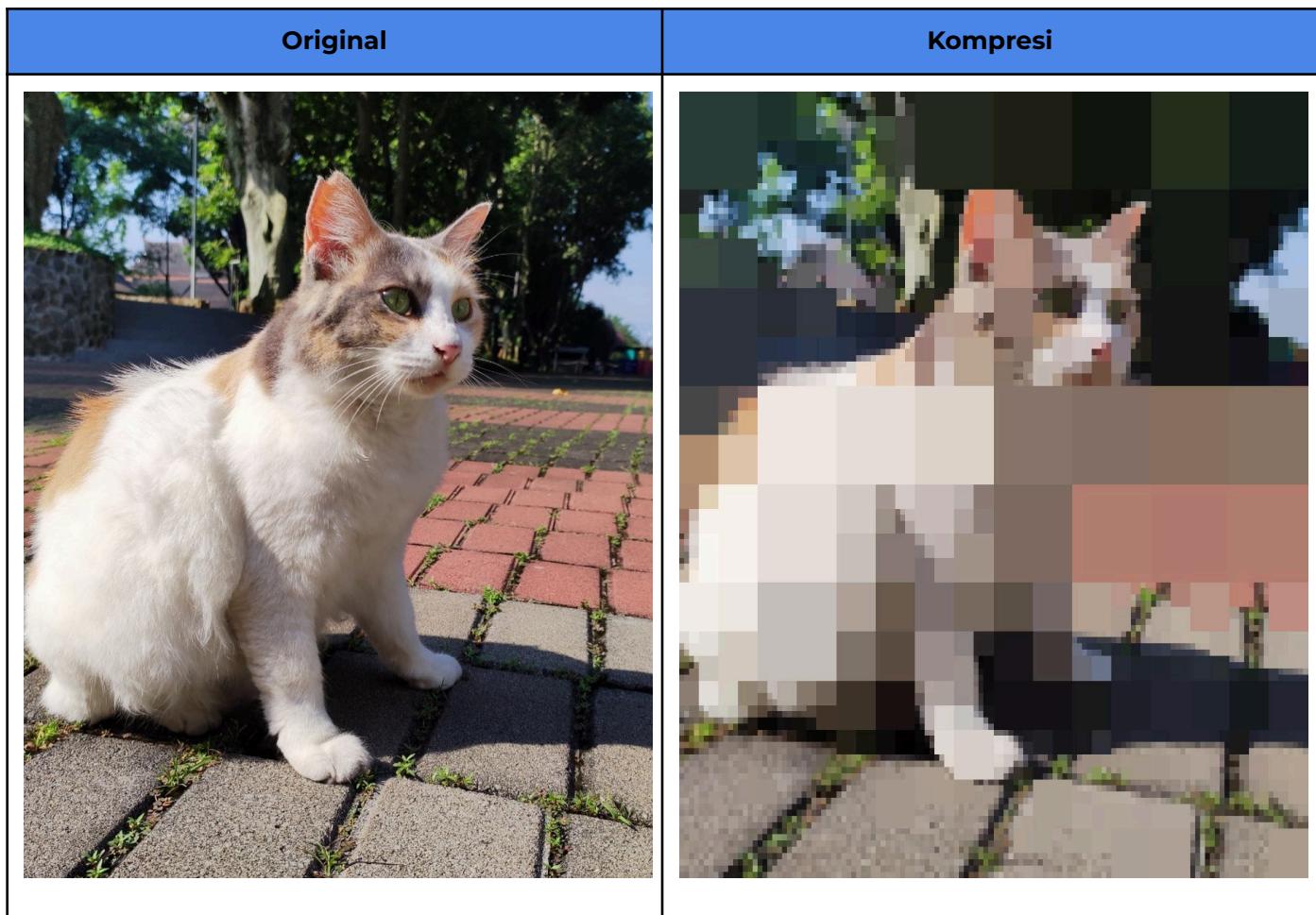
Jumlah Daun: 3685

Ukuran file gambar sebelum kompresi: 234888 bytes

Ukuran file gambar setelah kompresi: 73118 bytes

Persentase kompresi: 68,871124%

Gambar 4.A.14. Hasil output test case Madam.jpg dengan MAD



Tabel 4.A.7. Perbandingan gambar Madam.jpg sebelum dan setelah kompresi dengan MAD

## 8. Max Pixel Difference JPG

```
STATUS INPUT
1. Alamat gambar : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\NeuschwansteinCastle.jpg
2. Metode error : Max Pixel Difference
3. Target kompresi : 0
4. Threshold error : 30
5. Minimum block : 10
6. Output image path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.jpg
7. Output GIF path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.gif
```

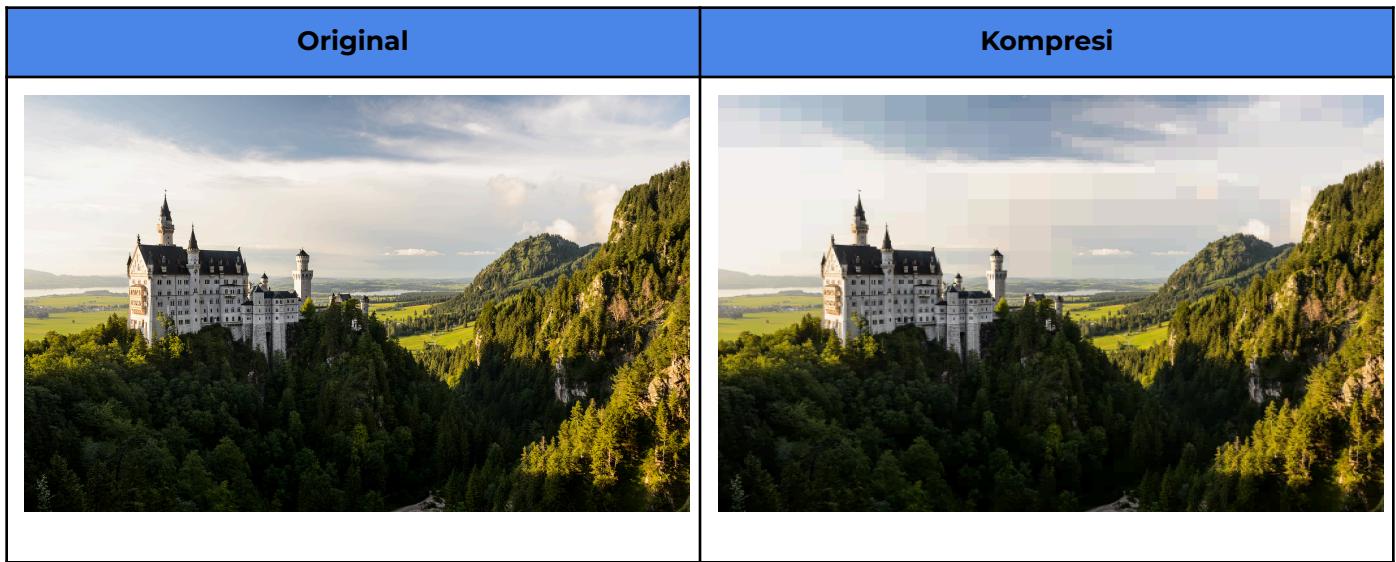
Gambar 4.A.15. Input NeuschwansteinCastle.jpg dengan MPD

```
OUTPUT
Waktu eksekusi gambar: 2158 ms
Waktu eksekusi GIF: 4964 ms

Kedalaman Pohon: 9
Jumlah Simpul: 171689
Jumlah Daun: 128767

Ukuran file gambar sebelum kompresi: 2027180 bytes
Ukuran file gambar setelah kompresi: 413943 bytes
Persentase kompresi: 79,58035%
```

Gambar 4.A.16. Hasil output test case NeuschwansteinCastle.jpg dengan MPD



Tabel 4.A.8. Perbandingan gambar NeuschwansteinCastle.jpg sebelum dan setelah kompresi dengan MPD

## 9. Entropy JPG

```
STATUS INPUT
1. Alamat gambar : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\chameleon.jpg
2. Metode error : Entropy
3. Target kompresi : 0
4. Threshold error : 1
5. Minimum block : 10
6. Output image path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.jpg
7. Output GIF path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.gif
```

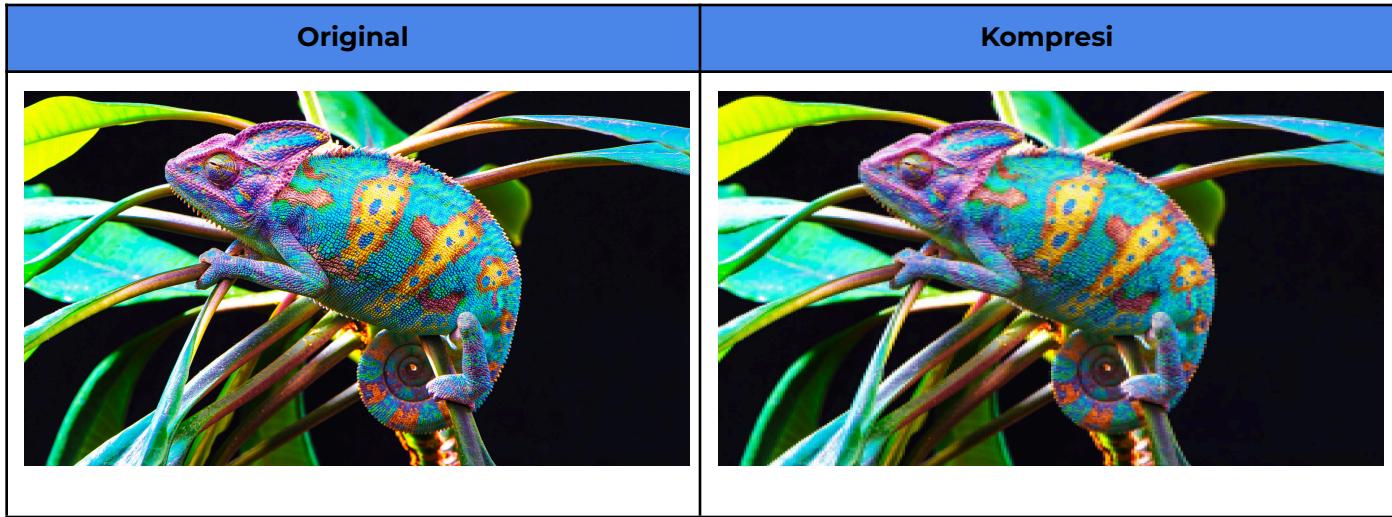
Gambar 4.A.17. Hasil output test case chameleon.jpg dengan entropy

```
OUTPUT
Waktu eksekusi gambar: 3023 ms
Waktu eksekusi GIF: 2666 ms

Kedalaman Pohon: 8
Jumlah Simpul: 69945
Jumlah Daun: 52459

Ukuran file gambar sebelum kompresi: 753412 bytes
Ukuran file gambar setelah kompresi: 311061 bytes
Persentase kompresi: 58,71303%
```

Gambar 4.A.18. Hasil output test case chameleon.jpg dengan entropy



Tabel 4.A.9. Perbandingan gambar chameleon.jpg sebelum dan setelah kompresi dengan entropy

## 10. Structural Similarity Index JPG

```
STATUS INPUT
1. Alamat gambar : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\NeuschwansteinCastle.jpg
2. Metode error : Structural Similarity Index (SSIM)
3. Target kompresi : 0
4. Threshold error : 0,005
5. Minimum block : 30
6. Output image path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.jpg
7. Output GIF path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.gif
```

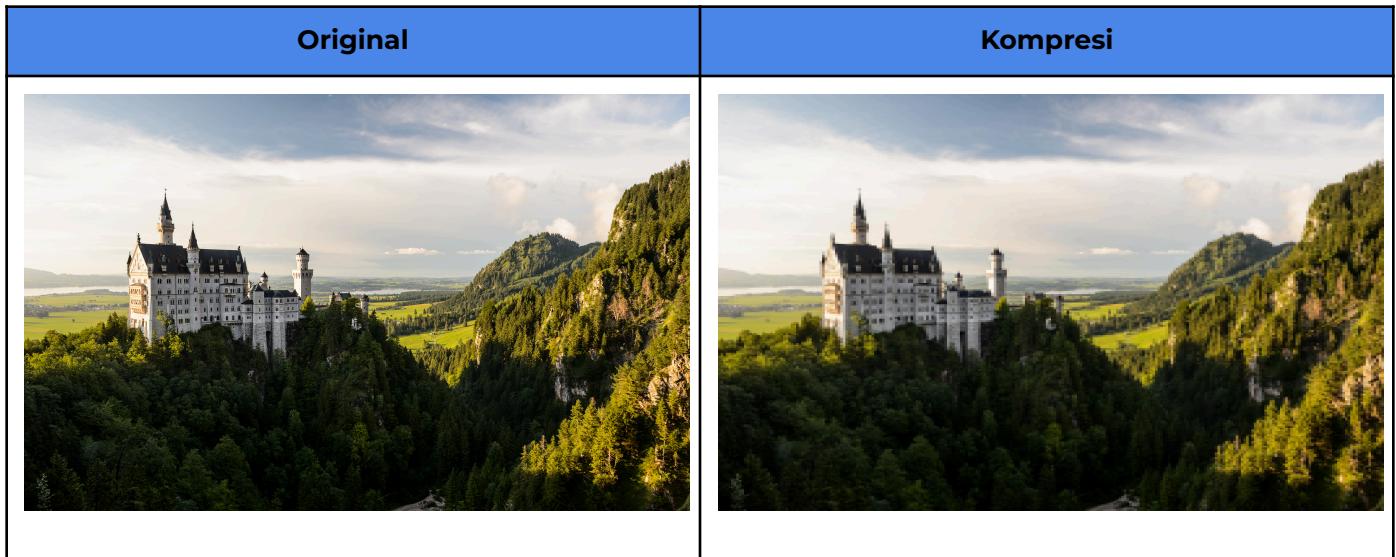
Gambar 4.A.19. Input NeuschwansteinCastle.jpg dengan SSIM

```
OUTPUT
Waktu eksekusi gambar: 12127 ms
Waktu eksekusi GIF: 4432 ms

Kedalaman Pohon: 8
Jumlah Simpul: 87381
Jumlah Daun: 65536

Ukuran file gambar sebelum kompresi: 2027180 bytes
Ukuran file gambar setelah kompresi: 323609 bytes
Persentase kompresi: 84,03649%
```

Gambar 4.A.20. Hasil output test case NeuschwansteinCastle.jpg dengan SSIM



Tabel 4.A.10. Perbandingan gambar NeuschwansteinCastle.jpg sebelum dan setelah kompresi dengan SSIM

## B. Uji Bonus Kompresi Target

### 1. Variance

```
STATUS INPUT
1. Alamat gambar : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Madam.jpg
2. Metode error : Variance
3. Target kompresi : 0,7
4. Threshold error : XXX
5. Minimum block : XXX
6. Output image path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.jpg
7. Output GIF path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Madam.gif
```

Gambar 4.B.1. Input Madam.jpg dengan target variansi

```
OUTPUT
Waktu eksekusi gambar: 6082 ms
Waktu eksekusi GIF: 819 ms

Ambang atas error: 762,01171875
Ukuran blok minimum: 101
Kedalaman Pohon: 6
Jumlah Simpul: 3061
Jumlah Daun: 1906

Ukuran file gambar sebelum kompresi: 234888 bytes
Ukuran file gambar setelah kompresi: 68949 bytes
Target kompresi: 70%
Persentase kompresi: 70,64601%
```

Gambar 4.B.2. Hasil output test case Madam.jpg dengan target variansi

Original	Kompresi
	

Tabel 4.B.1. Perbandingan gambar Madam.jpg sebelum dan setelah kompresi target variansi

## 2. Mean Absolute Deviation

```
STATUS INPUT
1. Alamat gambar : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\CasPrice.png
2. Metode error : Mean Absolute Deviation (MAD)
3. Target kompresi : 0,8
4. Threshold error : XXX
5. Minimum block : XXX
6. Output image path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.png
7. Output GIF path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.gif
```

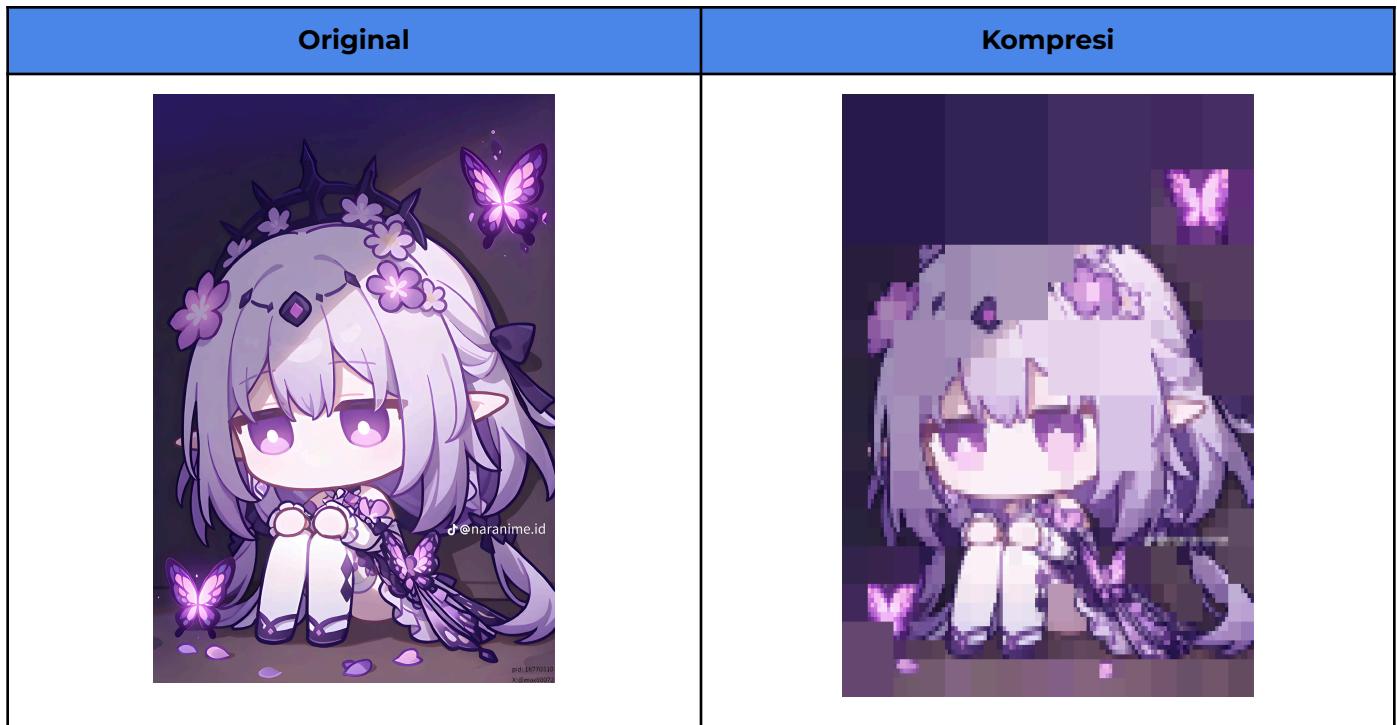
Gambar 4.B.3. Input CasPrice.jpg dengan target MAD

```
OUTPUT
Waktu eksekusi gambar: 66058 ms
Waktu eksekusi GIF: 6089 ms

Ambang atas error: 23,90625
Ukuran blok minimum: 170
Kedalaman Pohon: 7
Jumlah Simpul: 7769
Jumlah Daun: 4159

Ukuran file gambar sebelum kompresi: 304700 bytes
Ukuran file gambar setelah kompresi: 59984 bytes
Target kompresi: 80%
Persentase kompresi: 80,31375%
```

Gambar 4.B.4. Hasil output test case CasPrice.jpg dengan target MAD



Tabel 4.B.2. Perbandingan gambar CasPrice.jpg sebelum dan setelah kompresi target MAD

### 3. Max Pixel Difference

```
STATUS INPUT
1. Alamat gambar : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\chameleon.jpg
2. Metode error : Max Pixel Difference
3. Target kompresi : 0,4
4. Threshold error : XXX
5. Minimum block : XXX
6. Output image path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.jpg
7. Output GIF path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.gif
```

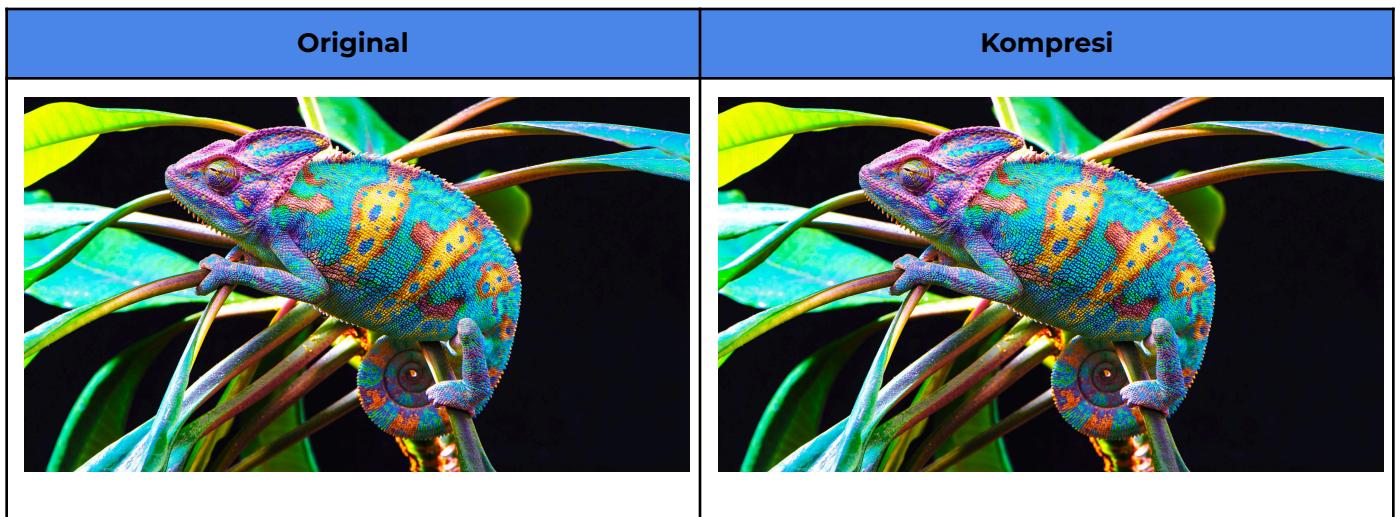
Gambar 4.B.5. Input chameleon.jpg dengan target MPD

```
OUTPUT
Waktu eksekusi gambar: 115559 ms
Waktu eksekusi GIF: 2895 ms

Ambang atas error: 0,12451171875
Ukuran blok minimum: 0
Kedalaman Pohon: 9
Jumlah Simpul: 318253
Jumlah Daun: 238690

Ukuran file gambar sebelum kompresi: 753412 bytes
Ukuran file gambar setelah kompresi: 381146 bytes
Target kompresi: 40%
Persentase kompresi: 49,410683%
```

Gambar 4.B.6. Hasil output test case chameleon.jpg dengan target MPD



Tabel 4.B.3. Perbandingan gambar chameleon.jpg sebelum dan setelah kompresi target MPD

## 4. Entropy

```
STATUS INPUT
1. Alamat gambar : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\JeanDarc.png
2. Metode error : Entropy
3. Target kompresi : 0,9
4. Threshold error : XXX
5. Minimum block : XXX
6. Output image path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.png
7. Output GIF path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.gif
```

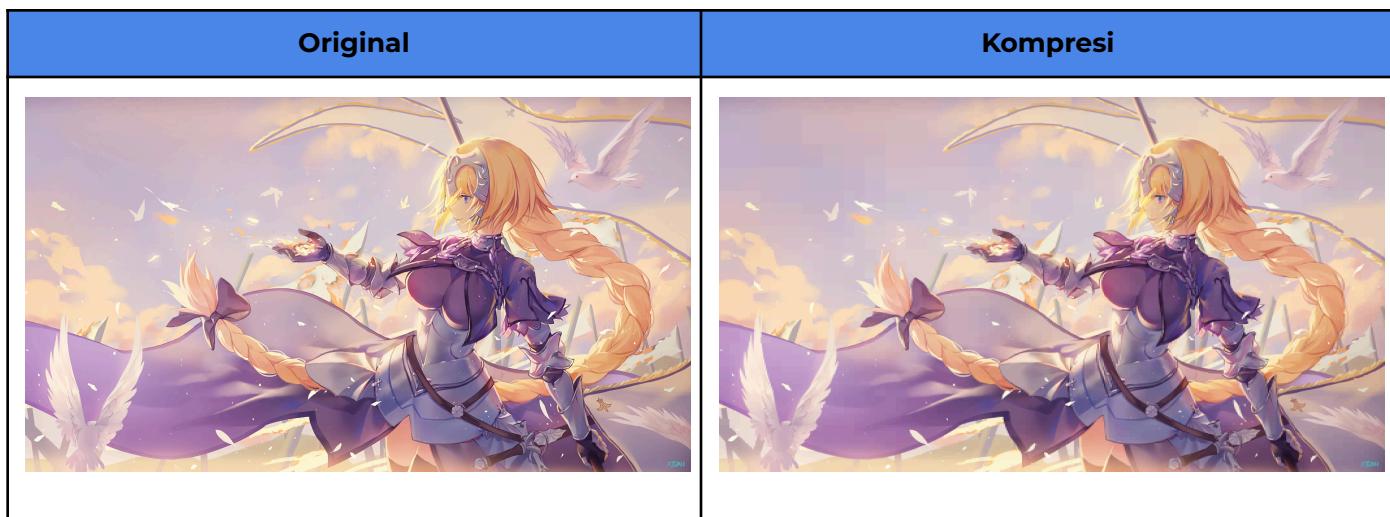
Gambar 4.B.7. Hasil output test case JeanDarc.jpg dengan target entropy

```
OUTPUT
Waktu eksekusi gambar: 110402 ms
Waktu eksekusi GIF: 6547 ms

Ambang atas error: 2,5
Ukuran blok minimum: 9
Kedalaman Pohon: 9
Jumlah Simpul: 202461
Jumlah Daun: 104872

Ukuran file gambar sebelum kompresi: 4454596 bytes
Ukuran file gambar setelah kompresi: 454990 bytes
Target kompresi: 90%
Persentase kompresi: 89,78606%
```

Gambar 4.B.8. Hasil output test case JeanDarc.jpg dengan target entropy



Tabel 4.B.4. Perbandingan gambar JeanDarc.jpg sebelum dan setelah kompresi target entropy

## 5. Structural Similarity Index

```
STATUS INPUT
1. Alamat gambar : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\sylphieeeeeee.jpg
2. Metode error : Structural Similarity Index (SSIM)
3. Target kompresi : 0,728
4. Threshold error : XXX
5. Minimum block : XXX
6. Output image path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.png
7. Output GIF path : C:\Users\Aryo\PersonalMade\ITB Kuliah Semesteran\Semester 4\Strategi Algoritma\Tucil-Tubes 2025\Tucil2_13523034_13523100\test\Output.gif
```

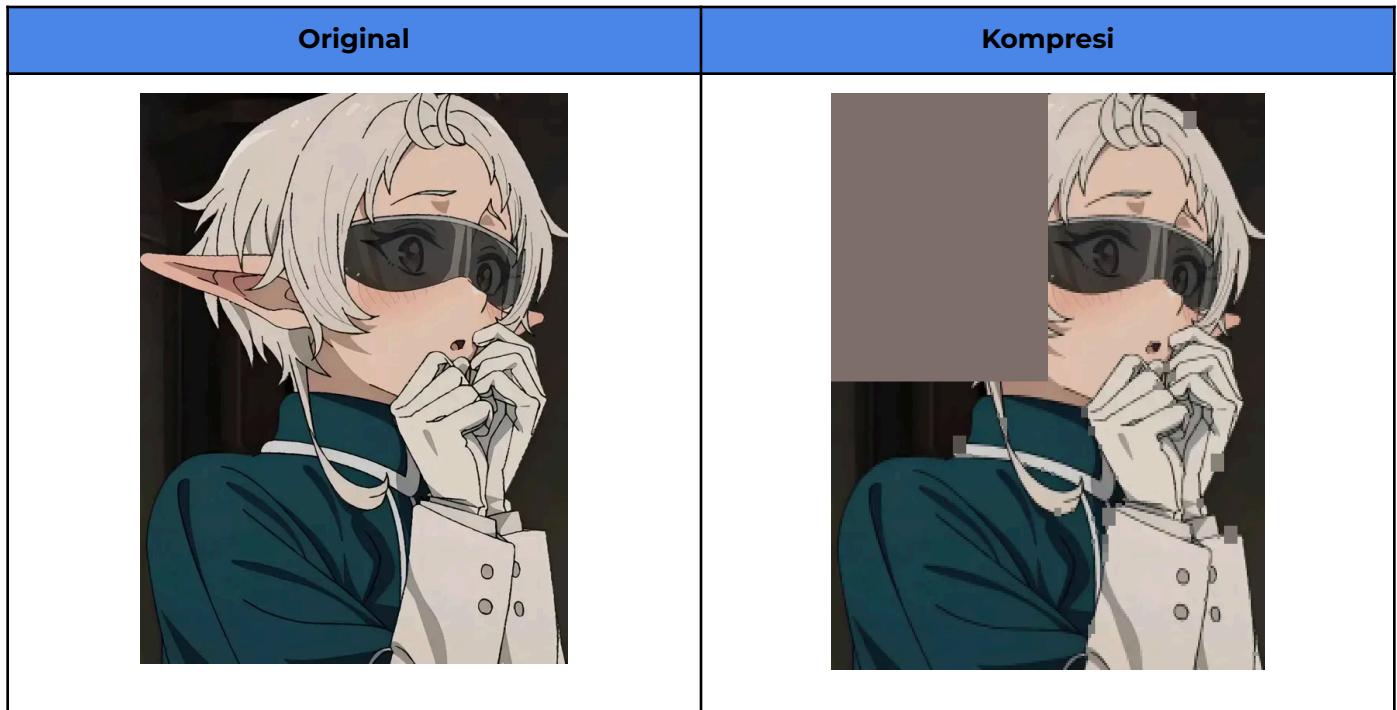
Gambar 4.B.9. Hasil output test case sylphieeeeeee.jpg dengan target SSIM

```
OUTPUT
Waktu eksekusi gambar: 11268 ms
Waktu eksekusi GIF: 455 ms

Ambang atas error: 0,00830078125
Ukuran blok minimum: 2
Kedalaman Pohon: 8
Jumlah Simpul: 149241
Jumlah Daun: 48526

Ukuran file gambar sebelum kompresi: 378415 bytes
Ukuran file gambar setelah kompresi: 102589 bytes
Target kompresi: 72,799995%
Persentase kompresi: 72,88982%
```

Gambar 4.B.10. Hasil output test case sylphieeeeeee.jpg dengan target SSIM



Tabel 4.B.5. Perbandingan gambar sylphieeeeeee.jpg sebelum dan setelah kompresi target SSIM

## V. Hasil Analisis Percobaan

- Semakin kecil nilai threshold yang digunakan, semakin tinggi kualitas gambar hasil kompresi karena blok-blok kecil tetap dipertahankan. Namun, ini juga menyebabkan ukuran file hasil kompresi menjadi lebih besar karena jumlah blok meningkat.
- Minimum block size berperan sebagai batas seberapa kecil area gambar bisa dibagi. Nilai minimum yang terlalu kecil memungkinkan detail tinggi namun meningkatkan kompleksitas dan ukuran hasil. Sebaliknya, nilai minimum yang besar menghasilkan kompresi yang lebih agresif, berisiko mengaburkan detail gambar.
- Menggunakan pilihan minimum block size dan nilai threshold yang tepat dapat menghasilkan gambar dengan nilai kompresi besar tanpa mengurangi kualitas secara signifikan.
- Metode perhitungan SSIM memberikan kompresi blok yang kurang konsisten dibanding metode-metode lainnya.
- Kompleksitas dari setiap metode perhitungan error adalah  $O(N^2)$  karena setiap metode mengiterasikan setiap pixel dalam satu blok.
- Dengan alasan yang identis, menormalisasikan warna sebuah blok juga memiliki kompleksitas yang sama, yaitu  $O(N^2)$ .
- Kompleksitas waktu algoritma divide and conquer adalah:

$$T(N) = 2 * N^2 \text{ jika } \frac{N^2}{16} < \text{minimum blok atau error} < \text{threshold}$$

$$T(N) = 4 * T\left(\frac{N}{4}\right) + N^2 \text{ jika } \frac{N^2}{16} \geq \text{minimum blok atau error} < \text{threshold}$$

Asumsikan  $N$  adalah  $4^k$ , maka:

$$\begin{aligned} T(N) &= N^2 + 4 * T\left(\frac{N}{4}\right) \\ &= N^2 + 4 * (N^2 + 4 * T\left(\frac{N}{16}\right)) \\ &= N^2 + 4 * (N^2 + 4 * (N^2 + 4 * T\left(\frac{N}{64}\right))) \\ &= \dots \\ &= \frac{4^k - 1}{3} * N^2 + 4^k * T\left(\frac{N}{4^k}\right) \end{aligned}$$

Karena  $N = 4^k$ , sehingga

$$\begin{aligned}
T(N) &= \frac{N-1}{3} * N^2 + N * T\left(\frac{N}{N}\right) \\
&= \frac{N-1}{3} * N^2 + N * T(1) \\
&= \frac{N^3 - N^2}{3} + N * 1 \\
&= \frac{N^3}{3} - \frac{N^2}{3} + N = \mathcal{O}(N^3)
\end{aligned}$$

## VI. Implementasi Bonus

### A. Target Persentase Kompresi

Target Persentase Kompresi adalah fitur untuk memungkinkan pengguna menentukan target persentase kompresi pada gambar. Mode ini dapat dimatikan dengan menginput nilai 0 ketika diminta. Ketika mode diaktifkan, algoritma akan menyesuaikan nilai threshold secara otomatis untuk mencapai target persentase kompresi yang ditentukan. Penyesuaian threshold ini dilakukan secara dinamis, memberikan fleksibilitas dalam proses kompresi untuk memenuhi target efisiensi sambil mempertahankan kualitas gambar.

Algoritma yang digunakan untuk menyesuaikan threshold secara dinamis adalah algoritma decrease and conquer (mirip dengan binary search) seperti ini:

1. Inisialisasi **range threshold** yang mungkin sesuai dengan metode perhitungan variansi dan **minimum block** awal,
2. Ambil **threshold** dari tengah-tengah range threshold,
3. Lakukan kompresi menggunakan **threshold** dan **minimum block**,
4. Kalkulasikan persentase hasil kompresi,
5. Untuk kasus persentase hasil kompresi dan target kompresi **cukup dekat** (selisih 1%),  
**SOLVE:** Keluarkan gambar hasil kompresi,
6. Untuk kasus lainnya,
  - (A) **DECREASE:** Bagi **range threshold** menjadi dua sama besar,
  - (B) **CONQUER:** Lakukan algoritma yang sama (balik ke langkah 2) untuk **range threshold** yang lebih kecil jika target kompresi lebih kecil dari persentase hasil kompresi atau sebaliknya,
7. Jika loop 2-6 sudah mencapai **10 iterasi**, balik ke langkah 1 dengan **minimum block** dibagi 4, (langkah ini untuk membuat gambar lebih konsisten)
8. Jika step 8 sudah mencapai **minimum block** kurang dari 1, hentikan proses dan keluarkan gambar hasil kompresi terakhir.

### B. Structural Similarity Index (SSIM)

Structural Similarity Index atau SSIM adalah alternatif metode pengukuran error. Rumus utama SSIM adalah sebagai berikut:

$$SSIM_c(x, y) = \frac{(2\mu_{x,c}\mu_{y,c} + C_1)(2\sigma_{xy,c} + C_2)}{(\mu_{x,c}^2 + \mu_{y,c}^2 + C_1)(\sigma_{x,c}^2 + \sigma_{y,c}^2 + C_2)}$$

Dengan:

x = blok gambar sebelum kompresi

y = blok gambar sesudah kompresi

C1 dan C2 = konstanta kecil

Diketahui blok gambar sesudah kompresi adalah blok dengan normalisasi warna blok sesuai dengan rata-rata nilai RGB blok, atau block dengan warna

$$Color_c = \frac{1}{N} \sum_{i=1}^N P_{i,c}$$

Kemudian, dapat diperoleh:

$$\mu_{y,c} = \frac{1}{N} \sum_{i=1}^N Color_c$$

$$\mu_{y,c} = \frac{1}{N} * Color_c * \sum_{i=1}^N 1$$

$$\mu_{y,c} = \frac{1}{N} * Color_c * N$$

$$\mu_{y,c} = Color_c$$

$$\mu_{y,c} = \frac{1}{N} \sum_{i=1}^N P_{i,c}$$

$$\mu_{y,c} = \mu_{x,c}$$

Dan karena blok gambar sesudah kompresi adalah blok dengan satu warna saja, dapat dipastikan bahwa  $\sigma_{y,c}^2 = 0$  dan  $\sigma_{xy,c} = 0$ . Sehingga rumus SSIM dapat disederhanakan menjadi:

$$SSIM_c(x, y) = \frac{(2\mu_{x,c}\mu_{x,c} + C_1)(2*0 + C_2)}{(\mu_{x,c}^2 + \mu_{x,c}^2 + C_1)(\sigma_{x,c}^2 + 0 + C_2)}$$

$$SSIM_c(x, y) = \frac{(2\mu_{x,c}^2 + C_1)(C_2)}{(2\mu_{x,c}^2 + C_1)(\sigma_{x,c}^2 + C_2)}$$

$$SSIM_c(x, y) = \frac{C_2}{\sigma_{x,c}^2 + C_2}$$

Karena setiap kanal pada gambar diasumsikan sebesar 8-bit,

$$C_2 = (0.03 * 2^8)^2 = 58.5225, \text{ sehingga:}$$

$$SSIM_c(x, y) = \frac{58.5225}{\sigma_{x,c}^2 + 58.5225}$$

## C. GIF

Visualisasi proses pembentukan QuadTree dalam kompresi gambar menggunakan GIF cukup sederhana. GIF cukup menampilkan satu gambar untuk setiap kedalaman tree, di mana setiap gambar ke-i memperlihatkan pewarnaan blok berdasarkan node-node QuadTree pada kedalaman ke-i.

Pseudo-code pembuatan GIF:

```
function GetImageAtDepth(image, depth):
    if node is leaf or node is at depth:
        NColor = normalize all of the pixels in the node
        Add NColor of node size to image
    else
        node.children[0].GetImageAtDepth(image, depth)
        node.children[1].GetImageAtDepth(image, depth)
        node.children[2].GetImageAtDepth(image, depth)
        node.children[3].GetImageAtDepth(image, depth)
```

```
New GIF
for i in range(0, max_tree_depth):
    New image
    tree_root.GetImageAtDepth(image, i)
    Add image to GIF
```

## VI. Lampiran

Tautan repositori: [https://github.com/Staryo40/Tucil2\\_13523034\\_13523100](https://github.com/Staryo40/Tucil2_13523034_13523100)

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. <b>[Bonus]</b> Implementasi persentase kompresi sebagai parameter tambahan	✓	
6. <b>[Bonus]</b> Implementasi Structural Similarity Index (SSIM) sebagai metode pengukuran error	✓	
7. <b>[Bonus]</b> Output berupa GIF Visualisasi Proses pembentukan Quadtree dalam Kompresi Gambar	✓	
8. Program dan laporan dibuat (kelompok) sendiri	✓	