

PENGOLAHAN CITRA DIGITAL
“PENGOLAHAN CITRA BERWARNA”



OLEH :

Hairul Yasin

F55121011

A

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS TADULAKO

2023

A. Dasar warna

Kemampuan manusia untuk melihat warna disebabkan oleh cahaya yang dipantulkan oleh objek, dengan spektrum kromatis berkisar antara 400-700 nm. Istilah kromatis berarti kualitas warna cahaya yang ditentukan oleh panjang gelombang.

Karakteristik persepsi mata manusia dalam yang membedakan antara satu warna dengan warna yang lain berupa *hue*, *saturation*, dan *brightness*.

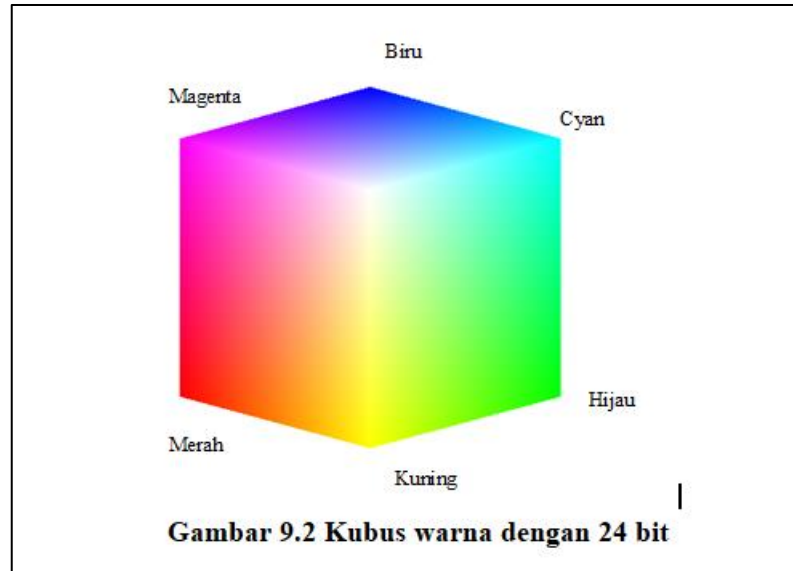
- *Hue* merujuk ke warna yang dikenal manusia, seperti merah dan hijau, yang mencerminkan warna yang ditangkap oleh mata.
- *Saturation* mengindikasikan tingkat kemurnian warna, sedangkan *brightness* mewakili intensitas pantulan objek yang diterima mata.
- *Brightness* (kecerahan) menyatakan intensitas pantulan objek yang diterima mata.

B. Ruang warna

Gonzalez & Woods (2002) mendefinisikan ruang warna sebagai sistem koordinat yang terdiri dari subruang yang mewakili setiap warna dengan satu titik di dalamnya. Tujuan dari ruang warna adalah untuk memfasilitasi spesifikasi warna dalam bentuk suatu standar. RGB adalah ruang warna yang paling dikenal pada perangkat komputer karena sesuai dengan cara manusia dalam menangkap warna. Namun, ada banyak ruang warna lainnya seperti HSI, CMY, LUV, dan YIQ yang juga telah dikembangkan.

1. Ruang Warna RGB

Ruang warna RGB umumnya digunakan pada monitor CRT dan sistem grafika komputer. Ruang warna ini terdiri dari tiga komponen dasar yaitu merah, hijau, dan biru, yang membentuk setiap piksel. Model RGB disajikan dalam bentuk kubus tiga dimensi, dengan warna merah, hijau, dan biru pada pojok sumbu dan warna hitam pada titik asal. Warna putih berada di ujung kubus yang berseberangan dengan warna hitam. Dalam resolusi 24 bit, jumlah warna yang dapat dihasilkan mencapai 16.777.216. Kubus warna nyata dapat dilihat pada Gambar 9.2.



RGB digunakan secara umum karena kemudahan perancangan hardware, tetapi ruang warna ini tidak ideal untuk beberapa aplikasi. Karena warna merah, hijau, dan biru berkorelasi erat, beberapa algoritma pemrosesan citra dapat menjadi sangat sulit. Misalnya, untuk memperoleh warna alami seperti merah, menggunakan RGB menjadi sangat kompleks karena nilai R dapat berpasangan dengan G dan B dalam nilai apa pun. Ruang warna seperti HLS atau HSV akan lebih mudah digunakan dalam hal ini.

2. Ruang Warna CMY/CMYK

Model warna CMY (*cyan, magenta, yellow*) mempunyai hubungan dengan RGB sebagai berikut:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (9.1)$$

Dalam hal ini, R, G dan B berupa nilai warna yang telah dinormalisasi, dengan jangkauan [0, 1].

Pada aplikasi printer, model warna CMY ditambahkan dengan warna hitam menjadi CMYK, dengan K menyatakan warna hitam. Hal ini dilakukan karena warna hitam dapat diambil langsung dari tinta hitam tanpa perlu mencampur dengan warna lain, dan lebih

ekonomis. Lagipula, tinta warna hitam lebih murah daripada tinta berwarna dan paling sering digunakan terutama untuk teks.



Perlu diketahui, konversi dari CMY ke CMYK dapat menggunakan berbagai cara perhitungan. Salah satu rumus yang digunakan sebagai berikut (Crane, 1997):

$$K = \min(C, M, Y) \quad (9.2)$$

$$C' = C - K \quad (9.3)$$

$$M' = M - K \quad (9.4)$$

$$Y' = Y - K \quad (9.5)$$

Dengan pendekatan seperti itu, salah satu dari C' , M' , atau Y' akan bernilai 0. Namun, ada pula yang menggunakan rumus seperti berikut (Dietrich, 2003):

$$K = \min(C, M, Y) \quad (9.6)$$

$$C = (C - K) / (1 - K) \quad (9.7)$$

$$M = (M - K) / (1 - K) \quad (9.8)$$

$$Y = (Y - K) / (1 - K) \quad (9.9)$$

3. Ruang Warna YIQ

Ruang warna YIQ dikembangkan oleh NTSC untuk sistem televisi berwarna di Amerika Serikat. Model ini terdiri dari *luma* (Y) dan *chroma* (I dan Q). Konversi YIQ dari RGB dilakukan dengan matriks koefisien dan memiliki ciri-ciri tertentu. YIQ secara statistik optimal untuk ditampilkan pada penerima TV hitam-putih. Konversi dari YIQ ke RGB juga dapat dilakukan menggunakan matriks koefisien yang berbeda. Adapun konversi RGB dari YIQ sebagai berikut:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1,000 & 0,956 & 0,621 \\ 1,000 & -0,272 & -0,647 \\ 1,000 & -1,106 & 1,703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} \quad (9.3)$$

4. Ruang Warna YCbCr

Ruang warna YCbCr biasanya digunakan pada video digital, di mana komponen Y menyatakan intensitas dan Cb dan Cr menyatakan informasi warna. Konversi dari RGB ke YCbCr dapat dilakukan menggunakan formula seperti berikut:

$$Y = 0,29900R + 0,58700G + 0,11400B \quad (9.4)$$

$$C_b = -0,16874R - 0,33126G + 0,5000B \quad (9.5)$$

$$C_r = +0,5000R - 0,41869G - 0,08131B \quad (9.6)$$

Sedangkan konversi dari YCbCr ke RGB dapat dilakukan dengan formula sebagai berikut:

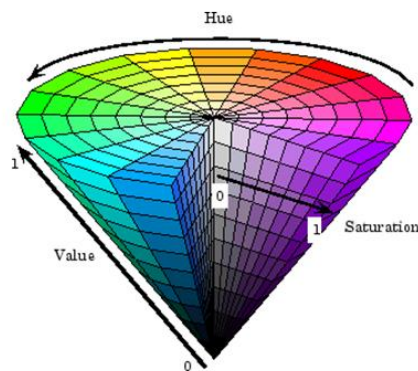
$$R = Y + 1.40200C_r \quad (9.7)$$

$$G = Y - 0,34414C_b - 0,71414C_r \quad (9.8)$$

$$B = Y + 1,77200C_b \quad (9.9)$$

5. Ruang Warna HSI, HSV, dan HSL

HSV dan HSL merupakan contoh ruang warna yang merepresentasikan warna seperti yang dilihat oleh mata manusia. H berasal dari kata “hue”, S berasal dari “saturation”, L berasal dari kata “luminance”, I berasal dari kata “intensity”, dan V berasal dari “value”.



Gambar 9.4 Ruang warna HSV
(Sumber: MATLAB)

Model HSV ditunjukkan pada gambar dan cara yang tersederhana untuk mendapatkan nilai H, S, V berdasarkan R, G, dan B, terdapat beberapa cara. Cara yang tersederhana (Acharya & Ray, 2005) adalah seperti berikut.

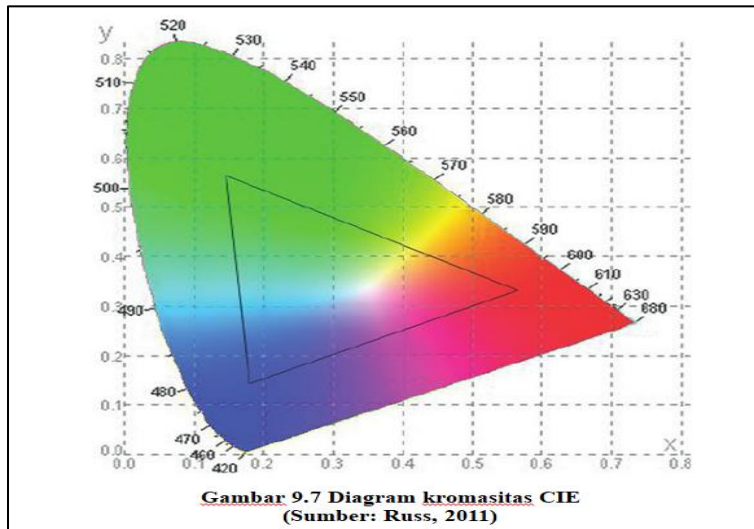
$$H = \tan \left(\frac{3(G-B)}{(R-G)+(R-B)} \right) \quad (9.10)$$

$$S = 1 - \frac{\min(R,G,B)}{V} \quad (9.11)$$

$$V = \frac{R+G+B}{3} \quad (9.12)$$

6. Ruang Warna CIELAB

CIELAB (atau CIE Lab*) adalah suatu sistem warna yang digunakan untuk merepresentasikan warna. Warna yang terletak di dalam segitiga menyatakan warna-warna umum di monitor CRT, yang dapat dihasilkan oleh komponen warna merah, hijau, dan biru.



Transformasi RGB ke CIELAB dimulai dengan melakukan perhitungan sebagai berikut:

$$X = 0,412453R + 0,357580G + 0,180423B \quad (9.40)$$

$$Y = 0,212671R + 0,715160G + 0,072169B \quad (9.41)$$

$$Z = 0,019334R + 0,119193G + 0,950227B \quad (9.42)$$

Selanjutnya, $L^*a^*b^*$ didefinisikan sebagai berikut:

$$L^* = 116f\left(\frac{Y}{Y_n}\right) - 16 \quad (9.43)$$

$$a^* = 500 \left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right) \right] \quad (9.44)$$

$$b^* = 200 \left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right) \right] \quad (9.45)$$

Dalam hal ini, $f(q)$ dihitung seperti berikut:

$$f(q) = \begin{cases} q^{\frac{1}{3}}, & \text{jika } q > 0,008856 \\ 7,787q + 16/116, & \text{untuk yang lain} \end{cases} \quad (9.46)$$

X_n, Y_n, Z_n diperoleh melalui $R=G=B=1$ dengan jangkauan R, G, B berupa $[0, 1]$.

C. Memperoleh statistika warna

Fitur warna dapat diperoleh melalui perhitungan statistis seperti rerata, deviasi standar, skewness, dan kurtosis. Fitur-fitur tersebut dapat digunakan untuk identifikasi tanaman hias.

Perhitungan dikenakan pada setiap komponen R, G , dan B . Rerata memberikan ukuran mengenai distribusidan dihitung dengan menggunakan rumus:

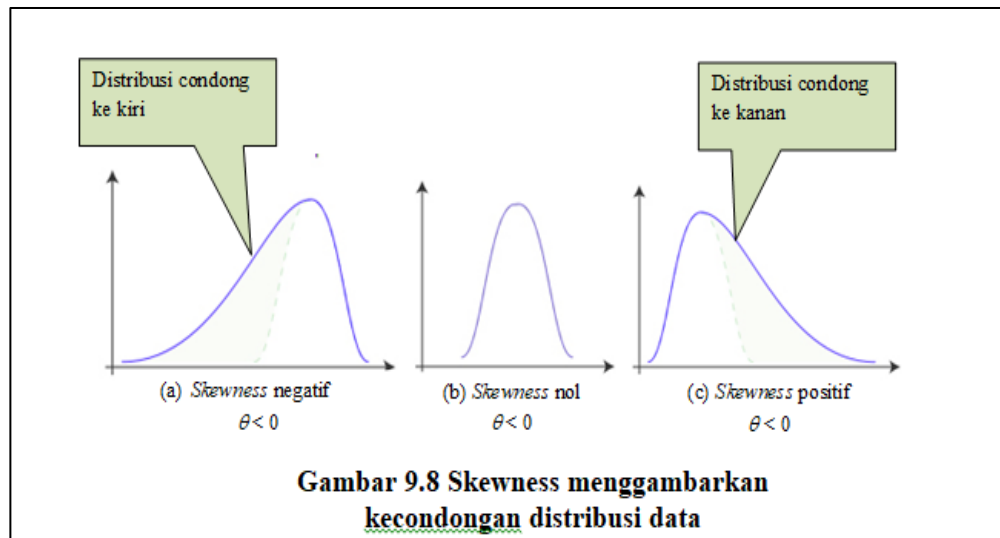
$$\mu = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N P_{ij} \quad (9.47)$$

Varians menyatakan luas sebaran distribusi. Akar kuadrat varians dinamakan sebagai deviasi standar. Adapun rumus yang digunakan untuk menghitungnya sebagai berikut:

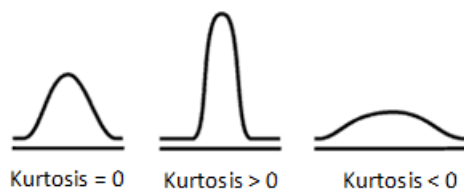
$$\sigma = \sqrt{\frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (P_{ij} - \mu)^2} \quad (9.48)$$

Skewness atau kecondongan menyatakan ukuran mengenai ketidaksimetrisan, dan kurtosis merupakan ukuran yang menunjukkan sebaran data bersifat meruncing atau menumpul. Ilustrasi *skewness* dapat dilihat pada Gambar 9.8. *Skewness* dihitung dengan cara seperti berikut:

$$\theta = \frac{\sum_{i=1}^M \sum_{j=1}^N (P_{ij} - \mu)^3}{MN\sigma^3} \quad (9.49)$$



Definisi di atas membuat distribusi normal standar memiliki kurtosis nol. Nilai positif mengindikasikan bahwa distribusi bersifat lancip dan nilai negatif menyatakan distribusi yang datar (lihat Gambar 9.9).



Gambar 9.9 Kurtosis menggambarkan keruncingan distribusi normal

D. Mengatur kecerahan dan kontras

Sebelumnya telah dijelaskan cara mengatur kontras dan kecerahan pada citra berskala keabuan. Cara seperti itu, secara prinsip dapat diterapkan pada citra berwarna. Untuk melihat efek kecerahan dan kontras, cobalah beberapa perintah berikut.

```
>> Img = imread('C:\Image\inns.png'); ⚡
>> imshow(Img) ⚡
>>
```

Kode di atas digunakan untuk melihat citra inns.png (Gambar 9.10.(a)). Selanjutnya,

```
>> C = Img + 30; ⚡
>> imshow(C) ⚡
```



```
>>
```

membuat citra dicerahkan sejauh 30. Hasilnya ditunjukkan pada Gambar 9.10(b). Lalu, cobalah kode berikut:

```
>> K = 2 * C; ⌘
```

```
>> imshow(K) ⌘
```

```
>>
```

Hasilnya dapat dilihat pada Gambar 9.10(c). Hal yang menarik dapat diperoleh dengan hanya memberikan kontras pada komponen R. Caranya:

```
>> K = C; ⌘
```

```
>> K(:, :, 1) = 2 * K(:, :, 1); ⌘
```

```
>>
```

Kode di atas mula-mula membuat K bernilai sama dengan C (efek pencerahan). Selanjutnya, $K(:, :, 1) = 2 * K(:, :, 1);$ membuat hanya komponen R saja yang dinaikkan dua kali. hasilnya ditunjukkan pada Gambar 9.10(d).



E. Menghitung jumlah warna

Berapa jumlah warna yang menyusun suatu citra? Bila terdapat kebutuhan seperti itu, jumlah warna dapat dihitung dengan memanfaatkan fungsi `jumwarna` berikut.

```
function [jumlah] = jumwarna(berkas)
% JUMWARNA Menghitung jumlah warna pada citra RGB
%     Masukan:
%         berkas berupa citra warna
%     Nilai balik berupa jumlah warna

RGB = double(imread(berkas));
[m,n,d] = size(RGB);

if (d ~= 3)
    disp('Citra harus berupa citra berwarna');
    return;
end
RGB = double(RGB);
Data = zeros(1, m * n); % Array kosong
jum = 0;
for i=1:m
    for j=1:n
        jum = jum + 1;

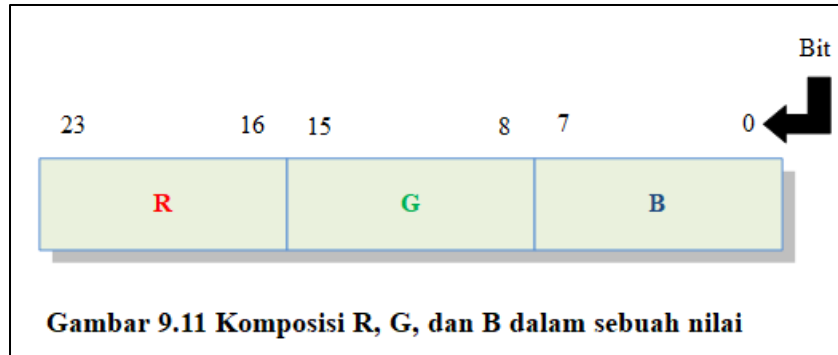
        r = RGB(i,j,1);
        g = RGB(i,j,2);
        b = RGB(i,j,3);

        Data(jum) = bitshift(r,16) + bitshift(g, 8) + b;
    end
end
% Urutkan data pada array Data
Data = sort(Data);
% Hitung jumlah warna
jwarna = 1;
for i = 1 : jum - 1
    if Data(i) ~= Data(i+1)
        jwarna = jwarna + 1;
    end
end
jumlah = jwarna;
```

Penghitungan warna dilakukan dengan mula-mula menyusun komponen R, G, dan B untuk setiap piksel menjadi sebuah nilai dengan komposisi seperti terlihat pada Gambar 9.11. Untuk keperluan seperti itu, maka:

- R perlu digeser ke kiri sebanyak 16 bit dan
- G perlu digeser ke kiri sebanyak 8 bit.

Pada *Octave* dan *MATLAB*, penggeseran bit dilakukan melalui fungsi **bitshift**.



Setelah nilai gabungan R, G, dan B terbentuk dan diletakkan ke larik Data, isi larik tersebut diurutkan. Pengurutan tersebut dimaksudkan untuk mempermudah penghitungan jumlah warna. Implementasi penghitungan pada data yang telah urut seperti berikut:

```
jwarna = 1;
for i = 1 : jum - 1
    if Data(i) ~= Data(i+1)
        jwarna = jwarna + 1;
    end
end
```

Berdasarkan kode di atas, nilai jwarna dinaikkan sebesar satu sekiranya suatu nilai dan nilai berikutnya tidak sama.

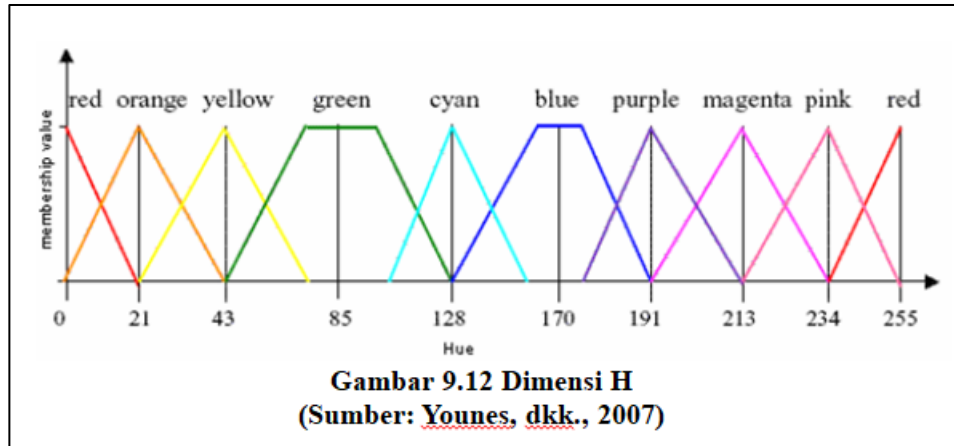
Contoh penggunaan fungsi jumwarna:

```
>> C = jumwarna('C:\Image\lapangan.png')
C = 92475
>>
```

Hasil di atas menyatakan bahwa jumlah warna yang terkandung pada lapangan.png adalah 92.475.

F. Aplikasi pencarian citra berdasarkan warna dominan.

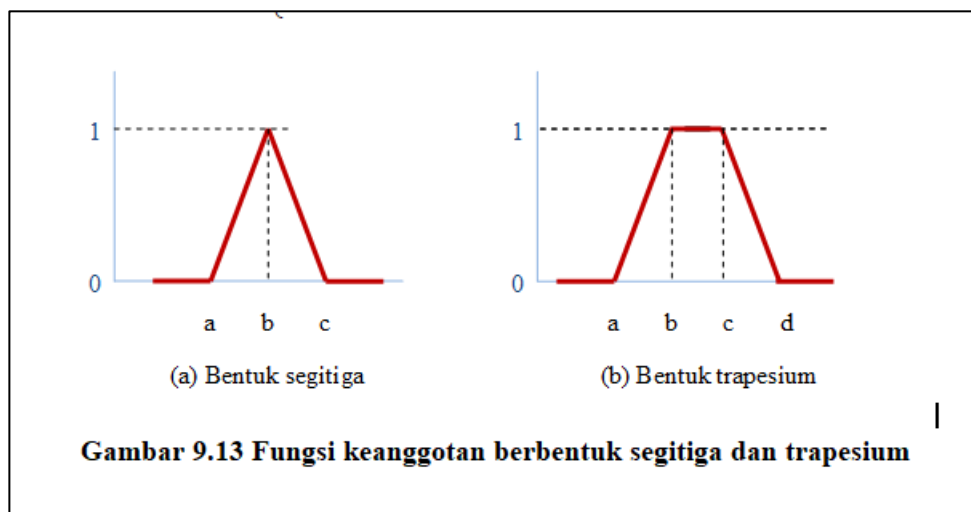
Kadir (2011d) membahas contoh aplikasi penentuan warna dominan pada daun dengan mengimplementasikan model sederhana dalam bentuk program. Untuk melakukan pencarian citra berdasarkan warna dominan seperti merah atau hijau, setiap piksel harus dipetakan ke dalam warna alamiah seperti merah atau hijau dengan menggunakan ruang warna HSV. Pada sistem HSV, komponen hue menyatakan warna seperti yang dipahami oleh manusia. *Younes, dkk.* (2007) membuat model fuzzy untuk menyatakan warna yang terlihat pada Gambar 9.12.



Berdasarkan Gambar 9.12, dimungkinkan untuk menerapkan *fuzzy logic* untuk menentukan suatu area warna beserta derajat keanggotaannya. Model tersebut didasarkan pada sumbu melingkar pada komponen *Hue* (H). Mengingat warna merah berada pada nilai H sama dengan nol, maka jangkauan warna merah berada di sekitar angka 0 dan 255.

Dua jenis fungsi keanggotaan *fuzzy* yang digunakan pada Gambar 9.12 berbentuk segitiga dan trapesium dan digambarkan kembali pada Gambar 9.13. Berdasarkan Gambar 9.13 tersebut, fungsi keanggotaan berbentuk segitiga didefinisikan sebagai berikut:

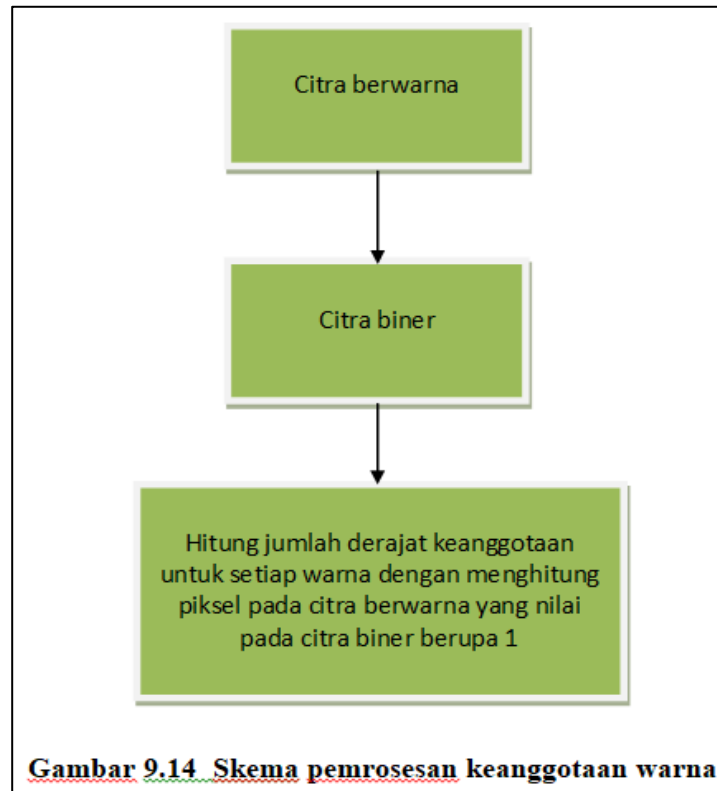
$$\Lambda(x; a, b, c) = \begin{cases} 0 & x \leq a, \\ (x - a) / (b - a) & a < x \leq b, \\ (c - x) / (c - b) & b < x \leq c, \\ 0 & x > c \end{cases} \quad (9.51)$$



Adapun fungsi keanggotaan berbentuk trapesium didefnisikan sebagai berikut:

$$\Pi(x; a, b, c) = \begin{cases} 0 & x \leq a, \\ (x - a)/(b - a) & a < x \leq b, \\ 1 & b < x \leq c, \\ (c - x)/(c - b) & c < x \leq d, \\ 0 & x > d \end{cases} \quad (9.52)$$

Penanganan khusus dilakukan untuk warna hitam, putih, dan abu-abu dengan memperhatikan komponen S dan V. Warna hitam didapatkan saat nilai V sama dengan 0, sedangkan warna putih didapatkan saat nilai S sama dengan 0. Warna abu-abu didapatkan saat nilai S rendah, dan semakin rendah nilai S maka warna abu-abu semakin tua. Untuk memasukkan setiap piksel ke dalam kategori warna seperti merah, hijau, dan lainnya, digunakan mekanisme yang terlihat pada Gambar 9.14.



Citra biner digunakan untuk menentukan area pada citra berwarna yang diproses khusus yang merupakan bagian daun. Dalam hal ini bagian yang berisi daun akan bernilai 1 pada citra biner dan 0 untuk latarbelakang. Selanjutnya, nilai H, S, V digunakan untuk menentukan kelompok warna piksel.


TRANSFORMASI GAMBAR RGB KE CMY

Project RGBtoCMY.py

```
1 import cv2
2 import numpy as np
3
4 # Load the image in RGB format
5 img = cv2.imread('lena.jpg', cv2.IMREAD_COLOR)
6
7 # Convert RGB image to CMY
8 cmy_img = 1 - (img / 255.0)
9
10 # Display the CMY image
11 cv2.imshow('CMY Image', cmy_img)
12 cv2.waitKey(0)
13 cv2.destroyAllWindows()
14
```

Commit Pull Requests Structure

CMY Image



TRANSFORMASI GAMBAR CIELAB KE RGB

```
1 import cv2
2 import numpy as np
3
4 # Memuat gambar dalam format YCbCr
5 img = cv2.imread('lena.jpg')
6 img_ycrb = cv2.cvtColor(img, cv2.COLOR_BGR2YCrCb)
7
8 # Mengambil komponen YCbCr
9 y, cb, cr = cv2.split(img_ycrb)
10
11 # Mengubah komponen YCbCr menjadi RGB
12 r = np.clip(1.402*(cr-128) + y, 0, 255).astype(np.uint8)
13 g = np.clip(-0.34414*(cb-128) - 0.71414*(cr-128) + y, 0, 255).astype(np.uint8)
14 b = np.clip(1.772*(cb-128) + y, 0, 255).astype(np.uint8)
15
16 # Menggabungkan kembali komponen RGB
17 img_rgb = cv2.merge((r, g, b))
18
19 # Menampilkan gambar asli dan hasil konversi
20 cv2.imshow('Original Image', img)
21 cv2.imshow('YCbCr Image', img_ycrb)
22 cv2.imshow('RGB Image', img_rgb)
23
24 # Menunggu tombol key ditekan dan tutup jendela
25 cv2.waitKey(0)
26 cv2.destroyAllWindows()
```

