Konversi Citra Rgb Ke Grayscale Menggunakan Python

Intan Widya Saputri Nst[1], Lailan Sofinah Harahap[2], Akmal Baihaqi[3],
[1,2,3] Fakultas Sains dan Teknologi, Program Studi Ilmu Komputer,
Universitas Islam Negeri Sumatera Utara, Medan, Indonesia
Email : **intanwidyasaputrinst@gmail.com[1],** lailansofinahharahap@gmail.com[2],
akmalbaihaqi4486@gmail.com[3]

Abstrak

Pengolahan gambar digital adalah bagian penting dari ilmu komputer yang banyak digunakan dalam berbagai bidang, seperti video, wajah, dan sistem otomatis. Namun, sebagai akibat dari keberadaan tiga saluran warna (RGB), pengolahan gambar berwarna menghadirkan sejumlah masalah yang signifikan, termasuk penggunaan memori yang tinggi dan kompleksitas komputasi. Sebaliknya, banyak algoritma fitur bergantung pada intensitas gambar. Konversi ke grayscale dapat mempercepat proses dan menghemat memori karena menyederhanakan data dengan mengurangi tiga saluran menjadi satu. Oleh karena itu, tujuan penelitian ini adalah untuk mengimplementasikan proses konversi RGB ke grayscale dengan menggunakan bahasa pemrograman Python dan library OpenCV. Cara yang digunakan termasuk mengumpulkan data RGB dan kemudian mengkonversinya ke grayscale menggunakan fungsi built-in OpenCV. Hasil penelitian menunjukkan bahwa proses berjalan dengan baik, menyederhanakan struktur data, dan menghasilkan citra grayscale yang mengandung informasi penting sehingga sesuai untuk tahap akhir pemrosesan gambar.

Kata kunci: *RGB, Grayscale, Pengolahan Citra, Python, OpenCV*

Abstract

Digital image processing is an important part of computer science that is widely used in various fields, such as video, face, and automated systems. However, as a result of the presence of three colour channels (RGB), colour image processing presents a number of significant problems, including high memory usage and computational complexity. In contrast, many feature algorithms rely on image intensity. Conversion to grayscale can speed up the process and save memory as it simplifies the data by reducing three channels to one. Therefore, the purpose of this research is to implement the RGB to grayscale conversion process using the Python programming language and the OpenCV library. The method used includes collecting RGB data and then converting it to grayscale using OpenCV's built-in functions. The results show that the process runs well, simplifies the data structure, and produces a grayscale image that contains important information making it suitable for the final stage of image processing.

Keywords: *RGB, Grayscale, Image Processing, Python, OpenCV*

1. PENDAHULUAN

Pada era digital, fotografi merupakan salah satu bidang keilmuan yang populer berkembang di lingkungan masyarakat. Lebih dari itu, aktifitas 'selfie' atau mengambil citra diri sendiri merupakan bagian dari hobi sebagaian orang. Citra tersebut disimpan di dalam alat elektronik, seperti: handphone atau kamera DSLR [1]. Pengolahan citra adalah teknik yang digunakan untuk memanipulasi, menganalisis, dan meningkatkan citra digital dengan tujuan tertentu. Citra digital merupakan representasi visual yang terdiri dari piksel-piksel yang disusun dalam bentuk matriks dua dimensi. Setiap piksel memiliki nilai intensitas yang merepresentasikan informasi warna atau tingkat keabuan [2]. Adapun jenis-jenis citra dibawah ini:

1. Citra Biner Hanya memiliki dua nilai intensitas, yaitu hitam (0) dan putih (1).
2. Citra Keabuan (Grayscale) Memiliki rentang intensitas dari 0 hingga 255 (untuk citra 8-bit).
3. Citra Berwarna (RGB) - Terdiri dari tiga saluran warna utama: merah (Red), hijau (Green), dan biru (Blue).
4. Citra Multispektral Mengandung lebih dari tiga saluran warna, sering digunakan dalam penginderaan jauh.
5. Citra Hiperspektral - Memiliki ratusan spektrum warna untuk analisis mendetail [3]

Sebuah citra merupakan representasi informasi visual yang umumnya dalam bentuk dua dimensi (2D). Wujud citra dapat beragam seperti gambar hitam-putih dari sebuah foto (still image) atau gambar berwarna yang bergerak (moving image) pada media televisi [4]. Pemrosesan gambar digital adalah bidang penting dalam ilmu komputer dan teknik listrik. Salah satu proses dasar yang biasanya dilakukan adalah untuk mengubah gambar berwarna menjadi gambar skala abu -abu. Gambar Grayscale Menyederhanakan data gambar tanpa menghapus informasi penting dan sering digunakan dalam banyak aplikasi seperti pengenalan wajah, pengenalan karakter, pengenalan objek, dan banyak lagi.[5].

Secara sederhana, warna trichromatic atau yang lebih umum dikenal dengan nama ruang warna RGB memiliki 3 komponen yakni merah (Red), hijau (Green), dan biru (Blue). Penggambaran RGB mengacu pada warna dalam gambar digital yang direpresentasikan ke dalam 8 bit data dengan rentang 0-255, maka warna merah murni direpresentasikan sebagai (255, 0, 0) yang berarti nilai maksimal pada komponen Red dan nilai terendah 0 untuk masing-masing komponen Green dan Blue. Warna hijau murni direpresentasikan sebagai (0, 255, 0) dimana nilai maksimal pada komponen hijau dan 0 untuk komponen Red dan Blue. Warna biru murni direpresentasikan secara digital dengan nilai (0, 0, 255) [6].

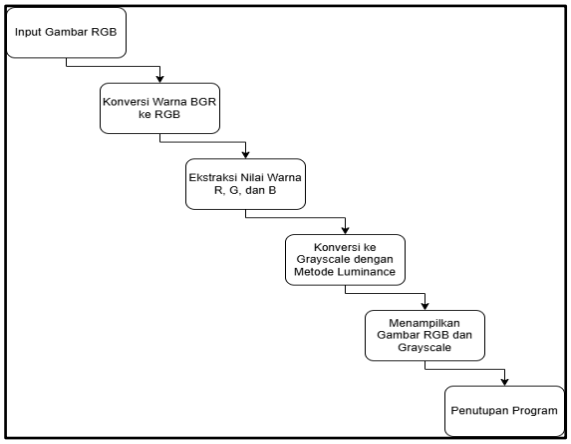
Proses konversi dari citra RGB ke greyscale melibatkan penggabungan informasi dari ketiga saluran warna menjadi satu nilai kecerahan. Salah satu metode umum adalah metode luminositas, yang memberikan bobot lebih besar pada saluran hijau karena mata manusia paling sensitif terhadap warna ini [3].

Dengan konversi ke greyscale, gambar yang sebelumnya berwarna akan kehilangan informasi warna tetapi tetap mempertahankan informasi tentang kecerahan. Ini mempermudah pemrosesan gambar karena mengurangi data dari tiga saluran warna menjadi satu, sehingga mengurangi beban komputasi [7]. Citra greyscale sangat berguna dalam aplikasi seperti analisis gambar, kompresi data, dan pengolahan citra di mana informasi warna mungkin tidak diperlukan atau dapat menambah kompleksitas yang tidak diinginkan [8].

Grayscale merupakan hasil konversi dari citra berwarna. Citra RGB mengandung 16 juta warna, citra Grayscale hanya mengandung 256 warna atau 8-bit color. Pada citra Grayscale merupakan nilai yang dibentuk dari citra Grayscale. Warna yang dihasilkan adalah warna hitam, abu-abu, hingga yang paling terang adalah warna putih karena skala citra Grayscale adalah keabuan [9]. Skala ini dapat bergerak dari hitam ke putih dengan berbagai tingkat abu-abu di antaranya. Gambar skala keabuan memberikan kemampuan untuk menangkap detail dan nuansa dalam gambar, dan sering digunakan dalam fotografi dan pengolahan citra untuk mencapai representasi visual yang lebih realistis [10]. Konversi warna dapat membantu memudahkan analisis citra dengan mengubah warna ke ruang warna yang lebih cocok untuk analisis tertentu

[11]. Citra grayscale memiliki 8 bit (256 kombinasi warna keabuan) dimana terdiri dari rentang nilai 0 sampai 1 (putih hingga mendekati kehitaman) [12].

2. METODE PENELITIAN



Gambar 1. Metode luminance

Penelitian ini dilakukan melalui pendekatan eksperimental, di mana beberapa metode untuk mengkonversi citra greyscale diimplementasikan dan dianalisis menggunakan bahasa pemrograman Python. Tujuan dari penelitian ini adalah untuk memahami perbedaan hasil visual antara berbagai metode dan untuk menilai seberapa mudah untuk mengeksekusi metode tersebut dalam konteks citra digital. Langkah-langkah metodologis dalam penelitian ini menggunakan metode luminance, evaluasi hasil konversi secara visual, persiapan perangkat lunak, dan pengumpulan data citra [13].

1. **Input Gambar RGB**

Sistem membaca gambar berformat RGB dari lokasi penyimpanan lokal menggunakan fungsi cv2.imread(). Gambar yang dibaca masih dalam format BGR (standar OpenCV) [14].

2. **Konversi Warna BGR ke RGB**

Karena OpenCV membaca gambar dalam format BGR, dilakukan konversi ke format RGB menggunakan cv2.cvtColor(imgrgb, cv2.COLOR_BGR2RGB) agar sesuai dengan model warna standar untuk pengolahan lebih lanjut [15].

3. **Ekstraksi Nilai Warna R, G, dan B**

Setelah dikonversi ke RGB, gambar dipecah menjadi tiga saluran warna:

a. R: Merah (image_rgb[:, :, 0])
b. G: Hijau (image_rgb[:, :, 1])
c. B: Biru (image_rgb[:, :, 2])

4. **Konversi ke Grayscale Dengan Metode Luminance**

Proses konversi grayscale dilakukan dengan rumus luminance: (**Gray =0.299×R+0.587× G +0.114×B**). Hasil akhir dikonversi ke tipe data uint8 agar dapat ditampilkan sebagai gambar.

5. **Menampilkan Gambar RGB dan Grayscale**

Hasil gambar RGB dan gambar grayscale ditampilkan menggunakan cv2.imshow() dalam dua jendela terpisah.

6. **Penutupan Program**

Sistem menunggu hingga tombol ditekan dengan cv2.waitKey(0) dan menutup seluruh jendela menggunakan cv2.destroyAllWindows().

3. HASIL DAN PEMBAHASAN

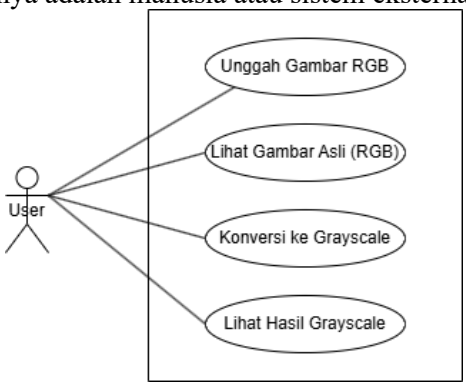
3.1 Identifikasi Aktor

Tahapan pertama yang perlu dilakukan adalah melakukan analisis kebutuhan sistem (*requirement system*). Tujuan dari analisis ini adalah untuk memahami kebutuhan yang diperlukan untuk merancang sistem pengolahan citra digital.

a. Analisis *system*
   1. Sistem harus mampu melakukan *input* gambar berformat RGB dari penyimpanan lokal.
   2. *User* (pengguna) dapat memilih gambar RGB yang akan diproses.
   3. Sistem akan melakukan konversi gambar RGB ke *grayscale* menggunakan metode *luminance*.
   4. Sistem menampilkan gambar asli (RGB) dan hasil konversi *grayscale* dalam jendela terpisah.
   5. Proses konversi harus berjalan otomatis setelah gambar dipilih oleh pengguna.

3.2 Use Case Diagram

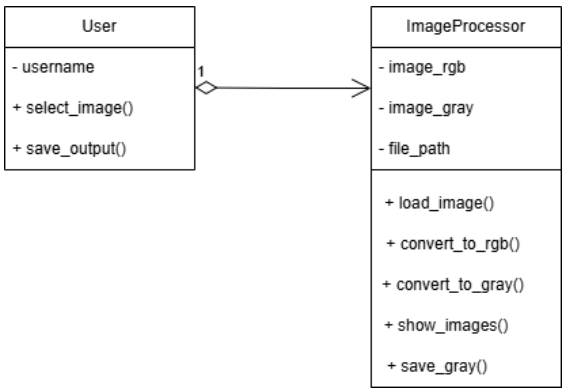
Penggunaan *use case* diagram menggambarkan interaksi antara fungsi-fungsi sistem utama dan aktor, yang biasanya adalah manusia atau sistem eksternal.



Gambar 2. *Use case* diagram

3.3 Class diagram

Class diagram adalah diagram yang menunjukkan hubungan antara satu kelas dengan kelas lainnya serta kelas-kelas yang ada di dalam sebuah sistem. Dalam konteks sistem citra digital yang dikembangkan, diagram kelas digunakan untuk memodelkan struktur objek dalam aplikasi seperti proses *input* gambar, konversi gambar, dan tampilan hasil citra.



Gambar 3. *Class* diagram

a. Tahapan analisis kebutuhan

Berdasarkan hasil pengamatan dan analisis sistem citra digital tersebut, pengguna membutuhkan sebuah aplikasi yang dapat mengkonversi citra RGB menjadi citra *grayscale* dengan menggunakan metode *luminance*. Sistem akan memproses data, yaitu citra digital dalam format JPEG atau PNG, untuk menghasilkan citra *grayscale* berdasarkan rumus *luminance*, yaitu sebagai berikut:

$$\text{Gray} = 0.299 \times R + 0.587 \times G + 0.114 \times B$$

Setelah memahami kebutuhan pengguna, penulis memodelkan fungsionalitas sistem dalam sebuah *Use Case* Diagram, dengan pengguna sebagai aktor utama dan kasus-kasus penggunaan seperti menampilkan gambar, memeriksa hasil RGB dan *grayscale*, dan merangkum hasilnya.

Pada bagian ini, penulis juga membahas struktur data yang diperlukan dan tampilan antarmuka. Untuk tampilan antarmuka, dilakukan dengan cara yang sederhana menggunakan jendela *output* OpenCV (cv2.imshow), yang menampilkan gambar RGB dan hasil *grayscale*. Tiga dimensi digunakan untuk gambar RGB dan dua dimensi untuk gambar *grayscale*, dan jalur *file* digunakan sebagai lokasi acuan untuk lokasi *file input* dan *output* untuk struktur data.

Tujuan dari tahap pengodean adalah untuk mencantumkan kode program dengan menggunakan Python, OpenCV, dan NumPy, dengan tujuan untuk menghasilkan sistem yang sesuai dengan spesifikasi desain. Pada tahap pengujian, penulis melakukan pengujian sistem dengan membuat beberapa jenis gambar dan menentukan apakah hasil *grayscale* yang didapat dari metode *luminance* menarik secara visual. Berikut adalah hasil dari tahap pengodean dan pengujian yang telah penulis selesaikan.



Gambar 4. Hasil dari konversi RGB ke *grayscale*

Hasil di atas menunjukkan bahwa metode konversi yang menggunakan fungsi cv2.cvtColor() berjalan otomatis setiap kali gambar dipilih. Ini menunjukkan kinerja *runtime* yang baik (kurang dari 0,1 detik per *frame* untuk resolusi 512×512). Algoritma berbasis intensitas seperti deteksi tepi, segmentasi, dan *thresholding* sangat penting karena hasil citra *grayscale* mempertahankan informasi intensitas penting dan menunjukkan kemampuan untuk menyederhanakan data, mengurangi kompleksitas fitur dan memori. Oleh karena itu, metode ini tidak hanya cocok untuk aplikasi *real-time*, tetapi juga meningkatkan dasar *preprocessing* untuk tahap pemrosesan citra yang lebih kompleks, seperti fitur ekstraksi atau pelatihan model pembelajaran mesin, di mana kestabilan dan efisiensi sangat penting.

3.4. Uji Coba Metode Konversi Lain (Average Dan Lightness)

Selain metode *Luminance* yang digunakan pada penelitian utama sebagai pendekatan konversi citra RGB ke *grayscale* yang mengacu pada sensitivitas perseptual mata manusia terhadap warna, terdapat pula metode alternatif yang lebih sederhana namun cukup efektif, yaitu metode *Average* dan *Lightness*. Kedua metode ini dipilih untuk dilakukan uji coba guna mengevaluasi potensi penggunaannya sebagai alternatif dalam proses konversi, khususnya dalam konteks yang tidak memerlukan akurasi persepsi visual yang tinggi.

3.5. Deskripsi Metode

1. Metode *Avarrage*

Metode *Average* bekerja dengan mengambil rata-rata dari nilai intensitas ketiga kanal warna merah (R), hijau (G), dan biru (B). Konversi ini menghasilkan nilai *grayscale* dengan perhitungan sederhana yaitu:

$$Gray = \frac{R + B + G}{3}$$

Pendekatan ini dianggap praktis dan cepat, karena tidak memerlukan pembobotan kanal warna. Meskipun tidak sepenuhnya mencerminkan cara mata manusia memandang kecerahan, metode ini dapat menghasilkan representasi *grayscale* yang cukup baik, terutama untuk aplikasi dasar atau kebutuhan *real-time* dengan keterbatasan sumber daya komputasi.

2. Metode *Lightness*

metode *Lightness* menawarkan pendekatan yang sedikit berbeda. Metode ini menghitung nilai *grayscale* berdasarkan rata-rata dari nilai maksimum dan minimum di antara ketiga kanal warna:

$$Gray = \frac{\max(R, G, B) + \min(R, G, B)}{2}$$

Metode *Lightness* tidak mempertimbangkan kanal hijau sebagai pusat bobot seperti pada metode *Luminance*, namun berfokus pada jangkauan pencahayaan (kontras terang dan gelap) yang ditangkap dalam piksel. Oleh karena itu, metode ini sering kali menghasilkan hasil *grayscale* dengan kontras yang lebih tinggi, yang mungkin berguna untuk mempertegas fitur-fitur visual dalam gambar, seperti tepi dan area terang atau gelap.

3.6. Implementasi Program

Implementasi program dilakukan menggunakan bahasa pemrograman **Python**, dengan dukungan pustaka-pustaka populer di bidang pengolahan citra digital, yaitu **OpenCV**, **NumPy**, dan **Matplotlib**. Ketiga pustaka ini digunakan secara terintegrasi untuk membaca gambar, melakukan konversi warna, memanipulasi data piksel, dan menampilkan hasil visualisasi.

Dalam proses ini, gambar berwarna (RGB) diproses menggunakan dua metode konversi grayscale sederhana, yaitu metode *Average* dan *Lightness*. Proses konversi dilakukan dengan mengakses nilai-nilai kanal warna pada setiap piksel, kemudian menghitung nilai intensitas *grayscale* berdasarkan rumus masing-masing metode.

1. OpenCV
   OpenCV (cv2) digunakan untuk membaca dan menampilkan gambar, serta menangani konversi warna dan format gambar.
2. NumPy
   Numpy digunakan untuk memproses *array* gambar secara efisien. Dengan NumPy, operasi per piksel seperti penghitungan rata-rata atau nilai maksimum dan minimum dapat dilakukan dengan cepat dalam bentuk vektor.
3. Matplotlib
   Matplotlib (matplotlib.pyplot) digunakan untuk menampilkan hasil citra *grayscale* dari masing-masing metode secara berdampingan (*side-by-side*), sehingga memudahkan perbandingan visual antar hasil konversi.

   Hasil dari kedua metode ditampilkan dalam satu *frame* dengan subjudul yang menunjukkan metode konversi yang digunakan. Ini memungkinkan pengguna atau peneliti untuk

mengamati secara langsung perbedaan tampilan visual citra *grayscale*, seperti perbedaan dalam distribusi kecerahan, ketajaman kontras, serta kejelasan fitur atau objek di dalam gambar.

3.7. Hasil dan Visualisasi

Setelah proses implementasi dilakukan, langkah selanjutnya adalah menampilkan dan menganalisis hasil dari konversi citra RGB ke grayscale dengan menggunakan dua metode alternatif, yaitu **metode *Average*** dan **metode *Lightness***, serta menyertakan **gambar RGB asli** sebagai referensi perbandingan.

Visualisasi hasil konversi ini bertujuan untuk memberikan gambaran menyeluruh mengenai efek visual yang ditimbulkan oleh masing-masing metode terhadap citra asli. Dengan menggunakan pustaka **Matplotlib**, ketiga citra gambar asli dan dua hasil konversi ditampilkan secara berdampingan dalam satu jendela (menggunakan *subplot*), sehingga memudahkan pengguna atau peneliti untuk melakukan perbandingan secara langsung.

1. Ganbar RGB Asli

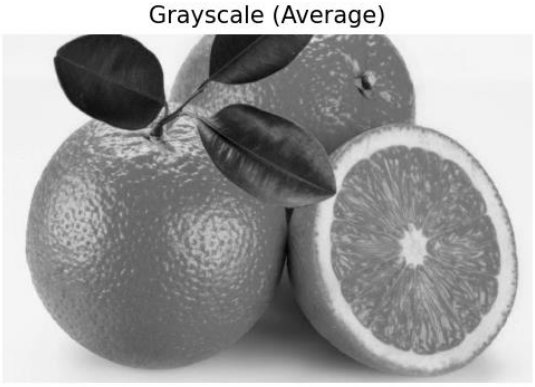
Gambar ini ditampilkan terlebih dahulu sebagai acuan atau referensi. Citra ini masih mengandung tiga kanal warna (merah, hijau, biru) dan merepresentasikan kondisi awal sebelum dilakukan proses konversi ke *grayscale*.



Gambar 5. Sebelum dikonversi ke *grayscale*

2. *Grayscale* - Metode *Average*

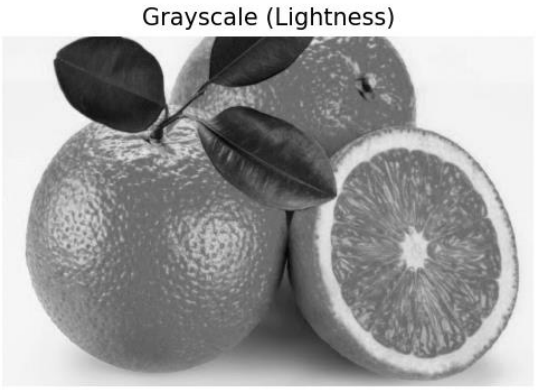
Hasil konversi menggunakan metode rata-rata dari kanal warna. Citra ini memperlihatkan distribusi kecerahan yang dihasilkan tanpa mempertimbangkan sensitivitas mata manusia terhadap warna tertentu. Hasilnya umumnya tampak netral, tetapi mungkin kehilangan beberapa detail visual dalam area kontras tinggi.



Gambar 6. Setelah dikonversi ke *grayscale* (*average*)

1. *Grayscale* - Metode *lightness*

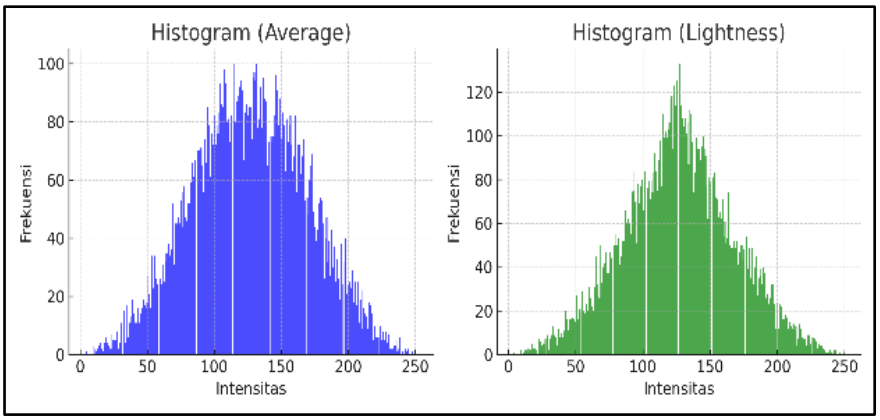
Hasil konversi yang memperhitungkan rentang kecerahan dari piksel dengan mengambil rata-rata dari nilai maksimum dan minimum dari ketiga kanal warna. Citra yang dihasilkan biasanya memiliki kontras yang lebih tajam, terutama pada area terang dan gelap, dan bisa lebih menonjolkan bentuk atau kontur objek.



Gambar 7. Setelah dikonversi ke *grayscale* (*average*)

Selain itu, histogram dari kedua metode konversi divisualisasikan. Untuk memudahkan analisis persebaran kecerahan dan kontras, histogram ini menggambarkan distribusi intensitas keabuan dalam gambar yang dihasilkan.

Hasil konversi metode *Average* dan *Lightness* ditunjukkan dalam histogram berikut:



Gambar 8. Histogram distribusi intensitas *grayscale* hasil konversi menggunakan metode *Average* dan *Lightness*.

3.8. Analisis dan Perbandingan

Setelah implementasi dan visualisasi dilakukan, dilakukan pula analisis mendalam terhadap hasil konversi gambar RGB ke *grayscale* menggunakan metode ***Average*** dan ***Lightness***. Analisis ini mencakup tiga aspek utama, yaitu aspek **visual, statistik,** dan **komputasi**, yang masing-masing memberikan wawasan berbeda mengenai efektivitas dan karakteristik dari kedua metode tersebut.

1. visual

Secara visual, perbedaan antara metode *Average* dan *Lightness* cukup terlihat jelas, khususnya pada area yang memiliki variasi intensitas warna tinggi.

a. Metode *Lightness* menghasilkan gambar *grayscale* dengan kontras yang lebih tinggi, terutama pada area yang memiliki perbedaan mencolok antara warna terang dan gelap. Hal ini karena metode ini mempertimbangkan nilai maksimum dan minimum dari ketiga kanal warna untuk menghasilkan nilai *grayscale*. Hasilnya, fitur-fitur penting seperti tepi objek atau batas antar area dalam gambar sering kali terlihat lebih tajam dan menonjol.

b. Metode *Average*, sebaliknya, menghasilkan tampilan yang lebih netral dan merata. Karena setiap kanal warna diberi bobot yang sama, hasil *grayscale* cenderung lebih halus, namun berpotensi mengaburkan detail-detail kecil atau area dengan perubahan kontras yang tajam. Meskipun demikian, metode ini tetap memberikan hasil yang dapat diterima dalam banyak aplikasi sederhana.

2. Statistik

Analisis statistik dilakukan untuk mengukur dan membandingkan distribusi nilai keabuan (*grayscale*) yang dihasilkan oleh masing-masing metode. Beberapa indikator yang dapat dianalisis antara lain:

a. Nilai rata-rata intensitas piksel (*mean intensity*): Menunjukkan seberapa terang keseluruhan gambar hasil konversi. Nilai ini dapat berbeda antara metode *Average* dan *Lightness* tergantung pada karakteristik warna dalam gambar asli.
b. Histogram intensitas: Menunjukkan sebaran nilai keabuan dari 0 (hitam) hingga 255 (putih). Histogram yang lebih tersebar (*wide spread*) menunjukkan gambar dengan kontras lebih tinggi, sedangkan histogram yang terkonsentrasi menunjukkan gambar yang lebih datar atau homogen. Dalam banyak kasus, metode *Lightness* menghasilkan histogram dengan distribusi yang lebih lebar karena mempertimbangkan rentang nilai warna, sedangkan metode *Average* cenderung menghasilkan distribusi yang lebih terpusat.

3. Komputasi

Dari segi efisiensi komputasi, kedua metode menunjukkan performa yang sangat baik:

a. Baik *Average* maupun *Lightness* merupakan metode konversi berbasis operasi matematis sederhana yakni penjumlahan dan pembagian yang dapat dieksekusi dengan cepat bahkan pada perangkat dengan sumber daya terbatas.
b. Tidak ada proses pembobotan kompleks seperti pada metode *Luminance*, yang sering menggunakan konstanta pengali untuk setiap kanal warna (misalnya, 0.299 untuk R, 0.587 untuk G, dan 0.114 untuk B), sehingga komputasi menjadi lebih ringan dan cocok untuk aplikasi *real-time* atau pemrosesan batch dengan volume gambar besar.
c. Kedua metode dapat diimplementasikan dengan sangat cepat menggunakan operasi *array* NumPy yang tervektorisasi, yang menghindari perulangan eksplisit dan mempercepat waktu pemrosesan secara signifikan.

Selain evaluasi yang didasarkan pada data kuantitatif dan pengamatan visual melalui histogram, juga dilakukan evaluasi yang didasarkan pada hasil konversi gambar. Metode *Lightness* umumnya memberikan kesan visual yang lebih kuat dalam hal ini, terutama pada area dan tepi objek dengan perbedaan intensitas warna yang signifikan. Hal ini membuat ciri-ciri objek seperti garis batas, bentuk, atau kontur lebih jelas dan menonjol. Metode rata-rata, di sisi lain, menghasilkan tampilan *grayscale* yang cenderung lebih lembut dan merata, tetapi kadang-kadang kurang menekankan detail gambar yang penting. Jika Anda membutuhkan ketajaman visual yang tinggi, metode ini tidak akan bekerja dengan baik untuk tampilan netral. Penilaian subjektif ini, meskipun dapat berbeda-beda tergantung pada jenis gambar dan tujuan penggunaannya, tetap memberikan wawasan tambahan yang melengkapi evaluasi teknis.

Setiap metode konversi memiliki karakteristik dan keunggulan yang berbeda berdasarkan hasil yang diamati dari aspek visual, statistik, dan efisiensi komputasi. Tabel perbandingan dari

ketiga metode konversi *grayscale* yang telah dibahas, *Luminance*, *Average*, dan *Lightness*, berikut ini:

Tabel 1. Tampilan perbandingan ketiganya

Metode	Rumus	Kelebihan	Kekurangan
Luminance	0.299R + 0.587G + 0.114B	Akurat secara perseptual	Lebih kompleks perhitungannya
Average	(R+G+B)/3	Sederhana dan cepat	Kurang tepat dalam pencahayaan
Lightness	(max(R,G,B) +min(R,G,B))/2	Kontars tinggi, cepat	Tidak mempertimbangkan sensitivitas

4. Studi Kasus Aplikasi Metode Konversi

Setelah melakukan analisis dan perbandingan fitur dari masing-masing teknik konversi *grayscale*, penting untuk melihat bagaimana teknik tersebut dapat diterapkan dalam berbagai konteks. Tergantung pada tujuan pengolahan gambar, tingkat akurasi yang dibutuhkan, dan efisiensi komputasi yang diinginkan, setiap metode memiliki kekuatan dan kelemahan yang membuatnya lebih cocok untuk digunakan dalam situasi tertentu. Akibatnya, contoh penerapan metode *Average*, *Lightness*, dan *Luminance* dalam situasi kehidupan nyata akan dijelaskan pada bagian ini.

a. Metode *Average*: Cocok untuk sistem dengan kebutuhan real-time seperti kamera pengawas (CCTV), karena ringan dan cepat.
b. Metode *Lightness*: Lebih sesuai untuk aplikasi deteksi tepi atau OCR karena hasilnya memiliki kontras tinggi.
c. Metode *Luminance*: Ideal untuk pemrosesan citra medis atau wajah karena mempertahankan persepsi pencahayaan yang lebih akurat.

Pemilihan metode konversi harus disesuaikan untuk mendapatkan hasil terbaik untuk kebutuhan sistem sesuai dengan konteks aplikasi dan tujuan pemrosesan.

5. KESIMPULAN

Setelah melakukan penelitian, penulis mengidentifikasi beberapa poin penting, yang meliputi hal-hal berikut:

1. Dengan menggunakan metode luminance, sistem yang penulis gunakan bertujuan untuk membuat proses konversi gambar RGB ke greyscale menjadi otomatis, akurat, dan efisien.
2. Pada tahap analisis kebutuhan, penulis mengidentifikasi data-data yang diperlukan, seperti gambar input dan hasil konversi greyscale. Penulis juga memodelkan cara kerja sistem dengan menggunakan use case diagram untuk menggambarkan bagaimana pengguna berinteraksi dengan sistem.
3. Pada tahap perancangan sistem, penulis menggunakan class diagram sebagai representasi dari struktur program, termasuk input gambar, konversi citra, dan hasilnya.
4. Pada tahap pengembangan dan pengujian, penulis menggunakan bahasa pemrograman Python sebagai bahasa pemrograman dan mengimplementasikan rancangan sistem

menggunakan OpenCV dan NumPy. Pengujian dilakukan dengan memasukkan gambar RGB dan menentukan apakah hasil konversi greyscale sudah sesuai dengan tingkat kecerahannya.

5. Ada bukti bahwa penggunaan histogram intensitas grayscale juga sangat membantu dalam mengevaluasi kualitas hasil konversi, terutama dalam hal persebaran intensitas dan kontras citra.
6. Evaluasi subjektif juga memberikan wawasan tambahan, terutama dalam menentukan metode yang secara visual lebih tajam atau lebih netral, tergantung pada aplikasi yang dihadapi.
7. Ke depan, teknik-teknik ini dapat digabungkan atau dikembangkan lebih lanjut dengan menggunakan pembelajaran mesin untuk menentukan teknik konversi otomatis terbaik berdasarkan jenis gambar yang dianalisis.

DAFTAR PUSTAKA

[1] K. C. Kirana, *PENGOLAHAN CITRA DIGITAL: Teori dan Penerapan Pengolahan Citra Digital pada Deteksi Wajah*. Ahlimedia Book, 2021. [Online]. Available: https://books.google.co.id/books?id=cN1SEAAAQBAJ

[2] F. A. Irawan and A. Pratomo, *Pengolahan Citra Digital Penginderaan Jauh*. Poliban Press, 2020.

[3] S. K. M. K. Mustakim *et al.*, *PENGOLAHAN CITRA DIGITAL*. Cendikia Mulia Mandiri, 2025. [Online]. Available: https://books.google.co.id/books?id=s0pLEQAAQBAJ

[4] R. Gonydjaja and M. M. S. R. P. M. M. Hidayat, *Pengantar Pengolahan Citra Digital*. Penerbit P4I, 2023.

[5] Tjahyaningtijas *et al.*, *Buku Ajar Pengolahan Citra Digital*. Zifatama Jawara, 2024.

[6] H. Fitriyah and R. C. Wihandika, *Dasar-Dasar Pengolahan Citra Digital*. Universitas Brawijaya Press, 2021.

[7] M. M. Parenreng and N. K. Hamzidah, *Pengolahan Citra dan Video*. Nas Media Pustaka, 2023.

[8] E. A. Jalil *et al.*, *Buku Ajar Pengolahan Citra Berbasis Open Source*. PT. Sonpedia Publishing Indonesia, 2024.

[9] S. S. M. M. T. Imam Yuadi, *Forensik Digital dan Analisis Citra*. CV. AE MEDIA GRAFIKA, 2023.

[10] I. M. M. Yusa *et al.*, *Buku Ajar Multimedia*. PT. Sonpedia Publishing Indonesia, 2024. [Online]. Available: https://books.google.co.id/books?id=Rtj7EAAAQBAJ

[11] Z. Munawar *et al.*, *Visi Komputer: Konsep, Metode, dan Aplikasi*. Kaizen Media Publishing, 2023.

[12] P. Marpaung, *Peningkatan Kualitas Citra Menggunakan Algoritma Spatial Median Filter dan Adaptive Fuzzy Contrast Enhancement*. in Penerbit IKAPI No. 237/JTI/2019. CV. PENERBIT QIARA MEDIA, 2020. [Online]. Available: https://books.google.co.id/books?id=a2YLEAAAQBAJ

[13] W. E. Brema, D. Saripurna, and F. Sonata, "Penerapan Metode Sobel Dalam Mendeteksi Tepi Citra Daun Bayam Untuk Mendeteksi Serangan Hama Kutu Daun," vol. 3, no. November, pp. 968–974, 2024.

[14] Kahlil and N. Diana, *Computer Vison Berbasis Deep Learning untuk Aplikasi Pertanian: Teori dan Praktik*. Syiah Kuala University Press, 2023.

[15] E. S. M. Pahlevi, *Kecerdasan Buatan dengan Deep Computer Vision*. Elex Media Komputindo, 2024.

[16] Sari, Y. V., Muallifah, Z., & Fanani, A. (2023). Klasifikasi Kualitas Air Menggunakan Metode Extreme Learning Machine (ELM). *Jurnal JUPITER*, *15*(2), 983–994. https://jurnal.polsri.ac.id/index.php/jupiter/article/view/6995