Implementasi Algoritma Daltonisasi Berbasis Transformasi Linear pada Ruang Warna LMS untuk Peningkatan Aksesibilitas Citra Digital bagi Penderita Dikromasi

Ramadhian Nabil Firdaus Gumay 13524126
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13524126@std.stei.itb.ac.id, [2]ramadhianx12@gmail.com

Abstrak—Defisiensi penglihatan warna, khususnya dikromasi (Protanopia dan Deuteranopia), menjadi tantangan signifikan dalam aksesibilitas informasi visual digital. Makalah ini membahas implementasi algoritma Daltonisasi untuk meningkatkan aksesibilitas citra bagi penderita dikromasi dengan memanfaatkan konsep Aljabar Linear, khususnya ruang vektor dan transformasi linear. Proses ini melibatkan transformasi citra RGB ke ruang warna LMS untuk mensimulasikan respons sel kerucut, memodelkan penglihatan dikromat menggunakan matriks proyeksi untuk mengidentifikasi informasi warna yang hilang, dan mendistribusikan kembali selisih tersebut ke spektrum yang dapat terlihat. Implementasi dilakukan menggunakan Python untuk memproses operasi matriks secara efisien. Hasil eksperimen menggunavektkan pelat Ishihara menunjukkan bahwa meskipun algoritma berhasil menggeser spektrum merah-hijau menjadi kontras biru-kuning, terdapat keterbatasan dalam implementasi. Analisis menunjukkan bahwa pemotongan nilai piksel (*clipping*) yang agresif dan operasi vektor dalam ruang warna non-linear menyebabkan hilangnya tekstur dan distorsi perseptual yang menekankan pentingnya penanganan numerik yang akurat dalam pengolahan citra ruang linear.

Keywords—Aljabar Linear, Buta Warna, Daltonisasi, Dikromasi, Ruang Warna LMS.

I. PENDAHULUAN

Penglihatan warna merupakan salah satu aspek fundamental dalam persepsi visual manusia yang memungkinkan identifikasi objek dan penyerapan informasi secara cepat. Dalam era digital saat ini, penyajian informasi visual melalui citra digital menjadi sangat dominan, mulai dari antarmuka pengguna (UI), visualisasi data, hingga konten hiburan. Bagi mayoritas populasi dengan penglihatan trikromat normal, membedakan spektrum warna merah, hijau, dan biru adalah hal yang intuitif. Namun, aksesibilitas informasi ini menjadi tantangan signifikan bagi individu dengan defisiensi penglihatan warna, khususnya penderita dikromasi.

Dikromasi adalah kondisi ketika retina mata hanya memiliki dua jenis sel kerucut yang berfungsi, berbeda dengan tiga jenis pada mata normal. Jenis dikromasi yang paling umum adalah Protanopia (buta warna merah) dan Deuteranopia (buta warna hijau), yang mengakibatkan kesulitan dalam membedakan kombinasi warna merah dan hijau. Hal ini menyebabkan hilangnya informasi visual yang krusial pada citra digital yang tidak dirancang dengan prinsip aksesibilitas, seperti grafik dengan kode warna atau rambu peringaltan.

Untuk mengatasi permasalahan tersebut, dikembangkan metode pemrosesan citra yang dikenal sebagai Daltonisasi. Metode ini bertujuan untuk memodifikasi komponen warna pada citra digital sehingga detail yang sebelumnya tidak terlihat oleh penderita buta warna dapat diperjelas, tanpa mengubah estetika visual secara drastis bagi pengamat normal. Proses ini pada dasarnya adalah masalah aljabar linear, di mana setiap piksel dalam citra direpresentasikan sebagai vektor dalam ruang warna.

Penerapan algoritma Daltonisasi sangat erat kaitannya dengan konsep dasar Aljabar Linear dan Geometri. Proses ini melibatkan transformasi basis vektor dari ruang warna RGB (Red-Green-Blue) yang umum digunakan pada perangkat digital, ke ruang warna LMS (Long-Medium-Short) yang merepresentasikan respons sel kerucut mata manusia. Dalam ruang vektor LMS ini, simulasi penglihatan dikromasi dapat dimodelkan menggunakan matriks proyeksi yang mereduksi dimensi informasi warna. Selanjutnya, selisih antara citra asli dan simulasi tersebut, yang merepresentasikan “error” atau informasi yang hilang, didistribusikan kembali ke spektrum warna yang dapat dilihat oleh penderita menggunakan transformasi linear.

Makalah ini membahas implementasi algoritma Daltonisasi dengan memanfaatkan operasi matriks dan transformasi linear. Fokus utama makalah ini adalah menganalisis bagaimana basis antara ruang vektor warna dan penggunaan matriks transformasi dapat meningkatkan kontras warna bagi penderita Protanopia dan Deuteranopia. Implementasi dilakukan menggunakan bahasa

pemrograman Python untuk memproses matriks piksel citra secara efisien. Diharapkan, penerapan konsep aljabar ini dapat memberikan solusi konkret dalam meningkatkan aksesibilitas teknologi bagi penyandang buta warna.

II. Landasan Teori

A. Ruang Vektor Umum

Dalam aljabar linear, konsep vektor tidak terbatas pada anak panah geometris di ruang Euclidean R^n, melainkan diperluas ke objek yang lebih abstrak dalam Ruang Vektor Umum. Misalkan V adalah sembarang himpunan benda tak kosong yang di dalamnya didefinisikan dua operasi, yaitu penjumlahan (*addition*) dan perkalian dengan skalar (*scalar multiplication*). Himpunan V disebut sebagai ruang vektor (*vector space*) jika memenuhi sepuluh aksioma berikut untuk setiap vektor $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$ dan skalar k, m:

1. jika $\mathbf{u}$ dan $\mathbf{v}$ adalah objek-objek di V, maka $\mathbf{u} + \mathbf{v}$ berada di V (Tertutup terhadap penjumlahan).
2. $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$ (Hukum komutatif).
3. $\mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}$ (Hukum asosiatif).
4. Terdapat objek 0 di V sehingga $0 + \mathbf{u} = \mathbf{u} + \mathbf{0}$ untuk semua $\mathbf{u}$ di V (Elemen identitas/vektor nol).
5. Untuk setiap $\mathbf{u}$ di V, terdapat objek $-\mathbf{u}$ di V sehingga $\mathbf{u} + (-\mathbf{u}) = 0$ (Invers aditif).
6. Jika k adalah sembarang skalar dan $\mathbf{u}$ adalah sembarang objek di V, maka $k\mathbf{u}$ berada di V (Tertutup terhadap perkalian skalar).
7. $k(\mathbf{u} + \mathbf{v}) = k\mathbf{u} + k\mathbf{v}$.
8. $(k + m)\mathbf{u} = k\mathbf{u} + m\mathbf{u}$.
9. $k(m\mathbf{u}) = (\, km)\mathbf{u}$.
10. $1\mathbf{u} = \mathbf{u}$.

Dalam konteks pengolahan citra, setiap warna dapat dianggap sebagai elemen dari ruang vektor V. Operasi pencampuran warna (penjumlahan) dan pengaturan intensitas (perkalian skalar) memenuhi aksioma-aksioma di atas, sehingga teknik-teknik aljabar linear dapat diterapkan untuk memanipulasi data warna.

B. Transformasi Linear

Transformasi linear adalah fungsi pemetaan $T : V \to W$ antara dua ruang vektor V dan W yang mempertahankan operasi penjumlahan vektor dan perkalian skalar. Suatu transformasi T dikatakan linear jika setiap vektor $\mathbf{u}, \mathbf{v} \in V$ dan skalar k, berlaku dua sifat berikut:

1. $T(\mathbf{u} + \mathbf{v}) = T(\mathbf{u}) + T(\mathbf{v})$
2. $T(k\mathbf{u}) = \mathrm{kT}(\mathbf{u})$

Dalam ruang berdimensi terhingga, setiap transformasi linear direpresentasikan sebagai perkalian matriks. Jika $V = \mathbb{R}^n$ dan $W = \mathbb{R}^m$, maka terdapat standar $\mathbf{A}$ berukuran $m \times n$ sedemikian sehingga untuk setiap $\mathbf{x} \in \mathbb{R}^n$:

$$T(\mathbf{x}) = \mathbf{A}\mathbf{x}$$

Dimana $\mathbf{A}$ adalah matriks transformasi yang kolom-kolomnya merupakan peta dari basis standar ruang vektor V.

C. Ruang Vektor Warna dan Citra Digital

Dalam pengolahan citra digital, warna direpresentasikan sebagai vektor dalam ruang dimensi tiga. Model yang paling umum digunakan adalah ruang warna RGB (Red-Green-Blue), di mana setiap piksel p dapat dinyatakan sebagai vektor kolom:

$$p = \begin{bmatrix} R \\ G \\ B \end{bmatrix} \in \mathbb{R}^3$$

dengan $R, G, B \in [0, 255]$ untuk citra 8-bit. Namun, ruang RGB didasarkan pada cara perangkat keras (monitor) memancarkan cahaya, bukan pada cara mata manusia memproses warna. Untuk mensimulasikan persepsi visual manusia, digunakan ruang warna LMS (Long-Medium-Short), yang merepresentasikan respons spektral dari tiga jenis sel kerucut pada retina manusia. Vektor dalam ruang ini didefinisikan sebagai:

$$c = \begin{bmatrix} L \\ M \\ S \end{bmatrix}$$

D. Transformasi Linear Antar Ruang Warna

Perpindahan antar ruang warna dapat dimodelkan sebagai transformasi linear menggunakan operasi perkalian matriks. Untuk mengonversi vektor warna dari ruang RGB ke LMS, digunakan matriks transformasi standar. Hubungan linear tersebut dinyatakan sebagai:

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \boldsymbol{T}_{RGB \to LMS} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Salah satu matriks konversi standar yang digunakan adalah:

$$\boldsymbol{T}_{RGB \to LMS} = \begin{bmatrix} 0.3139 & 0.6395 & 0.0465 \\ 0.1554 & 0.7579 & 0.0867 \\ 0.0178 & 0.1094 & 0.8726 \end{bmatrix}$$

Sebaliknya, untuk mengembalikan citra dari ruang LMS ke RGB, digunakan invers dari matriks tersebut ($\boldsymbol{T}^{-1}$):

$$\boldsymbol{T}_{LMS \to RGB} = \begin{bmatrix} 5.472 & -4.642 & 0.169 \\ -1.125 & 2.293 & -0.167 \\ 0.029 & -0.193 & 1.163 \end{bmatrix}$$

Sebelum transformasi ini dilakukan, biasanya

diterapkan koreksi *gamma* atau linearisasi pada nilai RGB agar operasi aljabar bersifat linear terhadap intensitas cahaya.

E. Simulasi Penglihatan Dikromasi

Dikromasi terjadi ketika satu jenis sel kerucut tidak berfungsi. Secara geometris dalam ruang LMS, ini berarti ruang warna penderita tereduksi dari dimensi tiga menjadi bidang dimensi dua. Simulasi penglihatan dikromat dilakukan dengan memproyeksikan vektor warna asli ke bidang yang didefinisikan oleh dua sel kerucut yang masih berfungsi. Operasi ini dilakukan dengan mengalikan vektor LMS asli dengan matriks simulasi **S**:

$$\boldsymbol{c}_{sim} = \mathbf{S} \times \boldsymbol{c}_{asli}$$

Untuk Protanopia (defisiensi sel L/Merah), informasi pada sumbu L hilang dan digantikan oleh proyeksi berdasarkan sumbu M dan S. Matriks simulasi $\mathbf{S}_P$ memiliki bentuk:

$$\mathbf{S}_P = \begin{bmatrix} 0 & a & b \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Sedangkan untuk Deuteranopia (defisiensi sel M/Hijau), sumbu M diproyeksikan ke bidang L dan S menggunakan matriks $\mathbf{S}_D$:

$$\mathbf{S}_D = \begin{bmatrix} 1 & 0 & 0 \\ c & 0 & d \\ 0 & 0 & 1 \end{bmatrix}$$

Nilai parameter a, b, c, d ditentukan berdasarkan garis kebingungan (*confusion lines*) yang spesifik untuk setiap jenis buta warna.

F. Algoritma Daltonisasi

Daltonisasi adalah proses redistribusi "error" warna untuk meningkatkan kontras. Algoritma ini memanfaatkan prinsip superposisi vektor. Langkah-langkah matematisnya adalah sebagai berikut:

1. Hitung selisih (*error vector*) antara citra asli dan citra simulasi dalam ruang RGB:

$$\mathbf{E} = \mathbf{p}_{\text{asli}} - \mathbf{p}_{\text{sim}} = \begin{bmatrix} R - R_{sim} \\ G - G_{sim} \\ B - B_{sim} \end{bmatrix}$$

2. Vektor **E** memuat informasi warna yang "hilang" bagi penderita dikromasi (biasanya pada spektrum merah-hijau).
3. Informasi ini dipetakan ulang (*shifted*) ke spektrum yang dapat dilihat (misalnya biru/kuning) menggunakan matriks distribusi $\mathbf{M}_{err}$:

$$\mathbf{p}_{\text{dalton}} = \mathbf{p}_{\text{asli}} + (\mathbf{M}_{\text{err}} \times \mathbf{E})$$

Hasil akhirnya, $\mathbf{p}_{\text{dalton}}$ adalah citra yang telah dimodifikasi di mana detail yang sebelumnya samar menjadi lebih kontras tanpa mengubah warna-warna lain secara drastis.

III. METODOLOGI

A. Metode Pengolahan

A.1. Representasi Citra sebagai Matriks

Sebuah citra digital berwarna dapat direpresentasikan sebagai matriks tiga dimensi berukuran $M \times N \times 3$, di mana *M* adalah tinggi citra, *N* adalah lebar citra, dan dimensi ketiga merepresentasikan kanal warna Merah, Hijau, dan Biru. Dalam pengolahan ini, setiap piksel pada koordinat (*i, j*) dipandang sebagai vektor kolom $\mathbf{p} \in \mathbb{R}^3$. Nilai intensitas setiap kanal yang awalnya berupa *integer* 8-bit [0, 255] dinormalisasi menjadi bilangan riil (*float*) dalam rentang [0, 1] untuk mempermudah operasi perkalian matriks dan menjaga presisi perhitungan.

$$\mathbf{p}_{\text{RGB}} = \frac{1}{255}\begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

A.2. Linearisi (Gamma Correction)

Ruang warna sRGB yang umum digunakan pada monitor bersifat non-linear. Sebelum melakukan transformasi linear ke ruang LMS, perlu dilakukan proses *inverse gamma correction* untuk mendapatkan nilai RGB linear. Hal ini penting agar operasi aljabar linear merepresentasikan pencampuran cahaya yang akurat secara fisik.

$$C_{linear} = \begin{cases} \dfrac{C_{sRGB}}{12.92}, & \text{jika } C_{sRGB} \leq 0.04045 \\ \left(\dfrac{C_{sRGB} + 0.055}{1.055}\right)^{2.4}, & \text{jika } C_{sRGB} > 0.04045 \end{cases}$$

Di mana *C* adalah nilai komponen *R, G,* atau *B*.

A.3. Transformasi Basis dan Simulasi

Proses ini dilakukan dalam tiga tahap perkalian matriks:

1. Vektor RGB linear ditransformasikan ke basis LMS menggunakan matriks $\mathbf{T}_{RGB \to LMS}$.
2. Vektor di ruang LMS dikalikan dengan matriks simulasi dikromasi **S** (misalnya $\mathbf{S}_{Protan}$ untuk Protanopia) yang memproyeksikan vektor warna ke bidang 2D, menghilangkan informasi pada sumbu yang defisit.
3. Selisih antara citra asli dan citra simulasi dihitung untuk mendapatkan vektor *error* **E**. Vektor ini memuat informasi warna yang hilang bagi penderita buta warna.

$$\mathbf{E} = \mathbf{p}_{asli} - \mathbf{p}_{simulasi}$$

A.4. Redistribusi Error (Daltonisasi)

Vektor error **E** kemudian didistribusikan kembali ke kanal warna yang masih dapat dilihat oleh penderita menggunakan matriks kompensasi $\mathbf{M}_{err}$.

$$\mathbf{p}_{\text{akhir}} = \mathbf{p}_{\text{asli}} + (\mathbf{M}_{\text{err}} \times \mathbf{E})$$

Hasil penjumlahan ini kemudian dikonversi kembali ke format sRGB dan dilakukan *clipping* agar nilai piksel tetap berada dalam rentang valid [0, 1] atau [0, 255].

B. Eksperimen

B.1. Deskripsi Program

Implementasi algoritma dilakukan menggunakan bahasa pemrograman Python. Python dipilih karena memiliki dukungan pustaka komputasi numerik yang kuat dan sintaks yang efisien untuk operasi matriks. Pustaka utama yang digunakan meliputi:

1. NumPy: Digunakan untuk struktur data *array* multidimensi dan operasi aljabar linear efisien. Fungsi numpy.dot() atau operator @ digunakan untuk perkalian matriks transformasi 3×3 terhadap ribuan vektor piksel simultan (*vectorized operation*).
2. Matplotlib: Digunakan untuk memuat, menampilkan, dan menyimpan hasil citra sebelum dan sesudah pemrosesan.
3. PIL: Digunakan sebagai alternatif untuk manipulasi file gambar.

B.2. Tahapan Implementasi Kode

B.2.1. Inisialisasi Pustaka dan Konfigurasi

Program diawali dengan mengimpor pustaka numpy untuk penanganan struktur data matriks dan operasi aljabar linear, matplotlib.pyplot untuk visualisasi hasil, dan PIL untuk memuat citra mentah.

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
```

Gambar 3.1. Kode Inisialisasi Pustaka

B.2.2. Deklarasi Matriks Transformasi

Seluruh matriks yang didefinisikan pada Landasan Teori diimplementasikan sebagai objek numpy.array. Matriks-matriks ini meliputi matriks konversi ruang warna ($\boldsymbol{T}_{RGB \to LMS}$), matriks simulasi dikromasi (S), dan matriks pergeseran *error* ($\boldsymbol{M}_{err}$).

```
m_rgb_to_lms = np.array([
    [0.3139, 0.6395, 0.0465],
    [0.1554, 0.7579, 0.0867],
    [0.0178, 0.1094, 0.8726]
])

m_lms_to_rgb = np.linalg.inv(m_rgb_to_lms)

m_sim_protan = np.array([
    [0.0, 2.02344, -2.52581],
    [0.0, 1.0,      0.0],
    [0.0, 0.0,      1.0]
])

m_sim_deutan = np.array([
    [1.0,      0.0, 0.0],
    [0.494207, 0.0, 1.24827],
    [0.0,      0.0, 1.0]
])

m_error_shift = np.array([
    [0.0, 0.0, 0.0],
    [0.7, 1.0, 0.0],
    [0.7, 0.0, 1.0]
])
```

Gambar 3.2. Kode Deklarasi Matriks Transformasi

B.2.3. Fungsi Pra-pemrosesan dan Operasi Matriks

Fungsi preprocess_image mengubah citra menjadi himpunan vektor normal. Fungsi apply_matrix merupakan inti dari implementasi transformasi linear, di mana persamaan $\boldsymbol{T}(\mathbf{x}) = \boldsymbol{A}\mathbf{x}$ dijalankan.

Selain itu, ditambahkan fungsi gamma_correction. Hal ini penting karena operasi aljabar linear secara fisik hanya valid pada ruang cahaya linear, sedangkan citra digital umumnya tersimpan dalam format sRGB (non-linear).

```
def preprocess_image(image_path):
    try:
        img = Image.open(image_path).convert('RGB')
        return np.array(img, dtype=float) / 255.0
    except FileNotFoundError:
        print(f"Error: File '{image_path}' tidak ditemukan.")
        return None

def apply_matrix(img_array, matrix):
    h, w, c = img_array.shape

    flat_img = img_array.reshape(-1, c)

    transformed_flat = np.dot(flat_img, matrix.T)

    return transformed_flat.reshape(h, w, c)

def gamma_correction(img_array, decode=True):
    if decode:
        mask = img_array <= 0.04045
        img_array[mask] /= 12.92
        img_array[~mask] = ((img_array[~mask] + 0.055) / 1.055) ** 2.4
    else:
        mask = img_array <= 0.0031308
        img_array[mask] *= 12.92
        img_array[~mask] = 1.055 * (img_array[~mask] ** (1/2.4)) - 0.055

    return img_array
```

Gambar 3.3. Kode Fungsi Utilitas Aljabar Linear

B.2.4. Algoritma Simulasi dan Daltonisasi

Fungsi simulate_view mensimulasikan bagaimana penderita dikromat dengan urutan: Linearisasi → Transformasi Basis LMS → Proyeksi Simulasi → Basis RGB → Gamma Encoding. Hasil simulasi ini digunakan oleh fungsi daltonize_correction untuk menghitung vektor *error* dan melakukan redistribusi warna.

```python
def simulate_view(img_array, type='protan'):
    linear_img = gamma_correction(img_array.copy(), decode=True)

    lms_img = apply_matrix(linear_img, m_rgb_to_lms)

    if type == 'protan':
        sim_lms = apply_matrix(lms_img, m_sim_protan)
    else:
        sim_lms = apply_matrix(lms_img, m_sim_deutan)

    sim_rgb = apply_matrix(sim_lms, m_lms_to_rgb)

    sim_rgb = np.clip(sim_rgb, 0, 1)
    return gamma_correction(sim_rgb, decode=False)

def daltonize(original_img, type='protan'):
    simulated_img = simulate_view(original_img, type)

    error_vector = original_img - simulated_img

    correction = apply_matrix(error_vector, m_error_shift)

    final_img = original_img + correction

    return np.clip(final_img, 0, 1)
```

Gambar 3.4. Kode Algoritma Simulasi dan Daltonisasi

B.2.5. Eksekusi Utama dan Visualisasi

Blok kode terakhir menjalankan *pipeline* dan menampilkan empat perbandingan visual, yaitu Citra Asli, Simulasi Protanopia (Sebelum Koreksi), Hasil Daltonisasi (Untuk Normal), dan Pandangan Protanopia terhadap Hasil Daltonisasi.

```python
if __name__ == "__main__":
    filename = 'test.jpg'

    print("Memproses citra...")
    original = preprocess_image(filename)

    if original is not None:
        simulated_view = simulate_view(original, type='protan')
        daltonized_result = daltonize(original, type='protan')

        daltonized_simulation = simulate_view(daltonized_result, type='protan')

        fig, axes = plt.subplots(2, 2, figsize=(12, 10))

        axes[0,0].imshow(original)
        axes[0,0].set_title("1. Citra Asli (Normal Vision)")
        axes[0,0].axis('off')

        axes[0,1].imshow(simulated_view)
        axes[0,1].set_title("2. Simulasi Protanopia (Sebelum Koreksi)")
        axes[0,1].axis('off')

        axes[1,0].imshow(daltonized_result)
        axes[1,0].set_title("3. Hasil Daltonisasi (Untuk Normal)")
        axes[1,0].axis('off')

        axes[1,1].imshow(daltonized_simulation)
        axes[1,1].set_title("4. Pandangan Protanopia terhadap Hasil Daltonisasi")
        axes[1,1].axis('off')

        plt.tight_layout()
        plt.show()
```

Gambar 3.5. Kode Program Utama

IV. PERCOBAAN DAN PEMBAHASAN

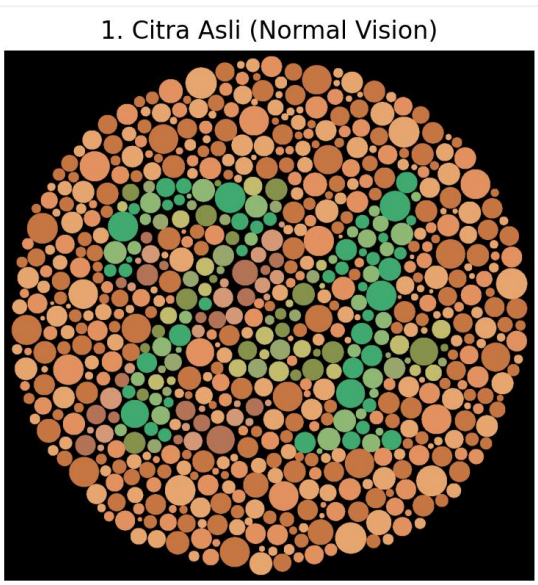
A. Deskripsi Data dan Skenario Uji

Pada eksperimen ini, digunakan citra uji standar *Ishihara Plate* (lempeng tes buta warna) yang dirancang khusus untuk mendeteksi defisiensi penglihatan warna merah-hijau. Citra ini memiliki karakteristik di mana angka atau pola tertentu disembunyikan dalam kombinasi titik-titik berwarna yang sulit dibedakan oleh penderita Protanopia maupun Deuteranopia.

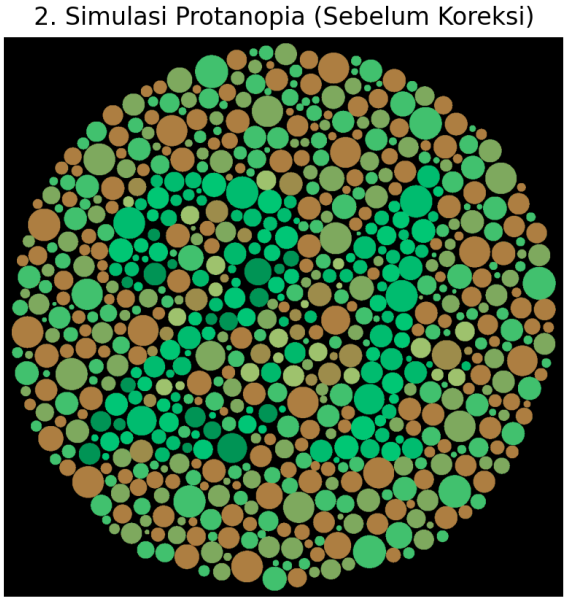
Citra masukan diproses menggunakan algoritma yang telah diimplementasikan dalam bahasa Python dengan resolusi asli dan kedalaman warna 24-bit RGB. Skenario pengujian difokuskan pada simulasi Protanopia (kebutaan warna merah), mengingat matriks simulasi $\mathbf{S}_{Protan}$ yang digunakan membuang informasi pada sumbu L (*Long-wavelength*) secara total.

B. Hasil Eksperimen

Hasil pemrosesan citra ditampilkan dalam empat panel visualisasi untuk membandingkan efektivitas algoritma.

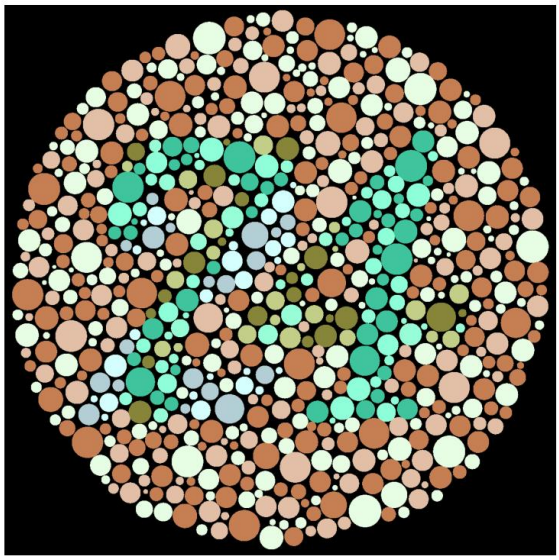


Gambar 4.1. Visualisasi Citra Asli Ishihara



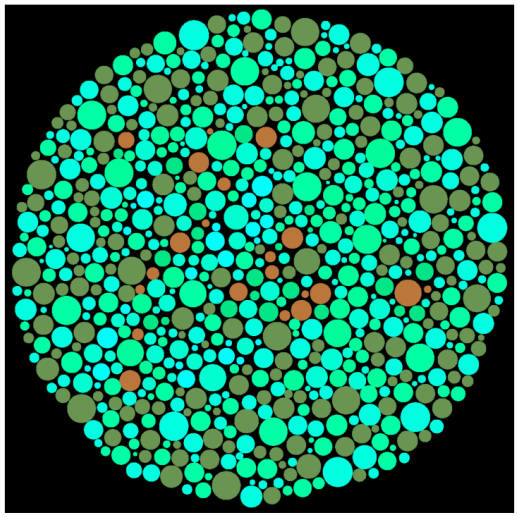
Gambar 4.2. Visualisasi Simulasi Pandangan Protanopia Sebelum Koreksi

3. Hasil Daltonisasi (Untuk Normal)



Gambar 4.3. Visualisasi Citra Hasil Daltonisasi yang Dilihat oleh Mata Normal

4. Pandangan Protanopia terhadap Hasil Daltonisasi



Gambar 4.4. Visualisasi Simulasi Pandangan Protanopia terhadap Hasil Daltonisasi

C. *Pembahasan*

Berdasarkan hasil visualisasi pada Gambar 4.3, algoritma memberikan perubahan warna yang drastis. Pola angka "74" yang pada citra asli terbentuk dari titik-titik hijau, berubah menjadi warna *Cyan* yang sangat mencolok bagi pengamat normal. Sementara itu, titik-titik latar belakang yang semua oranye/cokelat bergeser menjadi nuansa krem.

Perubahan warna ini terjadi karena operasi matriks redistribusi *error* yang diterapkan pada persamaan:

$$\mathbf{p}_{\text{dalton}} = \mathbf{p}_{\text{asli}} + (\mathbf{M}_{\text{err}} \times \mathbf{E})$$

Di mana $\mathbf{M}_{err}$ adalah matriks kompensasi yang didefinisikan sebagai:

$$\mathbf{M}_{err} = \begin{bmatrix} 0 & 0 & 0 \\ 0.7 & 1 & 0 \\ 0.7 & 0 & 1 \end{bmatrix}$$

Matriks $\mathbf{M}_{err}$ yang digunakan dalam percobaan ini mendistribusikan 70% dari selisih warna (vektor $\boldsymbol{E}$) ke kanal Biru dan Hijau. Karena pada kasus Protanopia selisih terbesar terjadi pada komponen Merah-Hijau (sumbu $L - M$), algoritma memindahkan informasi tersebut ke spektrum Biru (S-cone) yang masih berfungsi normal pada penderita. Akibatnya, area yang memiliki *error* tinggi (angka 74) mendapatkan intensitas biru yang signifikan, menghasilkan warna *cyan* pada hasil akhir.

D. *Evaluasi*

Evaluasi keberhasilan algoritma dilakukan secara kualitatif dengan membandingkan Gambar 4.2 (Visualisasi Simulasi Pandangan Protanopia Sebelum Koreksi) dengan Gambar 4.4 (Visualisasi Simulasi Pandangan Protanopia terhadap Hasil Daltonisasi).

1. Kondisi Awal (Gambar 4.2):
   Pada simulasi awal, angka "74" sulit dibedakan karena warna hijau dan oranye jatuh pada garis kebingungan yang sama pada penderita Protanopia. Akibatnya, kedua warna tersebut memiliki luminansi yang nyaris identik, sehingga angka terlihat menyatu dengan latar belakang.
2. Kondisi Setelah Koreksi (Gambar 4.4):
   Pada Gambar 4.4, hasil visual menunjukkan fenomena yang kontradiktif. Meskipun pada pandangan normal (Gambar 4.3) warna terlihat kontras, pada simulasi pandangan Protanopia, pola angka "74" justru terlihat semakin tidak jelas atau terdistorsi.
   Ketidajelasan ini kemungkinan disebabkan oleh beberapa faktor matematis:

   - Hilangnya Detail Akibat *Clipping*:
     Dalam fungsi daltonize(), operasi penambahan *error* ($p_{akhir} = p_{asli} + correction$) seringkali menghasilkan nilai piksel di atas 1.0 (atau 255), terutama pada latar belakang oranye yang sudah terang. Penggunaan fungsi np.clip(final_img, 0, 1) di akhir proses secara paksa memotong nilai-nilai tersebut menjadi putih datar. Hal ini menghilangkan tekstur dan bayangan halus pada titik-titik Ishihara yang sebenarnya krusial untuk membedakan batas objek. Akibatnya, angka dan latar belakang kehilangan definisi bentuknya.
   - Ketidakakuratan Operasi Aritmatika pada Ruang Non-Linear:
     Fungsi simulate_view mengembalikan citra yang telah dikonversi kembali ke format sRGB (*gamma-encoded*). Akibatnya,

perhitungan vektor *error* ($\mathbf{E} = p_{asli} - p_{simulasi}$) pada fungsi daltonize dilakukan dalam ruang warna non-linear. Operasi pengurangan dan penjumlahan vektor pada ruang non-linear ini menyebabkan distorsi nilai intensitas, sehingga hasil redistribusi warna tidak sepenuhnya akurat secara perseptual dibandingkan jika operasi dilakukan sepenuhnya di ruang linear sebelum konversi akhir.

V. KESIMPULAN

Berdasarkan hasil eksperimen, algoritma mampu menggeser spektrum warna merah-hijau yang bermasalah ke spektrum biru-kuning yang dapat dilihat oleh penderita dikromasi, menghasilkan citra dengan kontras warna yang baru bagi pengamat normal. Namun, evaluasi terhadap simulasi pandangan Protanopia pada citra uji Ishihara menunjukkan adanya keterbatasan implementasi. Objek angka yang diharapkan menjadi lebih jelas justru mengalami distorsi bentuk dan kehilangan detail.

Analisis teknis menunjukkan bahwa kegagalan tersebut bukan disebabkan kesalahan teori aljabar, melainkan ketidakakuratan dalam penanganan data numerik. Dua faktor utama yang teridentifikasi adalah hilangnya tekstur akibat proses pemotongan nilai piksel (*clipping*) yang agresif dan distorsi perseptual akibat operasi penjumlahan vektor yang dilakukan pada ruang warna non-linear (sRGB).

VI. LAMPIRAN

Source code yang digunakan dalam makalah ini adalah sebagai berikut:
https://github.com/Rmadhian/MakalahAlgeo.git

REFERENSI

[1] R. Munir, "Ruang Vektor Umum (Bagian 1)," Bahan Kuliah IF2123 Aljabar Linier dan Geometri, Program Studi Teknik Informatika, STEI-ITB, 2025. [Diakses 24 Desember 2025].

[2] R. Munir, "Ruang Vektor Umum (Bagian 4)," Bahan Kuliah IF2123 Aljabar Linier dan Geometri, Program Studi Teknik Informatika, STEI-ITB, 2025. [Diakses 24 Desember 2025].

[3] H. Brettel, F. Viénot, and J. D. Mollon, "Computerized simulation of color appearance for dichromats," *Journal of the Optical Society of America A*, vol.14, no. 10, pp. 2647-2655, 1997. [Diakses 24 Desember 2025].

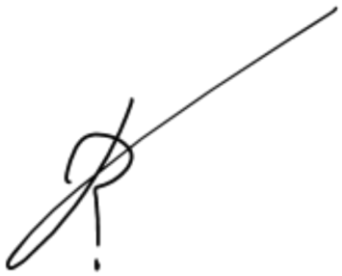
[4] F. Viénot, H. Brettel, and J. D. Mollon, "Digital video colourmaps for checking the legibility of displays by dichromats," *Color Research & Application*, vol. 24, no. 4, pp. 243-252, 1999. [Diakses 24 Desember 2025].

[5] I. B. Fidaner, P. Lin, and N. Ozguven, "Analysis of Color Blindness," Stanford University, Tech. Rep., 2005. [Diakses 24 Desember 2025].

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 24 Desember 2025



Ramadhian Nabil Firdaus Gumay 13524126