PemrosesanCitraBerwarnaAplikasidenganJava.pdf

Pemrosesan Citra Berwarna & Aplikasi dengan Java

Book · February 2017

CITATIONS
0

READS
458

1 author:

Erwin Erwin
Universitas Sriwijaya
6 PUBLICATIONS 0 CITATIONS

SEE PROFILE


PEMROSESAN CITRA
BERWARNA
& APLIKASI DENGAN JAVA
ERWIN
M. FACHRURROZI

PRAKATA

Puji syukur kehadirat Allah SWT atas limpahan rahmat dan karunianya sehingga Buku "Pemrosesan Citra Berwarna" dapat diselesaikan. Pada buku ini terdapat pengetahuan dalam pemrosesan citra berwarna serta terdapat implementasi program menggunakan Bahasa Java.

Banyak pihak yang telah membantu dalam penyelesaian buku ini. Untuk itu penulis mengucapkan rasa terima kasih kepada Junia Erlina, Rachmad Algani, Bahardiansyah Rua Saputra, Ahmad Fiqih, Auzan Lazuardi, dan Clara Fin Badilah selaku anggota Forum Riset Grup *Digital Image Processing*.

Penulis juga menyampaikan rasa terima kasih kepada Husnita Mala, Heny Maryani, Ari Heka Setiawan, Agum Panji Perdana, M. Bagus Affandi, Gilbert Christopher, Yaumil Fadhilah, Meila Kusuma Perdana, Kurnia Sandi Pratama, Masayu Rahma Yuliza, Juita Asry Lestary, Usman Firnandes, Ojhi Gusti Silaban, dan Jabes Putra P Pasaribu selaku mahasiswa mata kuliah Pengenalan Pola dan Komputer Visi Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Sriwijaya Tahun Ajaran 2016.

Kami menyadari masih terdapat kekurangan dalam buku ini. Untuk itu kritik dan saran terhadap penyempurnaan buku ini sangat diharapkan. Semoga buku ini dapat memberi manfaat bagi semua pihak yang membutuhkan.

Wassalam

Palembang, Desember 2016

Penulis

46

DAFTAR ISI

BAB 1 Model Warna & Pengolahan Citra Pseudocolor

1. Model Warna dan Pengolahan Citra *Pseudocolor*

Perkembangan informasi saat ini sangatlah pesat, informasi yang berkembang tidak hanya berupa teks, bahkan perkembangan ini pun terjadi pada multimedia. Dalam hal ini maksud dari multimedia yaitu citra (image), audio dan video. Dalam kegiatan yang berhubungan dengan internet segala sesuatu yang ada didalamnya dibuat semenarik mungkin. Di era sekarang tidak hanya dalam bentuk teks orang mengirimkan pesan namun dapat juga mengirim pesan dalam bentuk gambar atau video.

Citra merupakan salah satu komponen dari multimedia yang merupakan salah satu bentuk informasi visual. Citra memiliki sebuah karakteristik tersendiri dibandingkan dengan teks, salah satunya yaitu citra mampu memberikan banyak informasi dibandingkan dalam bentuk teks / kata – kata.

Citra dapat diartikan sebagai gambar pada bidang 2D. Citra sendiri merupakan fungsi suatu intensitas cahaya dari dua dimensi. Sumber cahaya yang dapat menerangi suatu objek sehingga objek tersebut dapat memantulkan sebagian dari cahaya yang diterimanya yang nantinya hasil pemantulan tersebut dapat diterima oleh alat – alat optik, seperti alat indra penglihatan manusi, scanner dan lain – lain. Sehingga bayangan dari objek tersebut dapat terekam.

Citra bewarna merupakan salah satu fungsi dari dua variabel f(A,B), dimana A dan B merupakan titik koordinat spasial dan nilai f(A,B) merupakan hasil intensitas citra dari titik koordinat tersebut, teknologi dasar yang dapat menghasilkan sebuah warna dalam citra digital merupakan gabungan antara 3 citra warna yang biasa disebut dengan kata RGB.

Format citra digital ini memiliki karakteristik masing – masing berdasarkan pada tipe dan cara kompresi yang digunakan. Berikut dua format citra digital yang sering dipakai :

1. Bitmap (BMP)

Bitmap merupakan format yang sering digunakan dan salah satu format standar yang digunakan oleh windows. Format dalam penyimpanan file ini dilakukan secara terbalik yang dimulai dari data yang bawah menuju atas. Ukuran file dalam format ini cukup besar sehingga dapat mencapai ukuran Megabytes. File yang menggunakan format ini belum dilakukan kompresi dan memakai sistem RGB (Red, Green and Blue) diman 3 warna tersebut dicampur menjadi satu. Sebuah citra dengan ukuran 8 bit, maka lebar dari citra itu adalah kelipatan dari empat jika hal itu tidak dilakukan maka pada saat penyimpananya nanti akan ditambahkan secara manual menjadi kelipatan empat (Lestari, 2012). Salah satu

software yang dapat membuka file tersebut adalah ACDSee, Paint, Irvan View dan lain sebagainnya.

2. Join Photographic Expert Group (JPEG/JPG)

JPG merupakan salah satu format file yang terkenal hingga sekarang ini. Dikarenakan format ini berukuran kecil dan bersifat portable serta sering digunakan dalam bidang fotografi.

Image processing merupakan proses manipulasi, analisis, memperbaiki, dan mengubah suatu informasi yang terdapat dalam gambar yang dilakukan dengan komputer. Maksud dari informasi gambar adalah gambar dalam bentuk visual dalam dua dimensi. Konsep dasar dari image processing ini dapat diperoleh dari kemampuan mata manusia yang nantinya akan disalurkan ke dalam otak. Image processing banyak digunakan karena tingkat keberhasilan yang didapat sangat tinggi. Image processing ini merupakan gabungan dari beberapa bidang ilmu misalnya optik, elektronik, matematika, fotografi, dan teknologi komputer(Davies, 2012).

Tujuan dari image processing adalah mentransformasikan dan menganalisis suatu gambar agar didapat informasi baru yang jelas dari gambar tersebut. Cara pengaplikasian image processing yaitu Sistem Temu Kembali Citra.

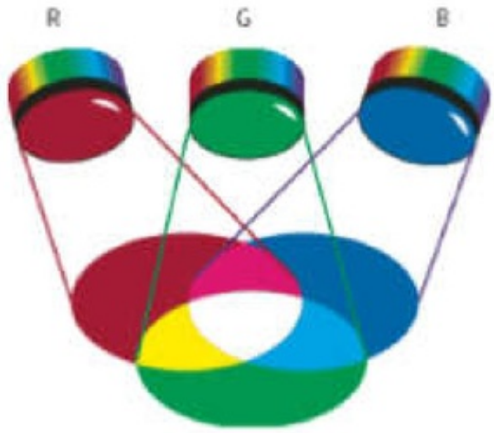
1.1 Model

Representasi Warna (Model – model warna)

Pada dasarnya warna dapat dipisahkan menjadi beberapa komponen baik itu dari segi warna, kecerahan dan lain sebagainya. Pada komponen warna terdapat beberapa model tertentu seperti model RGB, model CMY dan CMYK, dan model HSI.

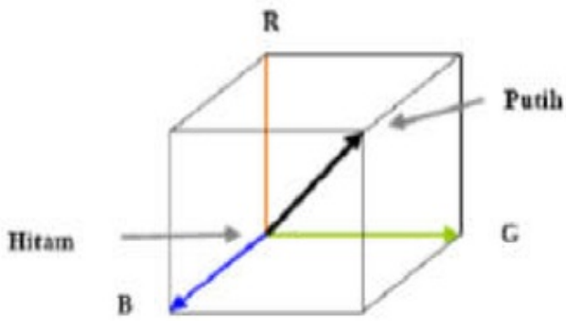
1.1.1 Model Warna RGB

Model warna Red Green Blue (merah, hijau dan biru). Apabila pada ruangan tidak terdapat cahaya (gelap) maka tidak ada gelombang cahaya yang dapat dihasilkan oleh mata sehingga nilai RGB (0,0,0) dan apabila kita tambahkan sebuah cahaya yang berwarna merah maka akan menghasilkan nilai RGB (255,0,0) dan semua benda yang ada dalam ruangan tersebut akan berwarna atau terlihat berwarna merah, begitu pun sebaliknya.



Gambar 1.1 Warna RGB

Menurut Tri Stimulus Vision teori yang mengatakan bahwa manusia dalam melihat warna yaitu dengan cara membandingkan cahaya yang diterima dengan sensor – sensor peka cahaya dalam retina. Sensor – sensor dapat melihat dengan panjang gelombang 630 nm (merah), 530 nm (hijau), dan 450 nm (biru)(Choi et al., 2013). Berikut contoh gambar yang dapat menerangkan sudut – sudut dari sumbu R,G, dan B :



Gambar 1.2 Sudut – sudut warna RGB

Model warna RGB adalah salah satu model additive dimana intensitas dalam suatu warna primer dijumlahkan sehingga akan menghasilkan warna lainnya. Model warna RGB dapat digabungkan dengan halftoning yang dapat menghasilkan warna yang lebih banyak. Dalam sistem on/off, model warna RGB hanya terdapat 8 warna, dimana nilai halftoning 2 x 2 yang menghasilkan 125 (5 warna merah x 5 warna hijau x 5 warna biru).

1.1.2 Model Warna CMY dan Warna CMYK

Warna CMY merupakan hasil dari campuran warna merah, hijau, dan biru. Dua warna dikatakan komplementer (campuran) karena warna yang digabung dengan jumlah persentasi yang sama akan mendapatkan warna putih. Contoh Cyan dicampurkan dengan red akan menghasilkan warna putih, maka dapat dikatakan bahwa cyan merupakan komplemen warna Cyan(Munir, 2004).

Model warna CMY dapat dipakai agar menghasilkan citra berwarna, disebabkan karena adanya ketidakmaksimalan tinta, model CMY tidak bisa mengeluarkan warna hitam yang sempurna. Oleh sebab itu warna CMY diperbaiki lagi dengan model CMYK, dimana K merupakan gabungan dari warna keempat

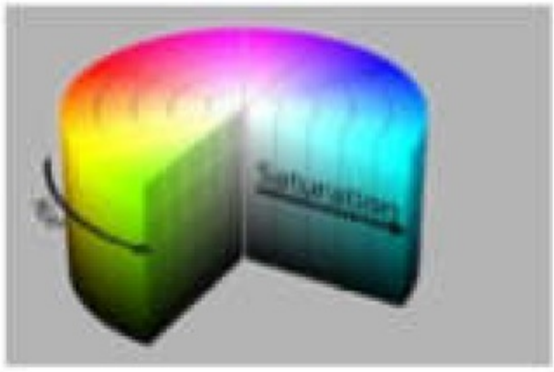
Model CMYK disebut juga sebagai warna subtractive colors. Dari sudut pandang tinta setiap tinta yang dicetakkan tidak menghasilkan warna yang maksimal, maka dari ketiga warna tinta tersebut hanya dapat menghasilkan warna coklat gelap. Sehingga untuk menghasilkan warna hitam dibutuhkan kombinasi dengan warna hitam. Hasil seluruh kombinasi tinta ini yaitu four-color process printing(Chiang, Tai, Yang, Huang, & Huang, 2003).

Gambar 1.3 Warna CMYK

1.1.3 Model Warna HSI

Hue Saturation Intensitas dimana Hue memdefinisikan sebagai warna murni misalkan warna merah, warna violet dan warna orange. Hue dapat dimaanfaatkan sebagai pembeda dan penentu kekuningan, kebiruan, kehijauan dan lain – lain pada suatu sinar. Saturasi merupakan derajat banyaknya warna murni dengan warna putih dan Intensitas merupakan gabungan dari informasi

warna H dan warna S. Intensity ini garis yang menghubungkan titik black dan white(Darmaga et al., 2006).



Gambar 1.4 Warna HSL

1.2 Algoritma

Untuk melalakukan tranformasi dari RGB ke HSL dapat diasumsikan dengan koordinat – koordinat RGB [0,1] adalah warna merah, warna hijau, warna biru dalam ruang warna RGB. Nilai max dihasilkan dari nilai maksimum dari warna merah warna hijau dan warna biru dan nilai min dihasilkan dari nilai minimum warna red, warna green, dan warna blue dapat bertujuan untuk menghasilkan sudut hue (0, 360) sehinga untuk menghasilkan warna HSL menggunakan rumus sebagai berikut :

$$h(hue) = \begin{cases} 0, jika \max = min \\ 60\ x\ \frac{G-B}{max-min} \quad \text{mod } 6. \text{jika max} = R \\ 60\ x\ \frac{B-R}{max-min} + 2. jika\ max = G \end{cases} \qquad (1.1)$$

$$\left\{60\ x \frac{R-G}{max-min} + 4\right\}. jika \max = B \qquad (1.2)$$

Kemudian untuk menghasilkan nilai saturation dan nilai lightnes dapat digunakan rumus sebagai berikut :

L (lightness) = ½ (max + min) (1.3)

$$S(\text{saturation}) \begin{cases} 0. jika \max = min \\ \frac{max-min}{2L}, jika\ 1 \leq 0.5 \\ \frac{max-min}{2-(2L)}, jika\ 1 > 0.5 \end{cases} \qquad (1.4)$$

Dari rumus diatas akan memperoleh hasil nilai lightness dan nilai saturation dalam jangkauan [0,1]. Sehingga untuk memperoleh nilai dengan jangkauan [0,255] dapat dikalikan dengan 255.

Tranformasi RGB ke HSL/HSI/HSV/HCL dan lainnya akan menhasilkan bilangan real. Sementara untuk media bitmap akan menggunakan bilangan integer diskrit oleh sebab itu perlu adanya pembulatan. Untuk jumlah kemungkinan mode RGB adalah 257 x 257 x 257 = 16. 974.593 warna.

1.3 Contoh perhitungan

1.3.1 Transformasi RGB ke HSL

Tranformasikan RGB (65,27,234) kedalam bentuk HSL, maka dapat dikerjakan sebagai berikut :

Setiap nilai RGB (65,27,234) akan diubah dalam jangkauan [0,1] dengan cara setiap nilai dibagi dengan 255: Menjadi (65/255, 27 / 255, 234 / 25) = (0.255,0.106, 0.918).

Max = nilai B (blue) = 0.918

Nilai R (Red) = 0.255

Min = nilai G (green) = 0.106

Max – min = 0.918 – 0.106 = 0.812.

H (hue) = 60 x (R – C / max – min + 4) = 60 x (0.255 – 0.106 / 0.812 + 4) = 251

L (Lightness) = ½ (max + min) = ½ (0.918 + 0.106) = 0.512

S(saturation) = max – min / 2 – (2l) = 0.812 / 2 – (2 x 0.512) = 0.832

Sehinga nilai yang didapat dari dasil transformasi adalah HSL (250 , 0.832, 0.612) dengan jangkauan RGB [0,1] (cahyanti margi, 2010).

1.3.2 Tranformasi HSL ke RGB

Sehingga untuk tranformasi HSL ke RGB dapat menggunakan rumus, dimana nilai hue H € (0^0, 360^0) Saturation S € (0,1) , Lightness L € (0,1)

$$c\begin{cases} 2LxS, jika\ L \leq \frac{1}{2} \\ (2-2L)xS.\ jika\ L > \frac{1}{2} \end{cases}$$

$$H = \frac{H}{60}$$

$$x = c\langle 1|(H\ mod\ 2) - 1| \ \rangle$$

$$(R_1,G_1,B_1)\begin{cases} (0,0,0), jika\ H\ tidak\ terdefinisi \\ (c,x,0), jika\ 0 \leq H < 1 \\ (x,c,0), jika\ 1 \leq H < 2 \\ (0,c,x), jika\ 2 \leq H < 3 \\ (0,x,c), jika\ 3 \leq H < 4 \\ (x,o,c), jika\ 4 \leq H < 5 \\ (c,0,0), jika\ 5 \leq H < 6 \end{cases}$$

M = L – ½ C

(Red, Green, Blue) =(Red1 + m1. Green1+m1. Blue1+m1

Jadi dari perhitungan diatas mendapatkan hasil RGB dalam jangkauan [0, 1] pada jangkauan [0,255] dan dikali 255.

Tranformasikan HSL (2510, 0.832, 0.512) menjadi RGB, berikut cara kerjanya :

L = 0.512 < 0.5 Sehingga

C = (2-2l) x S

= (2-(2 X 0.512)) X 0.832 = 0.812

H' = H /600 = 251 / 60 = 4.183

X = C (1- | H' mod 2 – 1 |)

= 0.812 (1- | (4.183 mod 2) – 1 |)

= 0.812 (1 – 0.187)

= 0.812 (0.183) = 0.149

(Red1, Green1, Blue1) = (X, 0, C), dik. H' = 4. 183 > 4 dan H' < 5 = (0.149, 0, 0.812)

M = L – 0.5 (C)

= 0.512 – 0.5 (0.812) = 0.512 – 0.406 = 0.106

(Red, Green, Blue) = (Red1 + m, G1 + m, B1 + m)

= (0.149 + 0.106, 0 + 0.106, 0.812 + 0.106)

= (0255, 0.106, 0.918)

Sehinga Red Green Blue (0.255, 0.106, 0.918) nilai jangkauan (0,1) nilai jangkauan [0,255] x 255 : (0.255 x 255, 0.106 x 255, 0.918 x 255) menjadi (65.025, 27,03, 234.09) atau (65, 27, 234).

1.4 Java Source Code

```
1. package RGB;
2. import java.awt.Color;
3. public class RGB extends javax.swing.JFrame {
```

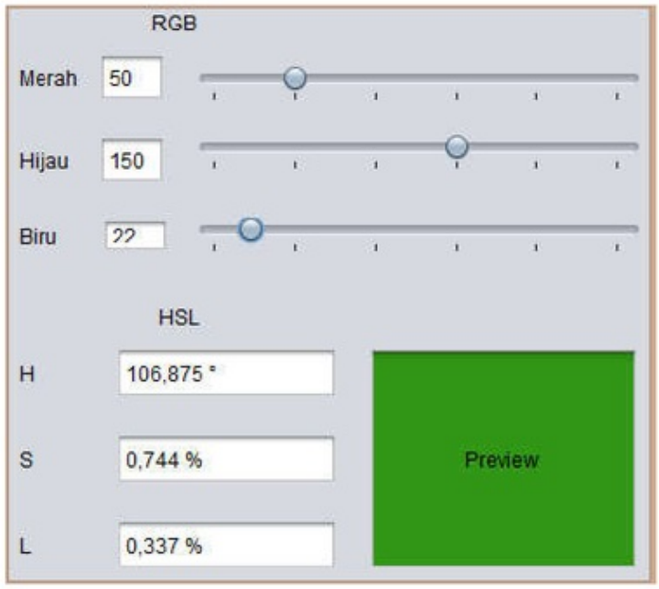
```
4.   public RGB() {
5.  initComponents();
6.  private void editWarna() {
7.  int R,G,B;
8.   R=Merah.getValue();
9.  G=Hijau.getValue();
10.  B=Biru.getValue();
11. jTextField1.setText(""+Merah.getValue());
12.  jTextField2.setText(""+Hijau.getValue());
13. jTextField3.setText(""+Biru.getValue());
14.  warnaRGB.setBackground( new Color(R, G, B) );
15. va.text.DecimalFormatdf=new
16. java.text.DecimalFormat("0.###");
17.  StringstrRet =
18. df.format(R)+","+df.format(G)+","+df.format(B);
19.  convertRGBtoHSL(strRet);
20.  }
21.  private String convertRGBtoHSL( String RGB ){
22. String[] strData;
23. String strRet="";
24. double H = 1;
25. double S = 1;
26. uble L = 0;
27. strData = RGB.split(",");
28.  if(strData.length>=2){
29. double Red;
30. double Green;
31. double Blue;
32. double minimum;
33.   double m imum;
34.   Red   = Double.parseDouble(strData [0])/ 255
35.   Green = Double.parseDouble(strData [1])/ 255;
36.   Blue  = Double.parseDouble(strData [2])/ 255;
37.  maximum = Red;
38. if (maximum < Green ) maximum = Green;
39.   if (maximum < Blue ) maximum = Blue;
40.   minimum = Red;
```

```
41.  if ( minimum > Green ) minimum = Green;
42.  if ( minimum > Blue ) minimum = Blue;
43.   if( maximum==minimum )
44. H=0;
45. else if( maximum==Red )
46. H=60*(((Green-Blue)/(maximum-minimum))%6);
47.  else if( maximum==Green )
48. H=60*(((Blue-Red)/(maximum-minimum))+2);
49. else if( maximum==Blue )
50.  H=70*(((Red-Green)/(maximum-minimum))+4);
51.  L =0.5*(maximum+minimum);
52.  if( maximum==minimum )
53. S=
54.   if( L<=0.5 )
55.   S=(maximum-minimum)/(2*L);
56.  else if(L>0.5)
57.   S=(maximum-minimum)/(2-(2*L));
58.  }
59. java.text.DecimalFormatdf=newjava.text.
60. DecimalFormat("0.###");
61.
62. strRet=df.format(H)+","+df.format(S)+","+df.format(L);
63.  tulisH.setText(df.format(H)+" °");
64. tulisS.setText(df.format(S)+" %");
65.   tulisL.setText(df.format(L)+" %");
66.  return strRet;
67. }
68. }
69. }
```

1.5 Hasil Percobaan

Percobaan 1 dengan nilai RGB (10,17, 137) dan hasil konvert HSL (276,142 °, 0,864 %, 0,288 %) dan hasil warna disamping.

Percobaan 1 dengan nilai RGB (50,150, 22) dan hasil konvert HSL (106,875 °, 0,744 %, 0,337 %) dan hasil warna disamping.



1.6 Kesimpulan

Citra bewarna merupakan salah satu fungsi dari dua variabel f(A,B), dimana A dan B merupakan titik koordinat spasial dan nilai f(A,B) merupakan hasil intensitas citra dari titik koordinat tersebut. Format citra digital ini memiliki karakteristik masing – masing berdasarkan pada tipe dan cara kompres yang digunakan antara lain : Bitmap (BMP) merupakan format yang sering digunakan dan salah satu format standar yang digunakan oleh windows dan Join Photographic Expert Group (JPEG/JPG) merupakan salah satu format file yang terkenal hingga sekarang ini.

Pada dasarnya warna dapat dipisahkan menjadi beberapa komponen baik itu dari segi warna, kecerahan dan lain sebagainya. Pada komponen warna terdapat beberapa model tertentu seperti:

1. model RGB, Model warna RGB adalah salah satu model additive dimana intensitas dalam suatu warna primer dijumlahkan sehingga akan menghasilkan warna lainnya.
2. model CMY dan CMYK, Warna CMY merupakan hasil dari campuran warna merah, hijau, dan biru sedangkan model CMYK, campuran warna merah, hijau, dan biru dimana K merupakan gabungan dari warna keempat.

3. model HSI, Hue Saturation Intensitas dimana Hue memdefinisikan sebagai warna murni, Saturasi merupakan derajad banyaknya warna murni dengan warna putih dan Intensitas merupakan gabungan dari informasi warna H dan warna S.

BAB 2 Dasar Pengolahan Citra Full-Color dan Transformasi Warna

2. Dasar Pengolahan Citra *Full-Color* dan Transformasi Warna

Pengolahan citra warna merupakan pemrosesan citra yang dikhususkan untuk citra yang mempunyai warna (bukan hitam-putih atau *greyscale*). Pengolahan citra berwarna lebih banyak dikembangkan dibanding pengolahan citra hitam-putih atau *greyscale*. Salah satu penyebabnya adalah karena untuk citra hitam putih banyak memiliki kekurangan, salah satunya adalah lebih banyak informasi yang hilang dari citra hitam-putih dibanding dengan citra berwarna. *Color Image Processing* mempunyai berbagai proses dalam penerapannya, diataranya adalah *color transformation, color segmentation, smoothing, sharpening, color image compression,* dan lain sebagainya(Gonzalez, Woods, & Masters, 2007). Yang akan dibahas dalam makalah ini adalah *Color Transformation. Color Transformation* merupakan salah satu proses *Image Processing* yang khususnya digunakan untuk mengubah komponen sebuah citra berwarna pada konteks model warna tunggal(Othman & Sabudin, 2015).

2.1 Model

2.1.1 Formulasi

Pengolahan citra membutuhkan formula atau rumusan perhitungan sebelum dapat diterapkan pada citra yang akan diolah. *Color Trasformation* bisa dinyatakan dalam bentuk formulasi sebagai berikut :

$$q(x, y) = T\,[\,p(x, y)] \qquad (2.1)$$

Pada rumusan diatas, $p(x, y)$ merupakan citra(gambar) masukan, $q(x, y)$ citra keluaran yang sudah di proses, dan T merupakan operator(Gonzalez et al., 2007).

Dasar dari *color transformation* dapat dilihat dalam bentuk

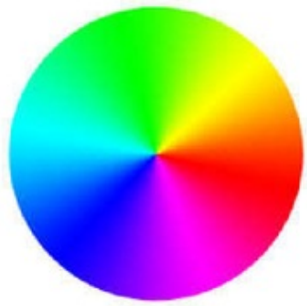
$$s_i = T_i\,(r_1\,, r_2\,, ..., r_n\,) \quad i = 1, 2, ..., n \qquad (2.2)$$

Dimana sederhananya, r_i dan s_i menunjukkan componen warna dari $p(x, y)$ dan $q(x, y)$ pada setiap titik (x, y), n merupakan identifiksi nomor untuk componen warna dan $\{T_1, T_2, ..., T_n\}$ merupakan satu set *transformation* atau *color mapping functions* yang mengoperasikan r_i untuk menghasilkan s_i. Rumusan tersebut akan menciptakan

komputasi yang berbeda, tergantung model warna yang digunakan(Nagai, Uchida, Myodo, & Sakazawa, 2013).

2.1.1.1 Color Complements

Color complements merupakan metode transformasi warna yang menggunakan *opposite color* pada lingkaran warna, yang artinya adalah menggantikan warna pada citra dengan warna pada sisi yang berlawanan yang ada pada lingkaran warna. Contohnya adalah warna *Red* (merah) digantikan dengan warna Cyans, warna hitam digantikan dengan warna putih, dan sebagainya(Gonzalez et al., 2007). Color *complements* biasanya digunakan pada citra *greyscale negatives.* Metode ini berguna untuk memperbaiki dan memperjelas batasan pada citra(Nakajima & Taguchi, 2014).



Gambar 2.1 Lingkaran Warna

Tidak seperti pemrosesan citra yang lain, *Color Complements* tidak tergantung pada HSI (*Hue, Saturation, and Intensity*)(Duangphasuk & Kurutach, 2013). Untuk metode *color complements* itu sendiri tidak merubahah

ketajaman warna (saturation) dari citra, metode ini hanya menampilkan pebedaan visual dari citra sebelum dan sesudah di transformasi(Chen, Hsieh, & Chen, 2010).

2.1.1.2 Color Slicing

Color Slicing atau yang bisa desebut juga pemisahan warna merupakan metode yang digunakan untuk memisahkan suatu objek dalam suatu citra dari warna atau objek lain disekitarnya. Tujuan utama dari metode ini ialah (1) menampilkan (memisahkan) objek atau warna yang menarik dari objek atau warna latarnya, (2) menggunakan daerah yang sudah disisihkan untuk pemrosesan lebih lanjut(Gonzalez et al., 2007).

Salah satu cara yang paling sederhana untuk "slice" (memisahkan) warna adalah memetakan warna yang ingin dipisahkan diluar dari warna netral yang kurang menonjol(Qian, Zhou, & Huang, 2012). Jika warna yang ingin dipisahkan disertai dengan sebuah *cube* (atau *hypercube* untuk $n > 3$) dengan lebar W dan terpusat pada warna-warna tipikal yang mempunyai komponen (a_1, a_2, ..., a_n), maka set transformasi yang biasanya dipakai adalah

$$s_i = \begin{cases} 0{,}5 & \text{if} \left[|r_i - a_i| > \frac{W}{2} \right]_{1 \le j \le n} \\ r_i & otherwise \end{cases} \qquad (2.3)$$

$$i=1,2,\ldots,n$$

Proses metode transformasi ini adalah menandai komponen warna disekitar warna tipikal dengan cara memaksa setiap warna ke titik tengah dari warna netral yang dipilih acak(Prasad, 2015). Jika sebuah bidang lingkar digunakan untuk menentukan warna yang ingin ditransformasi, maka rumusan diatas akan menjadi

$$s_i = \begin{cases} 0{,}5 & \text{if } \sum_{j=1}^{n} (r_j - a_j)^2 > R_0^2 \\ r_i & otherwise \end{cases} \qquad (2.4)$$

$$i=1,2,\ldots,n$$

Dengan R_0 sebagai radius dari bidang lingkar (*sphere* atau *hypersphere* untuk $n > 3$) dan (a_1, a_2,..., a_n) merupakan komponen titik tengahnya(Prasad, 2015).

2.1.1.3 Tone and Color Correction

Perbaikan terhadap gambar dan warna pada citra biasa dilakukan oleh orang di ruangan *digital darkroom*, namun saat ini kebanyakan desktop sudah bisa menggantikannya. Dengan kamera digital, *scanner,* dan *printer,* seseorang bisa

membuat komputer mereka menjadi *digital darkroom*(Gonzalez et al., 2007).

Keefektifan transformasi citra, baik peningkatan kualitas maupun reproduksi warna yang diterapkan harus juga diperhatikan. Karena sudah diiringi dengan pengembangan, perbaikan dan evaluasi yang dilakukan pada citra di monitor, harus juga diimbangi dengan *output* yang akurat. Hal ini akan bisa didapat dengan *device-independent color model*, dengan profil warna yang digunakan untuk memetakan setiap perangkat dan model itu sendiri(Han & Chen, 2016).

Model yang digunakan untuk banyak CMS (*Color Management System*) adalah CIE L*α*β yang juga disebut CIELAB. Komponen warna L*α*β memberikan rumusan

$$L^* = 116 \cdot h\left(\frac{Y}{Y_w}\right) - 16 \qquad (2.5)$$

$$\alpha^* = 500\left[h\left(\frac{X}{X_w}\right) - h\left(\frac{Y}{Y_w}\right)\right] \qquad (2.6)$$

$$\beta^* = 200\left[h\left(\frac{Y}{Y_w}\right) - h\left(\frac{Z}{Z_w}\right)\right] \qquad (2.7)$$

Dimana

$$h(t) = \begin{cases} \sqrt[3]{t} & t > 0.008856 \\ 7.787t + \frac{16}{116} & t \leq 0.008856 \end{cases} \quad (2.8)$$

Nilai X_w , Y_w *dan* Z_w merupakan referensi nilai tristimulus putih (*t*). *Space* Warna L*α*β adalah *colorimetric* (terkode), *perceptual uniform* (perbedaan warna diantara setiap warna dianggap sama) dan *device independent.* L*α*β sendiri mempunyai komponen warna, yaitu L^* untuk kecerahan (*lightness*), α^* untuk merah minus hijau, dan β^* untuk hijau minus biru yang berguna untuk memanipulasi citra maupun memampatkan citra (*compression*)(Prasad, 2015).

Keuntungan dari sistem penggambaran yang terkalibrasi adalah memungkinkan untuk melakukan penyesuaian dan penyeimbangan warna pada citra secara bebas dan interaktif(Othman & Sabudin, 2015).

2.1.1.4 Histogram Processing

Grey-level histogram processing transformation bisa diterapkan ke citra berwarna secara otomatis. Penyamaan histogram secara otomatis akan menentukan sebuah tranformasi yang ditujukan untuk memciptakan sebuah citra dengan histogram nilai intensitas yang seragam. Dikarenakan citra berwarna terdiri dari banyak komponen,

pengimplementasian teknik *grey-scale* harus dipertimbangkan lebih dalam(Gonzalez et al., 2007).

Histogram untuk citra berwarna menyediakan informasi tentang pencahayaan, kontras, *dynamic range*, efek ketajaman warna, dan lain sebagainya. Tidak ada informasi yang jelas mengenai pendistribusian warna dalam histogram ini, maka cara yang dapat dilakukan adalah mengkombinasikan histogram(Unal & Akoglu, n.d.).

Definisi *cumulative density function* (CDF)

$$\mathrm{H}(i) = \sum_{j=0}^{i} \mathrm{h}(j) \quad \text{untuk } 0 \leq i < K \qquad (2.9)$$

Definisi Recursif :

$$\mathrm{H}(i) = \begin{cases} \mathrm{h}(0) & \text{untuk } i = 0 \\ \mathrm{H}(i-1) + \mathrm{h}(i) & \text{untuk } 0 < i < K \end{cases} \qquad (2.10)$$

Monotonically Increasing

$$\mathrm{H}(K-1) = \sum_{j=0}^{K-1} \mathrm{h}(j) = M \cdot N \qquad (2.11)$$

Dengan $\mathrm{H}(K-1)$ merupakan masukan terakhir dari *cumulative histogram* dan $M \cdot N$ merupakan jumlah total *pixel* dalam citra(Khodambashi & Moghaddam, 2009).

Dalam penerapannya, *histogram equalization* menyebar frekuensi dalam sebuah citra, sederhananya untuk

meningkatkan tingkat kegelapan dari gambar. *Histogram equalization* dapat dirumuskan dengan persamaan

$$s_k = T(r_k) \qquad (2.12)$$

Dengan :

r_k : intensitas masukan

s_k : intensitas yang diproses

k : *range* dari intensitas

(Khodambashi & Moghaddam, 2009)

2.2 Algoritma

Berikut merupakan algoritma sederhana *color transformation*, dengan ketentuan :

- *p(x, y)* sebagai masukan
- *q(x, y)* sebagai citra keluaran yang sudah diproses

Algoritma :

1) Tentukan gambar yang ingin diproses → tentukan *p(x, y)*;
2) Hitung tinggi dan lebar dari gambar tersebut;
3) Ambil nilai RGB (nilai warna *Red, Green,* dan *Blue*) dari gambar;

4) Ubah atau transformasi setiap nilai RGB yang diperoleh sesuai dengan warna yang hendak dibuat;
5) Proses selesia dan didapat hasil transformasi (*q(x,y)*).

2.3 Contoh Perhitungan

a. Tentukan nilai *Intensity* (intensitas) dari sebuah gambar yang memiliki nilai RGB sebagai berikut :

R : 143

G : 29

B : 57

Penyelesaian :

$$Intensity\ (\mathrm{I}) = \frac{R+G+B}{3} = \frac{143+29+57}{3} = 76{,}3$$

b. Diketahui suatu gambar memiliki nilai R=200, G=10, dan B=55 , tentukan nilai *Saturation* (ketajaman warna)!

Penyelesaian :

- $Saturation\ (\mathrm{S}) : 1 - \frac{\min(R;G;B)}{\mathrm{I}}$
- $\mathrm{I} = \frac{R+G+B}{3} = \frac{200+10+55}{3} = 88{,}3$

- $S = 1 - \frac{\min(200;10;5\ 5}{8\ 8} = 1 - \frac{10}{8\ 8} = 1 - 0{,}11 =$

  0,89

(Nagai et al., 2013)

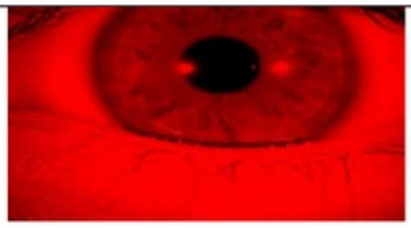
2.4 Java Source Code

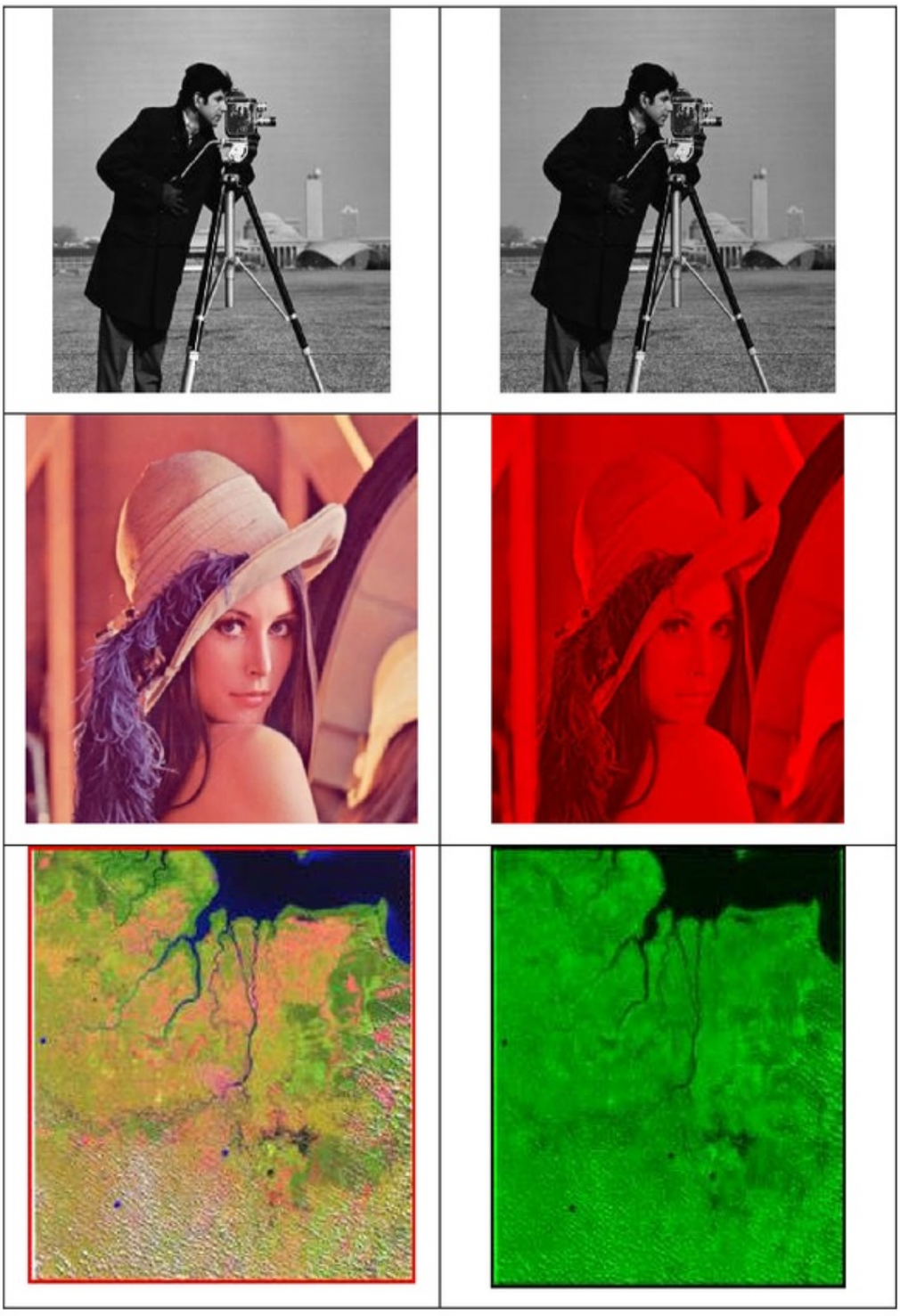
```
1. public class RubahWarna {
2. BufferedImage gambar;
3. File gbr;
4.  int lebar;
5   int tinggi;
6     try {
7     gbr = new File("cameraman.jpg"); //input gambar
8     gambar=ImageIO.read(gbr);  /*gambar dibaca oleh
9     sistam*/
10
11    lebar=gambar.getWidth();   /*mengambil nilai lebar
12    gambar (pixel)*/
13    tinggi=gambar.getHeight(); /*mengambil nilai tinggi
14    mbar (pixel)*/
15    for( int p=0; p<tinggi; p++){
16    for (int q=0; q<lebar; q++){
17        int ambil = gambar.getRGB(q, p);
18
19   /*mengambil nilai RGB gambar*/
20   int alfa = (ambil>>24)&0xff;     /* inisialisasi nilai
21   alpha*/
22   int merah = (ambil>>16)&0xff;    /* inisialisasi nilai
22   red*/
23   int hijau = (ambil>>8)&0xff;     /* inisialisasi nilai
24   hijau*/
```

```
25    int biru = 0&0xff;                /* inisialisasi nilai
26    biru*/
27    ambil=(alfa<<24)|(merah<<16)|(hijau<<8)|0;
28    //transformasi gambar
29          gambar.setRGB(q, p, ambil);
30     }
31         }
32  File hasiloutput = newFile("hasilcameraman.jpg");
33  ImageIO.write(gambar, "jpg",hasiloutput); /* menyimpan
    hasil pemprosean gambar*/
35         } catch (Exception e){
36     }
37  }
38
39
```

2.5 Hasil Percobaan

Hasil Percobaan *color transformation*

Citra Awal	Hasil Transformasi
	



Pada uji *Color transformation* yang diterapkan pada beberapa jenis citra didapatkan beberapa hasil, yaitu :

a. Untuk citra lidah, lenna, mata dan peta, uji penerapan *color transformation* berhasil dilakukan.
b. Untuk citra cameraman, setelah dilakukan transformasi warna tidak ada perubahan yang terjadi, penyebabnya adalah karena citra cameraman merupaka citra *greyscale* dan pada dasarnya citra *greyscale* mempunyai warna dasar hitam dan putih.

2.6 Kesimpulan

a. Citra mempunyai 3 komponen warna, yaitu Merah, Hijau dan Biru (RGB)

b. *Color Transformation* merupakan proses mengubah citra dengan mengubah nilai pixel pada citra.
c. Hasil dari *color transformation* merupakan citra yang sudah diproses dan mempunyai nilai pixel yang berbeda dari citra masukan.
d. Ada beberapa proses dalam *color tranformation*, diantaranya *color correction, color slicing,* dan *histogram processing.*

BAB 3 Penghalusan dan Penajaman Citra Warna

3. Penghalusan dan Penajaman Citra Warna

Penghalusan citra warna *(Color Image Smoothing*) digunakan pada proses untuk menekan gangguan *(noise)* (Sangdong & Ai Zhibin, 1999). Salah satu gangguan pada citra berupa tingginya frekuensi piksel komponen citra yang tak seimbang. Bentuk dari ketidakseimbangan frekuensi yakni adanya perbedaan intensitas piksel yang tidak berkolerasi dengan piksel tetangganya mengakibatkan gangguan yang dapat dilihat oleh mata. Secara visual citra terlihat pecah – pecah, pada citra berwarna terlihat lebih jelas pecah-pecah dari komponen citra itu. Proses penghalusan citra ini akan menghasilkan efek *blurring* yang mereduksi *noise* sehingga kualitas citra menjadi lebih baik (Pan, Tang, & Pan, 2008).

Salah satu metode penghalusan citra yang paling banyak digunakan adalah *average mean* atau *mean filter* (Fu, Xiong, & Sun, 2011). Dalam *average mean*, piksel tetangga dari gambar yang akan dihaluskan diganti dengan nilai rata-rata piksel tetangga sesuai dengan nilai mask filter yang sudah ditentukan (Bharath, Akkala, Rajalakshmi, & Kumar, 2015). Operasi dari average mean menekankan pada operasi penjumlahan dan perkalian dari piksel gambar.

Selain penghalusan citra, proses penajaman *(image sharpening)* juga sangat diperlukan untuk memperjelas tepi

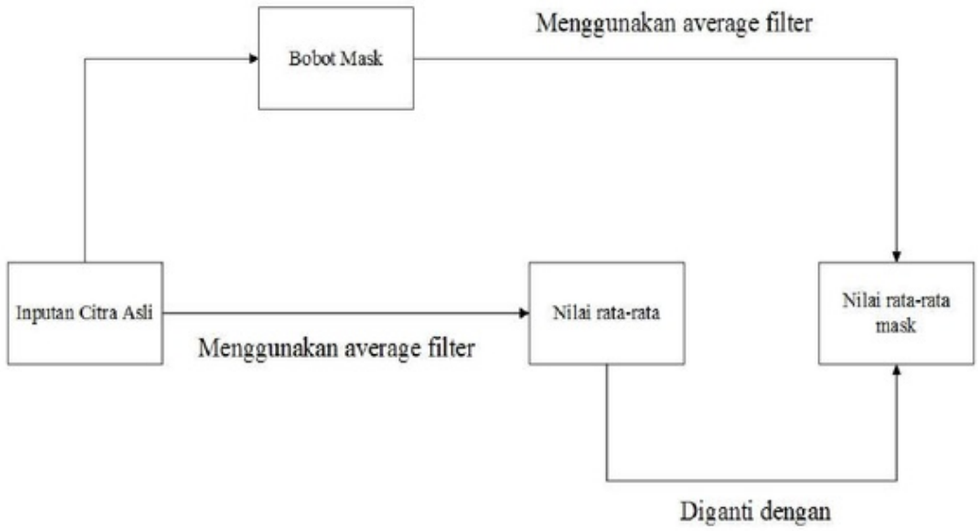
objek dalam suatu citra (Maa, et al., 2014). Penajaman citra warna *(Color Image Sharpening)* merupakan proses perbaikan citra bertujuan untuk memperjelas tepi objek citra berwarna. Pada operasinya proses penajaman ini akan menghilangkan bagian citra yang lembut, berkebalikan dari proses penghalusan citra. Penajaman citra ini akan memperkuat penapis lolos-tinggi (high-pass filter) komponen piksel citra yang berfrekuensi tinggi dan menekan komponen piksel yang berfrekuensi rendah. Dengan proses ini maka secara visual pinggiran objek citra akan terlihat lebih tajam dibandingkan sekitarnya. Oleh karena itu penajaman citra sering dikenal juga dengan penajaman tepi *(edge sharpening)*.

Salah satu metode untuk mempertajam citra adalah Laplacian dengan menggunakan metode laplace untuk mendeteksi tepi. Metode ini merupakan metode transformasi *image* dengan menggunakan penyelesaian persamaan diferensial linier. Dalam prosesnya, metode laplacian ini melakukan penajaman gambar dengan cara mengurangi bagian yang buram/kabur dari citra aslinya sehingga komponen tepi pada warna citra lebih tebal dan kuat (Gonzales & Woods, 2002).

3.1 Model

3.1.1 Penghalusan Citra Berwarna

Penghalusan citra pada domain spasial dapat dilakukan dengan mengganti nilai piksel hasil operasi tertentu piksel tetangga. Metode umum yang digunakan adalah *averaging mask* atau *box filter* atau *mean.* Pada *averaging mask*, rata – rata nilai intensitas piksel diganti dengan nilai rata – rata piksel tetangga. Piksel tetangga diberikan nilai yang sama kepada semua bobot mask yang digunakan (Trivedi & Kurz, 1992). Dimana semakin besar bobot *mask* maka efek *blurring* citra pun semakin meningkat. Hal ini dikarenakan oleh semakin luasnya jangkauan piksel yang akan menggantikan piksel awal dari rata – rata citra. Terdapat dua matrik yang akan dioperasikan yaitu matriks citra dan matriks mask dengan ukuran matriks yang sama. Tahapan umum dalam proses *smoothing* terdapat dalam gambar dibawah ini.



Gambar 3.1. Diagram blok *color image smoothing*.

Dari gambar diatas matrik citra yang akan diproses dinotasikan ke dalam M (baris) x N (kolom) yang memiliki fungsi piksel f(p,q). p untuk baris dan q untuk kolom. Fungsi dari citra hasil berupa $\bar{\boldsymbol{c}}(\boldsymbol{p},\boldsymbol{q})$ yang diformulasikan dalam persamaan di bawah ini :

$$\bar{\boldsymbol{c}}(\boldsymbol{p},\boldsymbol{q}) = \frac{\mathbf{1}}{\boldsymbol{K}_{(\boldsymbol{p},\boldsymbol{q})\in \boldsymbol{Spq}}} \sum \boldsymbol{c}(\boldsymbol{p},\boldsymbol{q}) \qquad (3.1)$$

Dimana :

$\bar{c}(p,q)$: nilai rata-rata dari piksel pada koordinat titik p,q

$K_{(\mathrm{p,q})\in \mathrm{pq}}$: jumlah piksel tetangga yang terlibat dalam perhitungan rata-rata

$c(p,q)$: nilai piksel pada titik koordinat p,q

Sedangkan untuk citra RGB, fungsi citra hasil *smoothing* dinyatakan dalam persamaan di bawah ini :

$$\bar{c}(p,q) = \begin{bmatrix} \frac{1}{K_{(p,q)\in Spq}} \sum \boldsymbol{R(p,q)} \\ \frac{1}{K_{(p,q)\in Spq}} \sum \boldsymbol{G(p,q)} \\ \frac{1}{K_{(p,q)\in Spq}} \sum \boldsymbol{B(p,q)} \end{bmatrix} \quad (3.2)$$

Dimana :

$\bar{c}(p,q)$: nilai rata-rata dari piksel pada koordinat titik p,q

$K_{(p,q)\in Spq}$: jumlah piksel tetangga yang terlibat dalam perhitungan rata-rata

$R(p,q)$: nilai piksel komponen warna merah pada koordinat titik p,q

$G(p,q)$: nilai piksel komponen warna hijau pada koordinat titik p,q

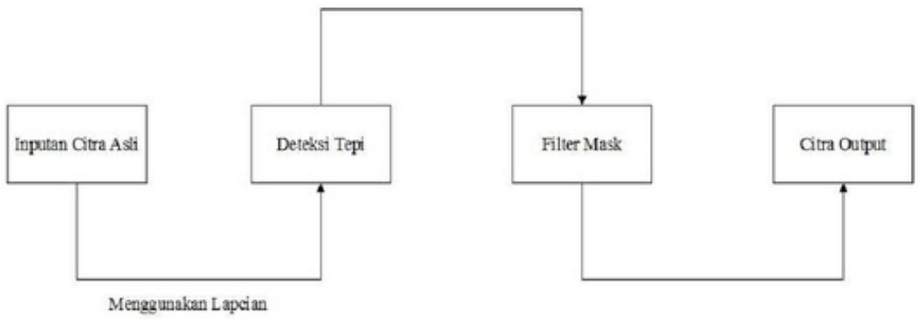
$B(p,q)$: nilai piksel komponen warna biru pada koordinat titik p,q

3.1.2 Penajaman Citra Berwarna

Penajaman atau memperjelas sisi pada citra umumnya dilakukan dengan menggunakan laplician. Prinsip dalam penajaman yakni dengan menjumlahkan citra yang asli dengan citra hasil dari operasi deteksi tepi. Dalam

mendeteksi tepi digunakan operator *laplace* atau dikenal dengan transformasi lapcian. Operator *laplace* digunakan untuk menyelesaikan permasalahan linear divergen suatu fungsi.

Tahapan dalam proses *sharpening* terdapat dalam gambar di bawah ini.



Gambar 3.2. Diagram blok *color image sharpening.*

Adapun kriteria dalam melakukan penajaman citra yaitu :

1. Nilai koefisian bernilai positif, negative, atau nol;
2. Jumlah seluruh koefisien = 0 atau 1, dimana jika koefisien = 0 berarti komponen berfrekuensi rendah akan turun nilainya, sedangkan untuk koefisien = 1 komponen berfrekuensi rendah nilainya akan tetap seperti semula (Gonzales & Woods, 2002).

Untuk nilai koefisian di titik pusat penapis memiliki peranan penting dalam proses konvolusi pada citra. Selanjutnya akan dilakukan perkalian antara intensitas citra

berfrekuensi tinggi dengan nilai tengah piksel citra yang akan dihitung. Sehingga piksel citra yang bernilai besar akan semakin diperbesar nilainya, namun jika piksel citra rendah tidak akan berubah nilainya dengan kata lain bernilai konstan. Dalam proses penajaman diperlukan *mask* yang dikenal dengan istilah *filter mask.* Semakin tinggi nilai poin dari filter mask maka ketajaman citra hasil akan semakin tinggi pula.

Operasi Laplacian dari citra *f(p,q),* di notasikan $\boldsymbol{\nabla^2 f(p,q)}$**,** didefinisikan pada persamaan di bawah ini.

$$\nabla^2 f(p,q) = \frac{\partial^2 f(p,q)}{\partial p^2} + \frac{\partial^2 f(p,q)}{\partial p^2} \qquad (3.3)$$

Secara umum, perhitungan dari turunan kedua tersebut adalah

$$\frac{\partial^2 f(p,q)}{\partial p^2} = f(p+1,q) + f(p-1,q) - 2f(p,q) \qquad (3.4)$$

dan

$$\frac{\partial^2 f(p,q)}{\partial p^2} = f(p,q+1) + f(p,q-1) - 2f(p,q) \qquad (3.5)$$

Jadi, persamaan untuk penajaman tepi adalah

$$\nabla^2 f = [f(p+1,q) + f(p-1,q) + f(p,q+1) + f(p,q-1)] - 4f(p,q) \qquad (3.6)$$

Hasil Penajaman menggunkaan Laplacian berdasarkan persamaan diatas diperoleh persamaan :

$$g(p,q) = f(p,q) + c[\nabla^2 f(p,q)] \qquad (3.7)$$

Dimana :

$g(p,q)$ = fungsi hasil penajaman image

$f(p,q)$ = fungsi input image

c = constant (bernilai (-1) jika menggunakan Laplacian filter (a) dan (b), sebaliknya bernilai 1)

$\nabla^2 f(p,q)$ = Laplician Filter

Berdasarkan pada persamaan di atas maka diketahui bahwa koefisien yang konstan sangat memengaruhi dalam menajamkan tepi objek citra. Jika konstan maka citra frekuensi citra tidak akan ditambah untuk dipertajam. Hal ini menunjukkan koefisien penajaman citra dapat disesuaikan tergantung kebutuhan. Dengan demikian, bagian tepi objek citra akan terlihat berbeda dibandingkan dengan latar belakang citra itu sendiri.

3.2 Algoritma

3.3.1 Algoritma Penghalusan Citra Berwarna

Berikut ini merupakan algoritma penghalusan citra berwana dengan menggunakan 5x5 *average mask*, dengan ketentuan sebagai berikut :

e. $f(p,q)$ = citra masukan

f. $R(p,q)$ = nilai pikel komponen warna merah pada titik p,q

g. $G(p,q)$ = nilai pikel komponen warna hijau pada titik p,q

h. $B(p,q)$ = nilai pikel komponen warna hijau pada titik p,q

i. $g(p,q)$ = citra hasil

Algoritma :

1) Tentukan nilai warna *R(p,q), G(p,q), B(p,q)* di masing-masing piksel citra.
2) Periksa apakah piksel yang akan dihitung nilainya berada di pinggir citra (tidak memiliki tetangga lengkap).
3) Jika iya, maka nilai piksel tetangga menjadi 0.
4) Hitung nilai baru untuk setiap *R(p,q), G(p,q), B(p,q),* dimana nilai tersebut merupakan rata-rata dari nilai piksel itu sendiri dan 4 tetangganya.
5) Ubah semua piksel sehingga menghasilkan $g(p,q)$.

3.3.2 Algoritma Penajaman Citra Berwarna

Berikut ini merupakan algoritma penajaman citra berwana dengan menggunakan metode Laplician, dengan ketentuan sebagai berikut :

j. $f(p,q)$ = citra masukan

k. $Spatial\ mask = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

l. $W(p,q)$ = nilai warna (merah, hijau, dan biru) yang ada di lokasi piksel (p,q)

m. $\nabla^2 f(p,q)$ = nilai warna baru (merah, hijau, dan biru) yang ada di lokasi piksel (p,q)

n. c = nilai 1 jika koofisien tengah dari mask positif dan -1 jika sebaliknya

o. M = matriks berordo 3 x 3 yang mewakili nila warna $W(p,q)$ dan 8 tetangga sekelilingnya.

p. $g(p,q)$ = citra keluaran

Algoritma :

1) Tentukan nilai warna $W\ (p,q)$ dimasing-masing piksel citra.
2) Bentuk sebuah matriks M dimasing-masing piksel (p,q). Apabila $W(p,q)$ tidak memiliki tetangga maka nilai tetangga sebelah merupakan nilai $W(p,q)$.
3) Lakukan perhitungan $\nabla^2 f(p,q)$ dengan cara melakukan perkalian secara posisi yang sama antara matriks M dan Spatial mask. Jika nilai baru dari $\nabla^2 f(p,q)$ negatif maka $\nabla^2 f(p,q) = 0$ dan jika $\nabla^2 f(x,y)$lebih besar dari 255 maka $\nabla^2 f(p,q) = 255$.

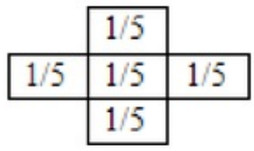
4) Kemudian hasil $\nabla^2 f(p,q)$tersebut dikalikan dengan nilai koefisien c.
5) Lakukan pengurangan *f(p,q)* terhadap $\nabla^2 f(p,q)$ dan terapkan nilai warna tersebut kedalam citra keluaran *g(p,q)*.

3.3 Contoh Perhitungan

a. Diberikan sebuah matriks citra berukuran 5x5 piksel, yang ditunjukkan oleh gambar di bawah. Tentukan nilai piksel pada titik (2,2) setelah dilakukan penghalusan dengan menggunakan 5x5 *spatial average mask*.

R : 226 G : 137 B : 125	R : 226 G : 137 B : 125	R : 226 G : 137 B : 125	R : 226 G : 137 B : 125	R : 226 G : 137 B : 125
R : 226 G : 137 B : 125	R : 226 G : 137 B : 125	R : 226 G : 137 B : 125	R : 226 G : 137 B : 125	R : 226 G : 137 B : 125
R : 227 G : 133 B : 125	R : 227 G : 133 B : 125	**R : 227** **G : 133** **B : 125**	R : 227 G : 133 B : 125	R : 227 G : 133 B : 125
R : 223 G : 136 B : 128	R : 223 G : 136 B : 128	R : 223 G : 136 B : 128	R : 223 G : 136 B : 128	R : 223 G : 136 B : 128
R : 226 G : 138 B : 120	R : 226 G : 138 B : 120	R : 226 G : 138 B : 120	R : 226 G : 138 B : 120	R : 226 G : 136 B : 120

Gambar 3.3 Matriks citra RGB input.



Gambar 3.4 Matriks 5x5 *spatial average mask*

Untuk menghitung nilai $\bar{c}(2,2)$, maka perhitungan dilakukan sebagai berikut :

$$\bar{c}(2,2) = \begin{bmatrix} \frac{1}{5}\, x\big(R(2,1) + R(1,2) + R(2,2) + R(2,3) + R(3,2)\big) \\ \frac{1}{5}\, x\big(G(2,1) + G(1,2) + G(2,2) + G(2,3) + G(3,2)\big) \\ \frac{1}{5}\, x\big(B(2,1) + B(1,2) + B(2,2) + B(2,3) + B(3,2)\big) \end{bmatrix}$$

$$\bar{c}(2,2) = \begin{bmatrix} \frac{1}{5}\, x(227 + 226 + 227 + 227 + 223) \\ \frac{1}{5}\, x(133 + 137 + 133 + 133 + 136) \\ \frac{1}{5}\, x(125 + 125 + 125 + 125 + 125) \end{bmatrix}$$

$$\bar{c}(2,2) = \begin{bmatrix} 226 \\ 134.4 \\ 125.6 \end{bmatrix}$$

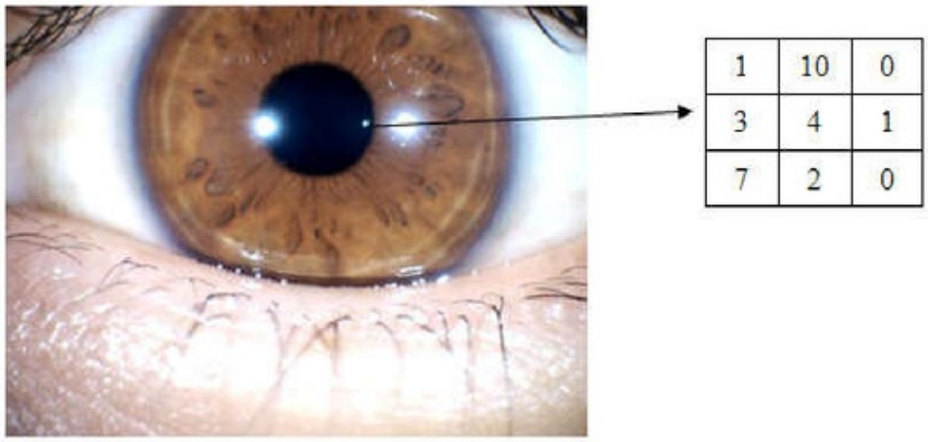
$$\bar{c}(2,2) \approx \begin{bmatrix} 226 \\ 134 \\ 126 \end{bmatrix}$$

Maka, hasil citra setelah dioperasikan 5x5 *spatial average mask* pada titik (2,2) menjadi:

R : 226 G : 137 B : 125	R : 226 G : 137 B : 125	R : 226 G : 137 B : 125	R : 226 G : 137 B : 125	R : 226 G : 137 B : 125
R : 226 G : 137 B : 125	R : 226 G : 137 B : 125	R : 226 G : 137 B : 125	R : 226 G : 137 B : 125	R : 226 G : 137 B : 125
R : 227 G : 133 B : 125	R : 227 G : 133 B : 125	**R : 226** **G : 134** **B : 126**	R : 227 G : 133 B : 125	R : 227 G : 133 B : 125
R : 223 G : 136 B : 128	R : 223 G : 136 B : 128	R : 223 G : 136 B : 128	R : 223 G : 136 B : 128	R : 223 G : 136 B : 128
R : 226 G : 138 B : 120	R : 226 G : 138 B : 120	R : 226 G : 138 B : 120	R : 226 G : 138 B : 120	R : 226 G : 136 B : 120

Gambar 3.5.Matriks Hasil Perhitungan Average Mask

b. Diberikan sebuah matriks citra berukuran 3x3 piksel, yang ditunjukkan oleh gambar di bawah. Tentukan nilai masing-masing piksel setelah dilakukan proses penajaman dengan menggunakan 3x3 *spatial average mask*.



Gambar 3.6 Matriks citra mata ukuran 3x3

$$\text{Lapmask} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Gambar 3.7 Matriks citra mata ukuran 3x3

Convutional Matriks:

$$\begin{bmatrix} 1 & 10 & 0 \\ 3 & 4 & 1 \\ 7 & 2 & 0 \end{bmatrix} x \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1*0 & 10*1 & 0*0 \\ 3*1 & 4*-4 & 1*1 \\ 7*0 & 2*2 & 0*0 \end{bmatrix}$$

$$\begin{bmatrix} 1*0 & 1*1 & 10*0 & \\ 1*1 & 1*-4 & 10*1 & 0*0 \\ 3*0 & 3*1 & 4*0 & 1*1 \\ & 7 & 2 & 0 \end{bmatrix} = \begin{bmatrix} 11 & x & x \\ x & x & x \\ x & x & x \end{bmatrix}$$

$$\begin{bmatrix} 1*0 & 10*1 & 0*0 \\ 1*1 & 10*-4 & 0*1 \\ 3*0 & 4*1 & 1*0 \\ 7 & 2 & 0 \end{bmatrix} = \begin{bmatrix} 11 & -25 & x \\ x & x & x \\ x & x & x \end{bmatrix}$$

$$\begin{bmatrix} 1 & 10*0 & 0*1 & 0*0 \\ 3 & 10*1 & 0*-4 & 0*1 \\ & 4*0 & 1*1 & 1*0 \\ 7 & 2 & 0 & \end{bmatrix} = \begin{bmatrix} 11 & -25 & 11 \\ x & x & x \\ x & x & x \end{bmatrix}$$

$$\begin{bmatrix} 1*0 & 1*1 & 10*0 & 0 \\ 3*1 & 3*-4 & 4*1 & 1 \\ 7*0 & 7*1 & 2*0 & 0 \end{bmatrix} = \begin{bmatrix} 11 & -25 & 11 \\ 3 & x & x \\ x & x & x \end{bmatrix}$$

$$\begin{bmatrix} 1*0 & 10*1 & 0*0 \\ 3*1 & 4*-4 & 1*1 \\ 7*0 & 2*1 & 0*0 \end{bmatrix} = \begin{bmatrix} 11 & -25 & 11 \\ 3 & 0 & x \\ x & x & x \end{bmatrix}$$

$$\begin{bmatrix} 1 & 10 & 0 & \\ 3 & 4*0 & 1*1 & 1*0 \\ 7 & 2*1 & 0*-4 & 0*1 \\ & 2*0 & 0*1 & 0*0 \end{bmatrix} = \begin{bmatrix} 11 & -25 & 11 \\ 3 & 0 & 1 \\ x & x & x \end{bmatrix}$$

$$\begin{bmatrix} & 1 & 10 & 0 \\ 3*0 & 3*1 & 4*0 & 1 \\ 7*1 & 7*-4 & 2*1 & 0 \\ 7*0 & 7*1 & 2*0 & \end{bmatrix} = \begin{bmatrix} 11 & -25 & 11 \\ 3 & 0 & 1 \\ -9 & x & x \end{bmatrix}$$

$$\begin{bmatrix} 1 & 10 & 0 \\ 3*0 & 4*1 & 1*0 \\ 7*1 & 2*-4 & 0*1 \\ 7*0 & 2*1 & 0*0 \end{bmatrix} = \begin{bmatrix} 11 & -25 & 11 \\ 3 & 0 & 1 \\ -9 & 5 & x \end{bmatrix}$$

$$\begin{bmatrix} 1 & 10 & 0 & \\ 3 & 4*0 & 1*1 & 1*0 \\ 7 & 2*1 & 0*-4 & 0*1 \\ & 2*0 & 0*1 & 0*0 \end{bmatrix} = \begin{bmatrix} 11 & -25 & 11 \\ 3 & 0 & 1 \\ -9 & 5 & -3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 10 & 0 \\ 3 & 4 & 1 \\ 7 & 2 & 0 \end{bmatrix} x \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 11 & -25 & 11 \\ 3 & 0 & 1 \\ -9 & 5 & -3 \end{bmatrix}$$

Maka, hasil citra setelah dioperasikan 3x3 *sharpening* menggunakan *Laplican Filtering* menjadi:

$$g(x,y) = f(x,y) + c[\nabla^2 f(x,y)]$$

$$= \begin{bmatrix} 1 & 10 & 0 \\ 3 & 4 & 1 \\ 7 & 2 & 0 \end{bmatrix} + (-1) \begin{bmatrix} 11 & -25 & 11 \\ 3 & 0 & 1 \\ -9 & 5 & -3 \end{bmatrix}$$

$$= \begin{bmatrix} -10 & 35 & -11 \\ 0 & 4 & 0 \\ 16 & -3 & -3 \end{bmatrix}$$

3.4 Java Source Code

3.4.1 Kode Penghalusan Citra Berwarna

```
1.  public class Penghalusan {
2.  BufferedImage citra; /*merupakan variabel citra yang
    menggunakan buffer java*/
3.  int lebarGbr, tinggiGbr; /*variabel menampung nilai
    lebar dan panjang citra*/
4.  String warna; /*variabel menampung jenis warna yang
    diminta*/
5.  int wMerah,wHijau,wBiru;  /*variabel menampung nilai
    merah, hijau, dan biru
    pada citra*/
6.  public Penghalusan() {
7.  try {
8.  File masukan = new File("peta.jpg"); //menampung lokasi
    masukan file citra
```

```
9.  citra = ImageIO.read(masukan); /*membaca file citra
    masukan*/
10. lebarGbr = citra.getWidth(); /*menampung lebar citra*/
11. tinggiGbr = citra.getHeight(); /*menampung panjang
    ra*/
12. for (int p = 0; p < lebarGbr; p++) { //perulangan
    putasi nilai citra baru
13. for (int q = 0; q < tinggiGbr; q++) {
14. Color warnaPiksel = new Color(citra.getRGB(p,q));
    //mengambil citra p,q
15. wMerah = hitungRataNilai(p,q,"MERAH",warnaPiksel);
    //hitung rerata merah
16. wHijau = hitungRataNilai(p,q,"HIJAU",warnaPiksel);
    //hitung rerata hijau
17. wBiru = hitungRataNilai(p,q,"BIRU",warnaPiksel);
    //hitung rerata biru
18. citra.setRGB(p,q,new
    Color(wMerah,wHijau,wBiru).getRGB());                }
19. }
20. File output = new File("petaSmoothing.png");  //membuat
    file baru
a.  ImageIO.write(citra,"png",output); /*menerapkan hasil
    nilai citra baru ke dalam file*/

b.  } catch (IOException entok) {        }
21. }
22. private int hitungRataNilai(int p, int q, String
    namaWarna,Color warnaPiksel) {
23. int nilaiRerata= 0;//inisial nilai piksel rerata
    nilaiRerata= (ambilNilaiPiksel(p-
    1,q,namaWarna,warnaPiksel)+ //hitung nilai
24. ambilNilaiPiksel(p,q,namaWarna,warnaPiksel)+
25. ambilNilaiPiksel(p+1,q,namaWarna,warnaPiksel)+
26. ambilNilaiPiksel(p,q-1,namaWarna,warnaPiksel)+
27. ambilNilaiPiksel(p,q+1,namaWarna,warnaPiksel))/5;
28. return nilaiRerata; /*mengembalikan nilai retata warna*/
29. }
```

```
30. public int ambilNilaiPiksel(int p, int q, String
    namaWarna,Color warnaPiksel){
31. int nilaiPiksel;
32. if(q>citra.getHeight() || p<0 || p>citra.getWidth() ||
    q<0){
33. return 0; //pengkondisian nilai citra jika lewat dari
    posisi p,q
34. }else{
35. switch (namaWarna) {
36. case "MERAH":   //pengkondisian jika parameter namaWarna
    merah
37. nilaiPiksel = warnaPiksel.getRed();
38. break;
39. case "HIJAU": //pengkondisian jika parameter namaWarna
    Hijau
40. nilaiPiksel = warnaPiksel.getGreen();
41. break;
42. case "BIRU": //pengkondisian jika parameter namaWarna
    Biru
43. nilaiPiksel = warnaPiksel.getBlue();
44. break;
45. }
46. }
47. return nilaiPiksel; //mengembalikan nilai piksel yang
    telah di peroleh
48. }
49. public static void main(String[] args) {
        50. Penghalusan smuting = new Penghalusan();
            //menjalankan objek smuting
51. }
52. }
```

3.4.2 Kode Penajaman Citra Berwarna

```
1.  public class Penajaman {
2.  BufferedImage citra; //merupakan variabel buffer gambar
    yang disediakan java
3.  Color[][] warnaPiksel = new Color[3][3]; //matriks
    berisi warna piksel citra
4.  Color warnaPq,wBaru; //variabel penampung warna pada p,q
    dan warna baru citra
5.  int[][] matriksSpatialMask = {{0, 1, 0}, {1, -4, 1}, {0,
    1, 0}};
6.  int wMerah,wHijau,wbiru,lebarGbr, panjangGbr;
7.  public Penajaman() { //class penajaman
8.  try {
9.  File masukan = new File("peta.jpg"); //memasukan citra
    kedalam file
10. citra = ImageIO.read(masukan); //membaca file masukan
11. lebarGbr = citra.getWidth(); //mendeteksi lebar gambar
12. panjangGbr = citra.getHeight(); //mendeteksi panjang
    gmbar
13. for (int p = 1; p < lebarGbr-1; p++){
14. for (int q = 1; q < panjangGbr-1; q++) {
15. warnaPq = new Color(citra.getRGB(p,q)); /*mengambil
    warna piksel p,q*/
16. warnaPiksel = ambilMatriks3x3(p, q, citra); //membuat
    matriks 3x3
17. wMerah =Math.min(255,Math.max(0,warnaPq.getRed()-
    hitung(warnaPiksel,matriksSpatialMask,"MERAH")));
    //hitung merah
18. wHijau =Math.min(255,Math.max(0,warnaPq.getGreen()-
19. hitung(warnaPiksel,matriksSpatialMask,"HIJAU")));
    //hitung hijau
20. wbiru = Math.min(255,Math.max(0,warnaPq.getBlue()-
    hitung(warnaPiksel,matriksSpatialMask,"BIRU")));
    //hitung biru
21. wBaru = new Color(wMerah,wHijau,wbiru); //menset ketiga
    warna baru
```

```
22. citra.setRGB(p,q,wBaru.getRGB()); //memasukan warna
    citra baru
23. }
24. }
25. File Output = new
26. ImageIO.write(citra,"jpg",Output); //memasukan citra
    kedalam file baru
27. } catch (IOException entok) {}
28. }
29. private int hitung(Color[][] c, int[][]
    matriksMask,String namaWarna) {
30. int hasil=0; // inisial warna hasil citra
31. int nWarna = 3; //jumlah warna yang dilakukan dalam hal
    ini 3 warna
32. for (int p = 0; p < 3; p++) { //looping sebanyak nilai
    warna
33. for (int q = 0; q < 3; q++) {
34. switch (nWarna){
35. case 1: namaWarna = "MERAH"; //case warna yang di input
    parameter merah
36. hasil=hasil+(c[p][q].getRed()*matriksMask[p][q]);
    //lakukan komputasi
37. break;       // perkalian matriks
38. case 2: namaWarna = "HIJAU"; //case warna yang di input
    parameter hijau
39. hasil=hasil+(c[p][q].getGreen()*matriksMask[p][q]);
    //lakukan komputasi
40. break;       // perkalian matriks
41. case 3: namaWarna = "BIRU"; //case warna yang di input
    parameter biru
42. hasil=hasil+(c[p][q].getBlue()*matriksMask[p][q]);
    //lakukan komputasi
43. break;       // perkalian matriks
44. }
45. }
46. }
47. return
```

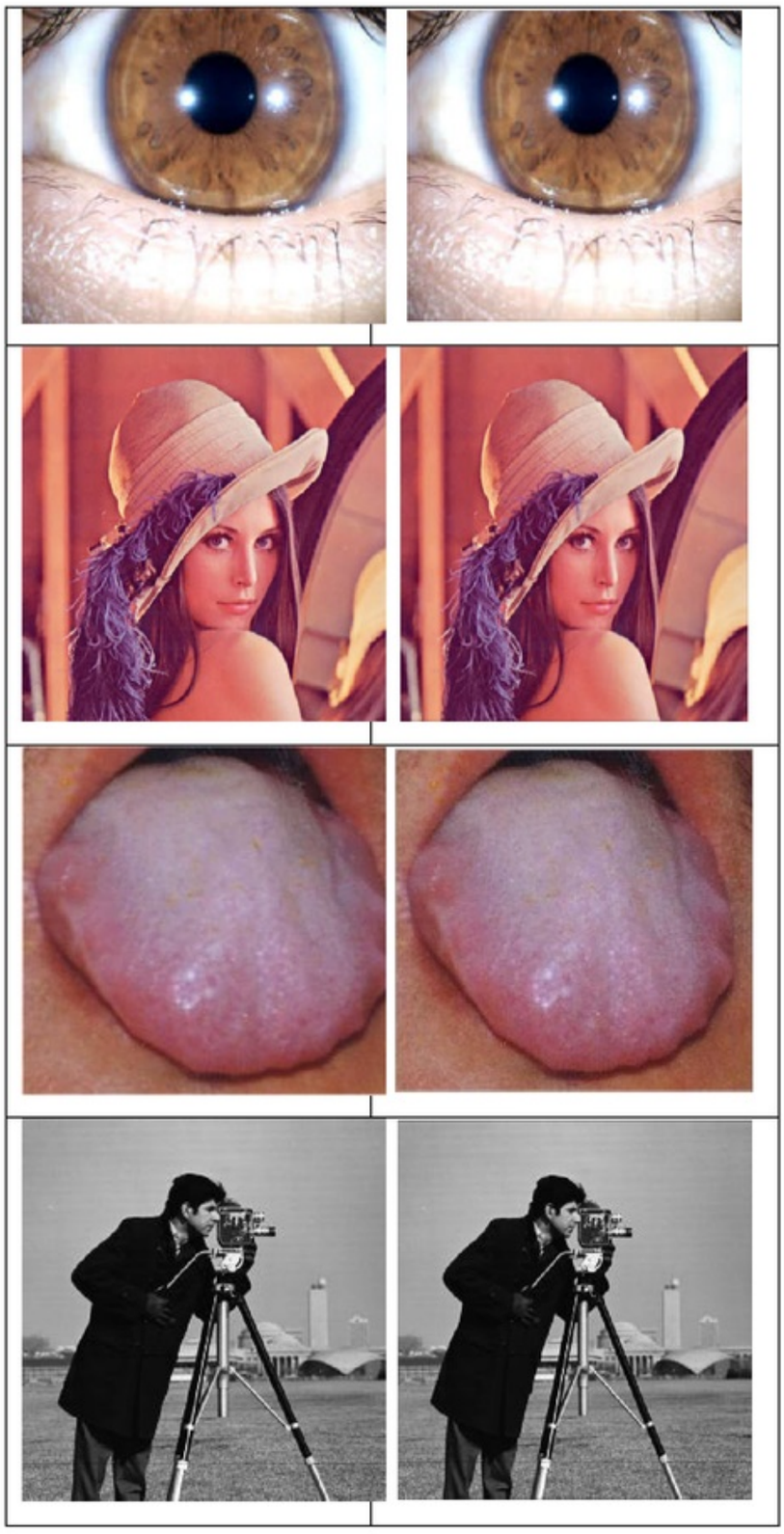
```
48. }
49. private Color[][] ambilMatriks3x3(int p, int q,
    BufferedImage citra) { /*fungsi membuat matriks 3x3*/
50. Color warnaPosisi[][] = new Color[3][3];
51. for (int s = -1; s <=1 ; s++) {
52. for (int d = -1; d <=1; d++) {
53. warnaPosisi[s+1][d+1]=new Color(citra.getRGB(p+s,q+s));
    //posisi warna
54. }
55. }
    return warnaPosisi;
57. }
58. public static void main(String[] args) {
59. Penajaman tajam = new Penajaman(); // menjalankan class
    penajaman
60. }
61. }
```

3.5 Hasil Percobaan

3.5.1 Hasil Penghalusan Citra *(Image Smoothing)*

Hasil Percobaan Color Image *Smoothing*

Citra Awal	Citra Hasil *Smoothing*
	

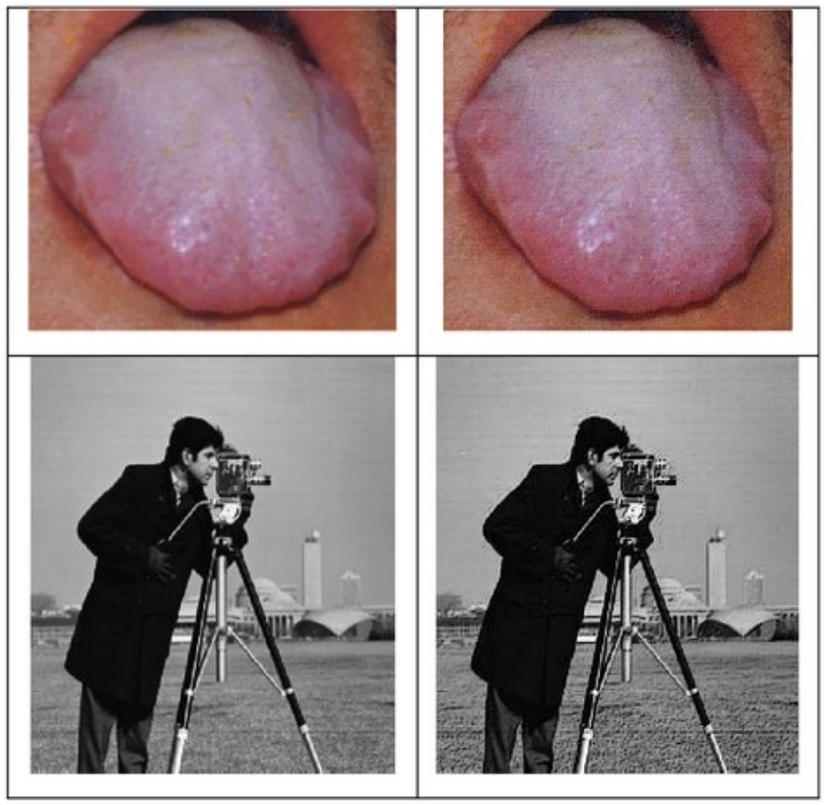


Dalam proses penghalusan dari kelima citra tidak terlalu nampak jika dilihat dengan kasat mata. Hal ini terjadi dikarenakan oleh gangguan *(noise)* yang dimiliki oleh citra cukup banyak sedangkan matriks *mask* yang digunakan berukuran hanya berukuran 5x5. Dalam prosesnya nilai rata – rata piksel citra asli yang dikali dengan 1/5 dari matriks mask, menjadikan jangkauan piksel citra hasil penghalusan yang menggantikan piksel citra asli lebih kecil cakupannya. Kecilnya cakupan piksel yang diganti tersebut mengakibatkan citra hasil penghalusan tidak begitu nampak secara kasat mata seperti yang ada pada gambar peta, mata, lenna, lidah, dan cameraman. Tingginya tingkat cahaya dari gambar mata dan lidah juga menjadi penyebab susahnya untuk mengetahui bahwa citra telah diproses dalam penghalusan citra.

3.5.2 Hasil Penajaman Citra *(Image Sharpening)*

Hasil Percobaan Color Image *Sharpening*

Citra Awal	Citra Hasil *Sharpening*
	
	
	



Proses penajaman objek dari kelima citra menunjukkan hasil terdapat tiga citra yang dapat dilihat dengan kasat mata bahwa sisi objek citra nya menjadi lebih tajam atau garisnya tampak lebih tebal dibandingkan dengan sisi objek lainnya. Adapun ketiga citra yang nampak lebih tajam sisi objeknya ialah gambar peta, lenna, dan cameraman pada tabel diatas. Adanya penajaman dapat dilihat melalui garis biru yang terdapat pada citra. Gambar peta terlihat lebih banyak garis – garis seperti gangguan yang muncul setelah dilakukannya proses penajaman citra. Garis – garis yang muncul paling

terlihat jelas dengan adanya warna hitam ditunjukkan oleh gambar lenna. Hal ini terjadi disebabkan karena gambar lenna komponen warna citra nya didominasi warna terang dan lembut. Sedangkan pada gambar cameraman sisi yang lebih menajam dibuktikan dengan garis yang muncul dibagian dekat sisi tepi dari objek.

Namun pada gambar mata dan lidah tidak begitu nampak hasil proses penajaman bagian objek citra dikarenakan oleh cahaya yang dimiliki oleh kedua citra terlalu terang. Terangnya pencahayaan yang dimiliki citra mengakibatkan penajaman objek tidak berwarna kehitam – hitaman atau bergaris – garis seperti yang nampak pada gambar peta, lenna, dan cameraman sehingga mata sulit mengetahui adanya penajaman dari gambar mata dan lidah.

3.6 Kesimpulan

a. Proses penghalusan citra dan penajaman citra dipengaruhi oleh tingkat terang gelapnya cahaya dari citra asli.
b. Pada penghalusan citra, peranan matriks *mask* sangat memengaruhi dalam prosesnya, terutama jika citra asli yang digunakan memiliki banyak gangguan *(noise)*.

Hasil akan lebih nampak jika matriks mask berukuran semakin besar.

c. Pada proses penajaman citra peranan komponen warna dari objek citra menentukan adanya kemunculan garis – garis sisi objek yang dapat dibedakan oleh mata.

BAB 4 Segmentasi Berdasarkan Warna

4. Segmentasi Berdasarka Warna

Segmentasi merupakan salah satu operasi dalam analisis citra yang mempartisi citra menjadi beberapa segmen atau objek. Segmentasi juga merupakan proses yang penting dalam pengenalan pola karena dapat menyederhanakan penyajian gambar ke sesuatu yang lebih bermakna sehingga lebih mudah untuk dianalisis. Segmentasi citra memisahkan suatu wilayah objek dengan wilayah objek lainnya yang bukan menjadi perhatian seperti wilayah latar belakang. Sehingga hasil dari objek yang telah tersegmentasi tersebut dapat digunakan untuk pemrosesan citra selanjutnya seperti deteksi tepi, pengenalan pola dan lainnya.

Sifat dari segmentasi citra pada umumnya didasarkan pada *diskontinuitas* atau *similaritas*. Pada diskontinuitas, citra dibagi berdasarkan perbedaan intensitasnya, misalnya *edge* (tepian) dari citra. Sedangkan pada similaritas, citra dibagi menjadi beberapa wilayah berdasarkan tingkat kemiripan yang dimiliki. Pada proses segmentasi citra, ada beberapa metode yang biasa digunakan seperti metode *thresholding, shapebased, region growing* dan metode *clustering*.

Dalam suatu citra, piksel-piksel yang saling berdekatan biasanya saling berhungan dan memiliki nilai fitur yang hampir mirip. Sehingga besar kemungkinan bahwa piksel-piksel yang berdekatan tersebut berada dalam *cluster* yang sama. Oleh karena

itu, pada penelitian ini digunakan metode *clustering* untuk segmentasi citra peta. Teknik c*lustering* dapat digunakan untuk segmentasi citra berdasarkan warna. *Clustering* dapat mengelompokkan data-data yang memiliki tingkat kemiripan yang tinggi ke dalam *cluster* atau kelas yang sama. Salah satu metode dalam teknik *clustering* adalah *Fuzzy c-Means* (FCM) yang dapat mendefinisikan nilai derajat keanggotaan untuk mendapatkan perhitungan *clustering*.

4.1 Model

Langkah-langkah dari metode yang digunakan :

1. Input gambar yang akan diuji pada cluster X (matriks a x b).

$$X = \begin{matrix} x_{11} & \dots & x_{1b} \\ \dots & \dots & \dots \\ x_{a1} & \dots & x_{ab} \end{matrix} \qquad (4.1)$$

Keterangan :

m = banyak data-data yang akan dikelompokkan

n = banyak variable

X_{ij} = data ke-i (I = 1,2,3, … a) dan data ke-j (j = 1,2,3, … b).

2. Tentukan variabel :

- Jumlah dari *cluster* yang akan dibentuk C = ≥ 2.
- Iterasi maksimum = MaxIter.
- Error minimum yang diharapkan = ε.

- Fungsi objektif awal ($P_0 = 0$).
- Iterasi awal (t = 1).

3. Mentriger bilangan acak μ_{ik} untuk komponen matriks partisi awal U. Dimana i = (1, 2, 3, … , a) dan k = (1, 2, 3, … c). Menghitung jumlah setiap atribut :

$$\mu_{ik} = \frac{\mu_{ik}}{Q_j} \qquad (4.2)$$

4. Menghitung nilai pusat *cluster* ke-*k* = $V_{kj.}$ Dimana k = (1, 2, 3, … c) dan j = (1, 2, 3, …, b).

$$V_{kj} = \frac{\sum_{i=1}^{a}((\mu_{ik})^{w} X_{ij}}{\sum_{ij}^{a}(\mu_{ik})^{w}} \qquad (4.3)$$

$$V = \begin{matrix} v_{11} & \cdots & v_{1b} \\ \cdots & \cdots & \cdots \\ v_{a1} & \cdots & v_{ab} \end{matrix} \qquad (4.4)$$

5. Menghitung nilai fungsi objektif dari iterasi ke-*t*.

$$P_t = \sum_{i=1}^{a}\sum_{k=1}^{c}([\sum_{j=1}^{b} X_{ik-V_{kj}})^2](\mu_{ik})^w \qquad (4.5)$$

6. Hitung perubahan matriks partisi :

$$\mu_{ik} = \frac{[\sum_{j=1}^{b}(X_{ik}-V_{kj})^2]^{\frac{-1}{w-1}}}{\sum_{j=1}^{b}[\sum_{j=1}^{b}(X_{ik}-V_{kj})^2]^{\frac{-1}{w-1}}} \qquad (4.6)$$

Dimana i = (1, 2, 3, … , a) dan k = (1, 2, 3, … c).

- Periksa kondisi berhenti.
- Saat $P_t - P_{t-1} < \varepsilon$ atau t > Maciter maka stop.
- Jika belum (t = t+1) maka ulangi langkah ke-4.

Hasil pengelompokan dari metode ini akan berbentuk bulat karena semua kelompok diasumsikan memiliki ukuran yang sama besar.

4.2 Algoritma

Langkah – langkah dalam melakukan segmentasi citra.

1. Tahap *Preprocessing*.
2. Hitung nilai pusat *cluster* dan nilai keanggotaan *cluster*.

a. Inisialisasi nilai awal seperti jumlah *cluster,* bobot, maksimum iterasi, error minimum, fungsi objektif dan iterasi awal.
b. Menghitung matriks partisi U_{ik}.
c. Menghitung pusat *cluster*.
d. Menghitung fungsi objektif.
e. Perbarui matriks partisi μ.

3. Menggabungkan nilai *Membership Function* dan *Spatial Function.*
4. Menghitung nilai *fitness.*
5. Periksa kondisi berhenti, ulangi langkah 2 jika kondisi berhenti belum terpenuhi.
6. Lakukan ekstrasi citra yang disegmentasi.
7. Lakukan pengujian dan evaluasi dari hasilnya.

4.3 Contoh Perhitungan

Langkah 1 :

Data ke-i	Atribut		
	A1	B2	C3
1	12	7	9
2	5	4	5
3	8	11	4
4	10	3	8
5	9	1	3

Langkah 2 : Inisialisasi

banyaknya cluster (c)	2
Pembobot (w)	2
maksimum iterasi (MaxIter)	5
error (e)	0,01
fungsi objektif (P0)	0
Iterasi awal (Iter)	1

Langkah 3 : Matriks U_{ic} dimana i = banyak data dan c = jumlah *cluster*.

I	k1	k2
1	0,3	0,7
2	0,2	0,8
3	0,4	0,6
4	0,8	0,2
5	0,4	0,6

Langkah 4 : Menghitung pusat *cluster*.

I	Uik		Xij			
	1	2	1	2	3	Ui1^w
1	0,3	0,7	12	7	9	0,09
2	0,2	0,8	5	4	5	0,04
3	0,4	0,6	8	11	4	0,16
4	0,8	0,2	10	3	8	0,64
5	0,4	0,6	9	1	3	0,16
						1,09

Pusat *cluster*-nya

Vkj	1	2	3
1	9,541,284,404	4,247,706,422	6,651,376,147
2	8,253,968,254	5,518,518,519	5,529,100,529

Langkah 5 : Menghitung fungsi objektif.

I	cluster-1				cluster-2	
	(Xi1-Vi1)^2	(Xi2-Vi1)^2	(Xi3-Vi1)^2	Total1	(Xi1-Vi2)^2	(Xi2-Vi2)^2
1	604,528,238	7,575,119,939	5,516,034,004	1,913,643,633	140,327,538	219,478,738
2	20,623,264	0,061358472	2,727,043,178	2,341,166,568	105,883,094	230,589,849
3	237,555,761	4,559,346,856	7,029,795,472	5,499,882,165	0,06449987	300,466,392
4	0,21042	1,556,771,316	1,818,786,297	3,585,977,611	304,862,686	634,293,553
5	0,29298881	10,547,597	1,333,254,777	2,417,313,357	0,55656337	204,170,096

Langkah 6 : Perbarui nilai U.

	(Xi1-Vi1)^2	(Xi2-Vi1)^2	(Xi3-Vi1)^2	(Xi1-Vi2)^2	(Xi2-Vi2)^2	(Xi3-Vi2)^2
1	6,04528238	7,575119939	5,516034004	14,03275384	2,19478738	12,0471431
2	20,623264	0,061358472	2,727043178	10,5883094	2,30589849	0,27994737
3	2,37555761	45,59346856	7,029795472	0,06449987	30,04663932	2,33814843
4	0,21042	1,556771316	1,818786297	3,048626858	6,34293553	6,1053442
5	0,29298881	10,547597	13,33254777	0,556563366	20,41700956	6,39634949

Maka didapatkan nilai U baru

i	k1	k2
1	0,4	0,6
2	0,6	0,4
3	0,6	0,4
4	0,2	0,8
5	0,5	0,5

Langkah 7 : Periksa kondisi berhenti.

Pada proses di atas, perhitungan belum memenuhi syarat berhenti sehingga ulangi langkah 4.

Setelah melakukan 5 kali iterasi maka didapatkan pusat *cluster* sebagai berikut :

10,1071	4,0943	7,157
7,2219	6,6961	4,5294

U =

0,26	0,74
0,71	0,29
0,76	0,24
0,05	0,95
0,43	0,57

Berdasarkan matriks diatas dapat diketahui pengelompokkan data setelah dilakukan 5 kali iterasi sebagai berikut

i	Cluster 1	Cluster 2
1	✓	
2		✓
3		✓
4		✓
5	✓	

Ui2^w	(Ui1^w)*Xi1	(Ui1^w)*Xi2	(Ui1^w)*Xi3	(Ui2^w)*Xi1	(Ui2^w)*Xi2	(Ui3^w)*Xi3
0,49	1,08	0,63	0,81	5,88	3,43	4,41
0,64	0,2	0,16	0,2	3,2	2,56	3,2
0,36	1,28	1,76	0,64	2,88	3,96	1,44
0,04	6,4	1,92	5,12	0,4	0,12	0,32
0,36	1,44	0,16	0,48	3,24	0,36	1,08
1,89	10,4	4,63	7,25	15,6	10,43	10,45

ter 2		Kluster1		Kluster2		P Kluster
Xi3 Vi2 ^2	Total2	Uik ^w	P	Uik ^w	P	
120,47 1,431	282,74 6,844	0,0 9	172,22 7,927	0,4 9	138,54 5,953	155,76 8,746
0,2799 4737	131,74 1,553	0,0 4	0,9364 6663	0,6 4	843,14 5,937	936,79 2,599
233,81 4,843	324,49 2,875	0,1 6	879,98 1,146	0,3 6	116,81 7,435	20,481, 555
61,053, 442	154,96 9,066	0,6 4	229,50 2,567	0,0 4	0,6198 7626	291,49 0,193
639,63 4,949	273,69 9,225	0,1 6	386,77 0,137	0,3 6	985,31 7,208	137,20 8,735

Total	U_{i1}	U_{i2}
474,111,207	0,40362759	0,59637241
365,858,209	0,6399109	0,3600891
874,481,092	0,62893094	0,37106906
190,828,842	0,18791591	0,81208409
51,543,056	0,46898914	0,53101086

4.4 Java Source Code

- *Image Clustering* dengan *FCM*

```
1.  public FCMImageClustering(PlanarImage input,int
2.  nClusters,int MaxIter,  float fuzzy,double epsilon)
3.  {
4.     this.input = input;
5.     width = input.getWidth();
6.     height = input.getHeight();
7.     numBands = input.getSampleModel().getNumBands();
8.     this.nClusters = nClusters;
9.     this.MaxIter = MaxIter;
10.    this.fuzzy = fuzzy;
11.    this.epsilon = epsilon;
12.    iteration = 0;
13.    PusatCluster = new float[nClusters][numBands];
14.    2mbership = new float[width][height][nClusters];
15.    Raster raster = input.getData();
16.    inputData = new int[width*height*numBands];
17.    aPixel = new float[numBands];
18.    outputData = new short[width][height];
19.    raster.getPixels(0,0,width,height,inputData);
20.    Random generator = new Random();
21.    5
22.    for(int p=0;p<height;p++)
23.    for(int q=0;q<width;q++)
24.    { 2
25.       float sum = 0f;
26.       for(int c=0;c<nClusters;c++)
27.       {
```

```
28.          membership[q][p][c] = 0.01f+generator.nextFloat();
29.          sum += membership[q][p][c];
30.        2
31.        for(int c=0;c<nClusters;c++) membership[q][p][c] /=
32.        sum;
33.      }
34.     position = 0;
35. }
36.
```

- Menghitung Pusat *Cluster*

```
1.  private void HitungPusatCluster ()
2.  {
3.    float atas,bawah;
4.    2
5.    for(int b=0;b<numBands;b++)
6.    for(int c=0;c<numClusters;c++)
7.    {                    2
8.      atas = bawah = 0;
9.      for(int p=0;p<height;p++)
10.     for(int q=0;q<width;q++)
11.     {
12.       int index = (p*width+q)*numBands;
13.       atas +=
14.       Math2ow(membership[q][p][c],fuzziness)*inputD
15.       ata[index+b];
16.       bawah +=
17.       Math.pow(membership[q][p][c],fuzziness);
18.     }
19.
20.     clusterCenters[c][b] = atas/bawah;
21.     position += width*height;
22.   }
23. }
```

- Menghitung *Member Function*

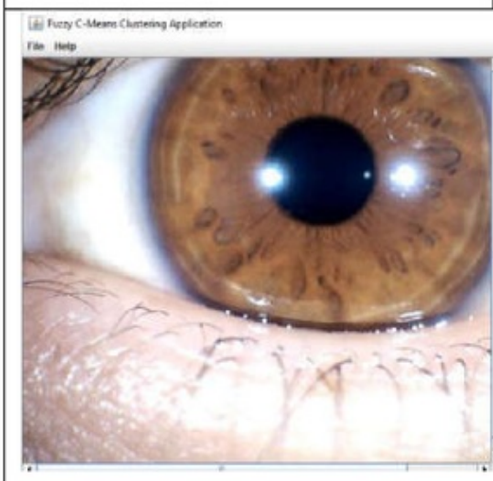
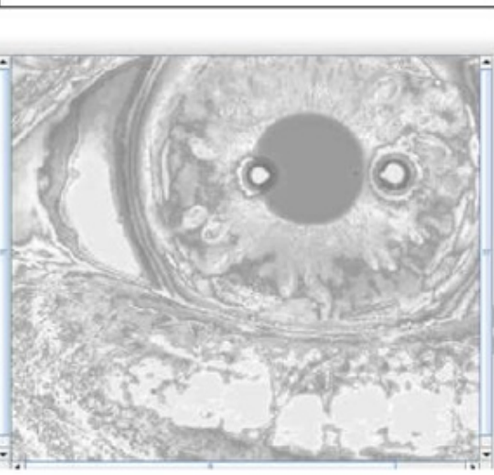
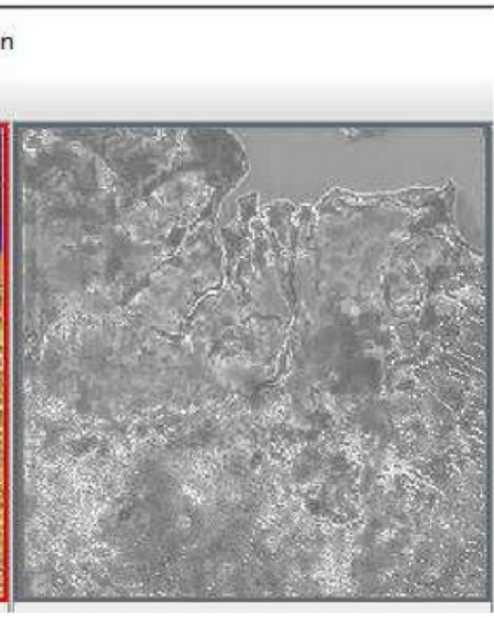
```
1.  private void calculateMFsFromClusterCenters()
2.  {
3.    float sumTerms;
4.    2
5.    for(int c=0;c<numClusters;c++)
6.    for(int p=0;p<height;p++)
7.    for(int q=0;q<width;q++)
8.    { 2
9.      int index = (p*width+q)*numBands;
10.     for(int b=0;b<numBands;b++)
11.     aPixel[b] = inputData[index+b];
12.     2oat top = calcDistance(aPixel,clusterCenters[c]);
13.     sumTerms = 0f;
14.     for(int ck=0;ck<numClusters;ck++)
15.     {
16.       float thisDistance =
17.       calcDistance(aPixel,clusterCenters[ck]);
18.       sumTerms +=
          Math.pow(top/thisDistance,(2f/(fuzziness-1f)));
19.     }
20.   membership[q][p][c] = (float)(1f/sumTerms);
21.   position += (numBands+numClusters);
22.   }
23. }
```

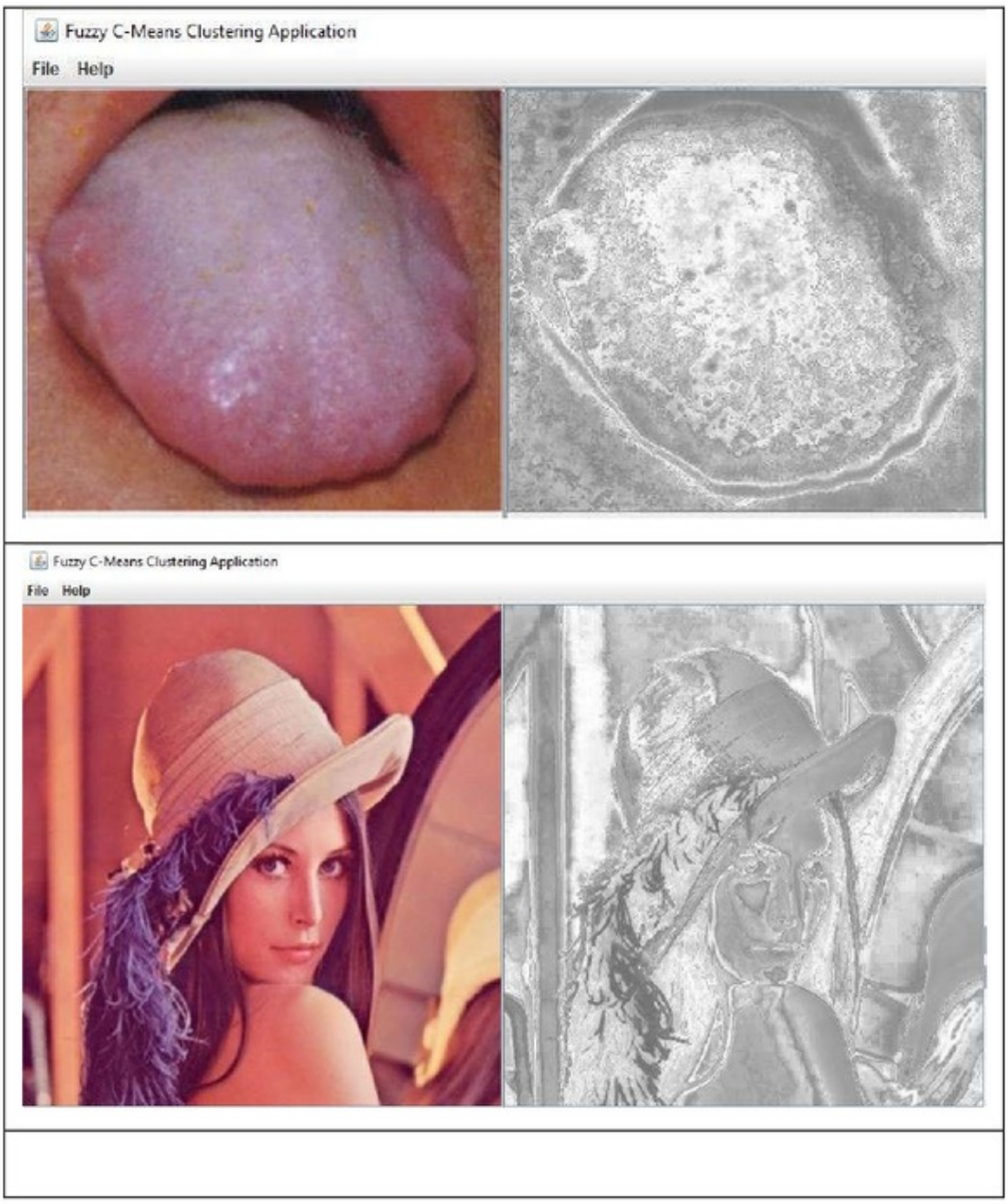
- Menghitung Fungsi Objektif

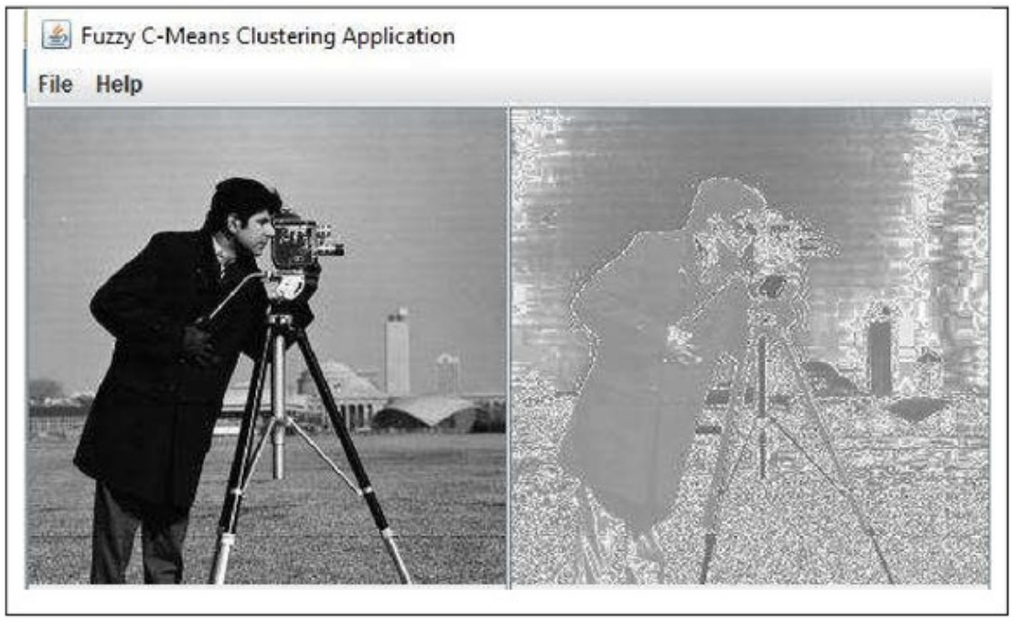
```
1.  private double HitungFungsiObjective()
2.  {
3.    5uble j = 0;
4.    for(int p=0;p<height;p++)
5.    for(int q=0;q<width;q++)
6.    for(int c=0;c<numClusters;c++)
7.    { 2
8.      int index = (p*width+q)*numBands;
9.      for(int b=0;b<numBands;b++)
10.     2ixel[b] = inputData[index+b];
11.     float distancePixelToCluster = calcDistance
12.     (aPixel,clusterCenters[c]);
13.     j += distancePixelToCluster*Math.pow
14.     (membership[q][p][c],fuzziness);
15.     position += (2*numBands);
16.   }
17.   return j;
18. }
```

4.5 Hasil Percobaan

Hasil percobaan *Segmentasi Citra*

Citra Awal	Citra Hasil Segmentasi
	
	


Fuzzy C-Means Clustering Application
File Help
Fuzzy C-Means Clustering Application
File Help



4.6 Kesimpulan

Dari penelitian ini dapat diketahui bahwa untuk melakukan segmentasi citra berwarna dapat digunakan teknik *clustering* untuk melakukan pengelompokkan data-data yang memiliki tingkat kemiripan yang tinggi dan mencari nilai pusat *cluster*.

BAB 5 Kebisingan Citra Berwarna dan Kompresi Citra Berwarna

5 Kebisingan Citra Berwarna dan Kompresi Citra Berwarna

Noise dapat didefenisikan sebagai suatu kecacatan, kekurangan atau gangguan yang terjadi pada suatu citra menyebabkan kekurangan informasi yang sesungguhnya pada citra berupa blur, banyak bintik-bintik, tidak jelas dan lain sebagainya. Hal ini dapat diakibatkan efek dari sifat fisika seperti energi cahaya berupa foton atau sensor kamera yang panas. Contohnya, saat mengambil citra dengan camera CCD, foton cahaya dan juga panas sensor kamera jadi penyebab banyak sedikitnya noise yang dihasilkan. Gangguan selam transmisi juga termasuk penyebab banyaknya noise.

Biasanya noise pada image berwarna punya karakteristik yang sama pada semua color channel, tapi mungkin juga punya perbedaan yang signifikan. Namun, perbedaan level noise lebih dikarenakan oleh perbedaan kekuatan penerangan (illumination) yang mungkin pada tiap color channel.

Image compression dapat didefenisikan sebagai pemampatan data terhadap suatu image untuk memadatkan data atau mengurangi redudansi data dalam suatu image sehingga dapat disimpan dan hanya memerlukan penyimpanan yang kecil atau ditransmisikan secara efisien.

5.1 Model dan Contoh

5.1.1 Gaussian Noise

Gaussian merupakan noise yang ditambahkan secara alami mengikuti distribusi Gaussian. disebabakan intensitas cahaya rendah, suhu tinggi, dan saaat transmisi selama proses akuisisi (Indepreet, 2014). Jika citra dinyatakan dengan I dan Gaussian dinyatakan dengan N, maka citra terkosupsi oleh Gausian noise dinyatakan dengan cara menambahkan keduanya yaito I + N (Susilawati, 2009). *Probability density function* (PDF) atau fungsi kepadatan probabilitas noise Gaussian ditunjukkan dengan persamaan :

$$p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(z-\mu)^2}{2\sigma^2}} \qquad (5.1)$$

Dimana :

z = intensitas piksel

σ^2 = variance dari z

σ = standar deveasi

μ = nilai rata-rata piksel z

Gambar 5.1 Citra Mengandung *Gaussian Noise*

5.1.2 Impuls Noise (Salt-and-paper noise)

Salt and paper noise juga sering disebut *Impuls noise, Shot noise,* atau *Binery noise*. Disebabkan gangguan yang tajam dan tiba-tiba *(Sharp and sudden)* pada sinyal citra. Citra akan tampak berupa titik-titik (piksel) hitam atau putih (atau kedua-duanya) yang tersebar pada citra (Susilawati, 2009). Penyebabnya adalah piksel yang rusak pada sensor kamera, lokasi memori rusak pada perangkat kearas atau proses transmisi pada saluran noise. Noise ini selalu independen dan tidak memiliki hubungan dengan piksel citra (Indepreet, 2014). *Probability Desenty Function* (PDF) atau fungsi kepadatan probabilitas *Salt-and-paper noise* ditujukkan dengan persamaan :

$$p(z) = \begin{cases} Pa \\ Pb \\ 0 \end{cases}$$

Untuk z = a

Untuk z = b dst..

Gambar 5.2 Citra Mengandung *Salt and Paper*

5.1.3 Spackle Noise

Spackle noise umunya mendegradasi sebagian besar citra *Synthetic Aperture Radar* (SAR). Disebabkan oleh naik turunnya sinyal yang datang kembali dari sebuah benda yang lebih kecil dari elemen pengolahan citra tunggal. Spackle noise meningkatkan rata-rata dari skala abu-abu yang mempengaruhi citra sehingga menciptakan banyak kesulitan dalam menafsirkan gambar.

Gambar 5.3 Citra Mengandung *Speckle Noise*

5.1.4 Poison Noise

Poison noise dapat terjadi ketika jumlah foton yang diterima oleh sensor tidak cukup untuk memberikan informasi statistik yang dapat dideteksi. Poison noise memiliki nilai akar kuadrat citra (Patidar, 2010). *Probability Density Function* (FDR) atau fungsi kepadatan probabilitas poison noise ditunjukan dengan persamaan :

$$P(x) = \frac{e^{-\lambda}\lambda^{x}}{x!} \; untuk \; \lambda > 0 \; dan \; x = 0,1,2 \ldots \qquad (5.2)$$

Dimana :

x = intensitas pixel

e= nilai exponent (2,71828...)

Gambar 5.4 Citra Mengandung Poison Noise

5.2 Algoritma

Berikut algoritma dari Gaussian *Noise,*

1. Represintasikan nilai piksel matriks citra pada sebuah array awal, citra(p,q), dimana p dan q adalah panjang piksel citra awal pxq.
2. Inisialisai nilai random untuk variabel mean (μ) dan standar deviasi (σ).
3. Hitung nilai pembagi $\sqrt{2\pi\sigma}$.
4. Hitung setiap nilai piksel citra(p,q)* exp($-(z-\mu)^2/2\sigma^2$), ulangi sampai seluruh koordinat citra awal terpenuhi. Exp sendiri bernilai 2,7182.
5. Hitung nilai akhir citra Gaussian, G(p,q) atau P(Zp,q), setiap piksel nya didapat dari hasil nilai koordinat piksel pada langkah 4 dibagi langkah 3.

5.3 Contoh Perhitungan

Pada awal tahun 2011, sebuah instansi dalam negeri memiliki total pegawai majerial dengan jumlah 1500 orang. Data penyebaran diketahui sebagai distribusi Normal atau Gaussian dengan standar deviasi 12,36 tahun dan umur rata-rata 40,25 tahun. Seorang pegawai akan habis tugas/pension pada umur > 56 tahun. Hitunglah jumlah pegawai yang akan pension pada akhir tahun 2011?

Perhitungan dengan manual:

Rata-rata umur (μ) = 40,25

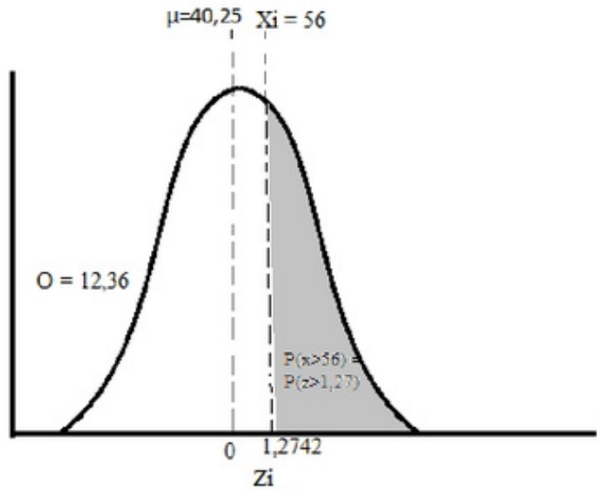
Standar Deviasi (σ) = 12.36

Batas usia, $P(x_i \geq 56)$

Jadi, perhitungan probabilitas:

$$z = (56 - 40.25)/12.36$$

$$= 1.2742$$



Kurva Gaussian

Perhatikan gambar diatas, nilai kemungkinan yang kita cari $P(X \geq 56)$ / $P(\geq z)$, yaitu perlusan yang dibatasi oleh garis $F(x, \mu, \sigma)$ dan Zi.

Selanjutnya, akan dicari probabilitas Z sesuai tabel probabilitas Z.

Z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.0000	0.0040	0.0080	0.0120	0.0160	0.0199	0.0239	0.0279	0.0319	0.0359
0.1	0.0398	0.0438	0.0478	0.0517	0.0557	0.0596	0.0636	0.0675	0.0714	0.0753
0.2	0.0793	0.0832	0.0871	0.0910	0.0948	0.0987	0.1026	0.1064	0.1103	0.1141
0.3	0.1179	0.1217	0.1255	0.1293	0.1331	0.1368	0.1406	0.1443	0.1480	0.1517
0.4	0.1554	0.1591	0.1628	0.1664	0.1700	0.1736	0.1772	0.1808	0.1844	0.1879
0.5	0.1915	0.1950	0.1985	0.2019	0.2054	0.2088	0.2123	0.2157	0.2190	0.2224
0.6	0.2257	0.2291	0.2324	0.2357	0.2389	0.2422	0.2454	0.2486	0.2517	0.2549
0.7	0.2580	0.2611	0.2642	0.2673	0.2703	0.2734	0.2764	0.2794	0.2823	0.2852
0.8	0.2881	0.2910	0.2939	0.2967	0.2995	0.3023	0.3051	0.3078	0.3106	0.3133
0.9	0.3159	0.3186	0.3212	0.3238	0.3264	0.3289	0.3315	0.3340	0.3365	0.3389
1.0	0.3413	0.3438	0.3461	0.3485	0.3508	0.3531	0.3554	0.3577	0.3599	0.3621
1.1	0.3643	0.3665	0.3686	0.3708	0.3729	0.3749	0.3770	0.3790	0.3810	0.3830
1.2	0.3849	0.3869	0.3888	0.3907	0.3925	0.3944	0.3962	0.3980	0.3997	0.4015
1.3	0.4032	0.4049	0.4066	0.4082	0.4099	0.4115	0.4131	0.4147	0.4162	0.4177
1.4	0.4192	0.4207	0.4222	0.4236	0.4251	0.4265	0.4279	0.4292	0.4306	0.4319
1.5	0.4332	0.4345	0.4357	0.4370	0.4382	0.4394	0.4406	0.4418	0.4429	0.4441
1.6	0.4452	0.4463	0.4474	0.4484	0.4495	0.4505	0.4515	0.4525	0.4535	0.4545
1.7	0.4554	0.4564	0.4573	0.4582	0.4591	0.4599	0.4608	0.4616	0.4625	0.4633
1.8	0.4641	0.4649	0.4656	0.4664	0.4671	0.4678	0.4686	0.4693	0.4699	0.4706
1.9	0.4713	0.4719	0.4726	0.4732	0.4738	0.4744	0.4750	0.4756	0.4761	0.4767
2.0	0.4772	0.4778	0.4783	0.4788	0.4793	0.4798	0.4803	0.4808	0.4812	0.4817
2.1	0.4821	0.4826	0.4830	0.4834	0.4838	0.4842	0.4846	0.4850	0.4854	0.4857
2.2	0.4861	0.4864	0.4868	0.4871	0.4875	0.4878	0.4881	0.4884	0.4887	0.4890
2.3	0.4893	0.4896	0.4898	0.4901	0.4904	0.4906	0.4909	0.4911	0.4913	0.4916
2.4	0.4918	0.4920	0.4922	0.4925	0.4927	0.4929	0.4931	0.4932	0.4934	0.4936
2.5	0.4938	0.4940	0.4941	0.4943	0.4945	0.4946	0.4948	0.4949	0.4951	0.4952
2.6	0.4953	0.4955	0.4956	0.4957	0.4959	0.4960	0.4961	0.4962	0.4963	0.4964
2.7	0.4965	0.4966	0.4967	0.4968	0.4969	0.4970	0.4971	0.4972	0.4973	0.4974
2.8	0.4974	0.4975	0.4976	0.4977	0.4977	0.4978	0.4979	0.4979	0.4980	0.4981
2.9	0.4981	0.4982	0.4982	0.4983	0.4984	0.4984	0.4985	0.4985	0.4986	0.4986
3.0	0.4987	0.4987	0.4987	0.4988	0.4988	0.4989	0.4989	0.4989	0.4990	0.4990

Tabel Probabilitas Z

Berdasarkan tabel Z diatas, diketahui probabiitas Z yaitu:

$P(z \geq 1.2742) = 1 - (0.5 + 0.3987) = 0.1012$

Jadi, jumlah pegawai yang akan lulus akhir tahun 2011 yaitu berjumlah

$0.1012 * 1500 = 152$ orang

5.4 Java Source Code

```
20 package imageviewer.NoiseFilters;
2. import java.awt.image.*;
3. lic class NoiseFilter extends Filter {
30
4. public final static int IMPULSE = 0;
5. public final static int GAUSSIAN = 1;
6. protected int noiseType = IMPULSE;
7. protected double stdDev = 10.0;
8. protected double impulseRatio = 0.05;
9. public NoiseFilter() {
10. }
11. public NoiseFilter(int noiseType) {
12. setNoiseType(noiseType);
13. }
14. public NoiseFilter(int noiseType, double parameter) {
15. setNoiseType(noiseType);
16. if (noiseType == IMPULSE) setImpulseRatio(parameter);
17. if (noiseType == GAUSSIAN) setGaussianStdDev(parameter);
18. }
19. public void setNoiseType(int noiseType) {
20. this.noiseType = noiseType;
21. }
22. public int getNoiseType() {
```

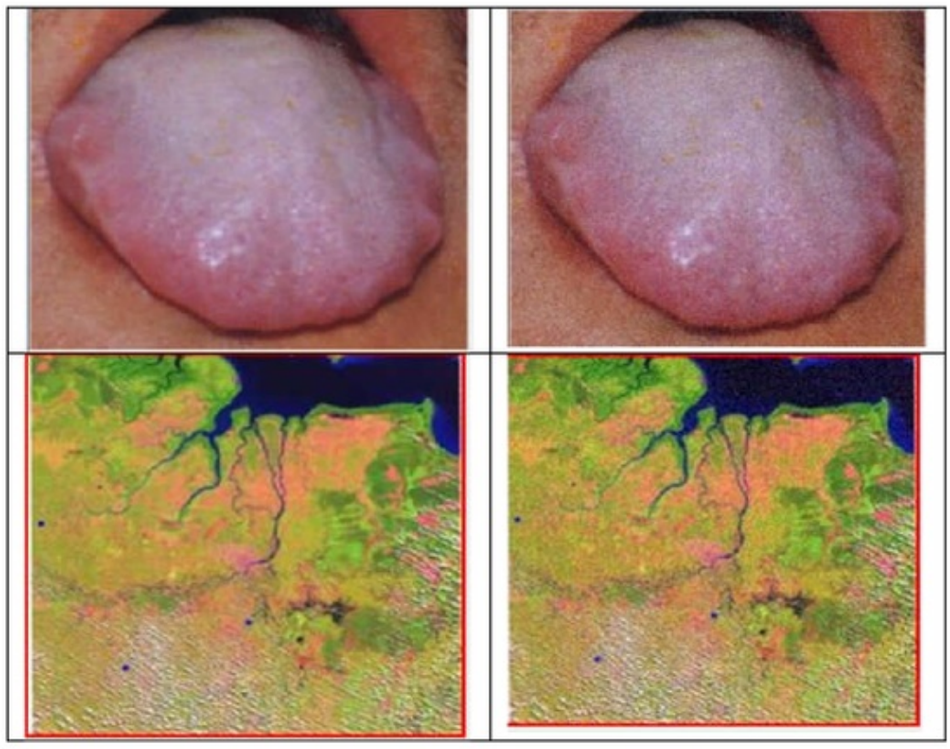
```
23. return noiseType;
24. }
25. public void setGaussianStdDev(double stdDev) {
26. this.stdDev = stdDev;
27. }
28. public double getGaussianStdDev() {
29. return stdDev;
30. }
31. public void setImpulseRatio(double impulseRatio) {
32. this.impulseRatio = impulseRatio;
33. }
34. public double getImpulseRatio() {
35. return impulseRatio;
36. }
37. public java.awt.image.BufferedImage filter(BufferedImage
    image, BufferedImage output) {
38. output = verifyOutput(output, image);
39. switch (noiseType) {
40. default:
41. case IMPULSE: return impulseNoise(image, output);
42. case GAUSSIAN: return gaussianNoise(image, output);
43. }
44. }
45. protected BufferedImage impulseNoise(BufferedImage image,
    BufferedImage output) {
46. output.setData(image.getData());
47. Raster source = image.getRaster();
48. WritableRaster out = output.getRaster();
49. double rand;
50. double halfImpulseRatio = impulseRatio / 2.0;
51. int bands  = out.getNumBands();
52. int width= image.getWidth();  // width of the image int
    height= image.getHeight(); // height of the image
53. java.util.Random randGen = new java.util.Random();
54. for (int j=0; j<height; j++) {
55. for (int i=0; i<width; i++) {
56. rand = randGen.nextDouble();
```

```
57. if (rand < halfImpulseRatio) {
58. for (int b=0; b<bands; b++) out.setSample(i, j, b, 0);
59. } else if (rand < impulseRatio) {
60. for (int b=0; b<bands; b++) out.setSample(i, j, b, 255);
61. }
62. }
63. }
64. return output;
65. }
66. protected BufferedImage gaussianNoise(BufferedImage image,
    BufferedImage output) {
67. Raster source = image.getRaster();
68. WritableRaster out = output.getRaster();
69. int currVal;             // the current value
70. double newVal;        // the new "noisy" value
71. double gaussian;     // gaussian number
72. int bands  = out.getNumBands(); // number of bands
73. int width  = image.getWidth();  // width of the image int
    height = image.getHeight(); // height of the image
74. java.util.Random randGen = new java.util.Random();
75. for (int j=0; j<height; j++) {
76. for (int i=0; i<width; i++) {
77. gaussian = randGen.nextGaussian();
78. for (int b=0; b<bands; b++) {
79. newVal = stdDev * gaussian;
80. currVal = source.getSample(i, j, b);
81. newVal = newVal + currVal;
82. if (newVal < 0)   newVal = 0.0;
83. if (newVal > 255) newVal = 255.0;
84. out.setSample(i, j, b, (int)(newVal));
85. }
86. }
87. }
88. return output;
89. }
90. }
```

5.5 Hasil Percobaan

Hasil Percobaan *Gaussian Noise*

Citra Awal	**Hasil *Gaussian Noise***
	
	
	



5.6 Kesimpulan

Citra noise ternyata bukan hanya berlaku pada image grayscale, tetapi bisa juga diaplikasikan pada image berwarna. Noise dan denoising dilakukan demi mendapatkan informasi yang kurang pada suatu image. Image compression juga sangat dibutuhkan saat ini dikarenakan kemajuan teknologi yang semakin canggih, semakin baik pula image yang dihasilkan saat menangkap image dengan kamera. Semakin baik suatu image semakin besar juga data dan ruang penyimpanan yang dibutuhkan. Oleh sebab itu, compression dibutuhkan untuk menghemat penyimpanan data dan juga demi efisiennya transmisi data.

DAFTAR PUSTAKA

Chen, S. Y., Hsieh, J. W., & Chen, D. Y. (2010). Cross-view object identification using principal color transformation. *2010 International Conference on Machine Learning and Cybernetics, ICMLC 2010*, *6*(July), 2777–2781. http://doi.org/10.1109/ICMLC.2010.5580787

Chiang, C., Tai, W., Yang, M., Huang, Y., & Huang, C. (2003). A novel method for detecting lips , eyes and faces in real time, *9*, 277–287. http://doi.org/10.1016/j.rti.2003.08.003

Choi, W.-J., Choi, T.-S., Hasanabadi, H., Zabihi, M., Mirsharif, Q., Ye, X., … Kandwal, R. (2013). Automated Pulmonary Nodule Detection System in Computed Tomography Images: A Hierarchical Block Classification Approach. *IEEE Transactions on Biomedical Engineering*, *5*(1), 507–523. http://doi.org/10.1109/TBME.2009.2017027

Darmaga, K., Proyek, B., Kualitas, P., Manusia, S., Jenderal, D., Tinggi, P., & Nasional, D. P. (2006). M o K u Penggunaan Warna Mode dan Pemodelan Warna.

Davies, R. (2012). *Computer and Machine Vision, 4th*

Edition Theory, Algorithms, Practicalities Opsylum. Zhurnal Eksperimental'noi i Teoreticheskoi Fiziki.

Duangphasuk, P., & Kurutach, W. (2013). Tattoo skin detection and segmentation using image negative method. *2013 13th International Symposium on Communications and Information Technologies (ISCIT)*, 354–359. http://doi.org/10.1109/ISCIT.2013.6645881

Gonzalez, R. C., Woods, R. E., & Masters, B. R. (2007). Digital Image Processing, Third Edition, (December), 976. http://doi.org/10.1117/1.3115362

Han, M., & Chen, C. (2016). Enhancing underwater image by dark channel prior and color correction. *6th International Conference on Information Science and Technology, ICIST 2016*, 505–510. http://doi.org/10.1109/ICIST.2016.7483466

Khodambashi, S., & Moghaddam, M. E. (2009). An impulse noise fading technique based on local histogram processing. *IEEE International Symposium on Signal Processing and Information Technology, ISSPIT 2009*, 95–100. http://doi.org/10.1109/ISSPIT.2009.5407484

Lestari, R. (2012). Analisis Dan Perancangan Perangkat Lunak Kompresi Citra Menggunakan Algoritma Fast

Fourier Transform (Fft), 6.

Lucchese, L., Mitra, S. K., & Barbara, S. (n.d.). Color Image Segmentation : A State-of-the-Art Survey.

Munir, R. (2004). Operasi - operasi Dasar Pengolahan Citra Dijital. *Pengolahan Citra Digital Dengan Pendekatan Algoritmik*, 41–60.

Nagai, Y., Uchida, Y., Myodo, E., & Sakazawa, S. (2013). A color transformation method based on color theme that takes constraints on color ratio and spatial coherence into consideration. *2013 IEEE International Conference on Image Processing, ICIP 2013 - Proceedings*, 108–112. http://doi.org/10.1109/ICIP.2013.6738023

Nakajima, N., & Taguchi, A. (2014). A novel color image processing scheme in HSI color space with negative image processing. *2014 International Symposium on Intelligent Signal Processing and Communication Systems, ISPACS 2014*, (1), 29–33. http://doi.org/10.1109/ISPACS.2014.7024419

Othman, A. H. C., & Sabudin, M. (2015). A study of colour transformation for colour deficient individual. *Proceeding - 2013 IEEE Student Conference on Research and Development, SCOReD 2013*, (December), 328–333.

http://doi.org/10.1109/SCOReD.2013.7002601

Prasad, K. S. (2015). A Novel Medical Image Fusion with l αβ Color Transformation.

Qian, Y., Zhou, L., & Huang, Y. (2012). Color Human Slice Segmentation based on Texture Information, (Fskd), 2620–2623.

Unal, B., & Akoglu, A. (n.d.). Resource Efficient Real-Time Processing of Contrast Limited Adaptive Histogram Equalization.

Indeks

Biodata Penulis

Pemrosesan Citra Berwarna & Aplikasi Dengan JAVA

Erwin, S.Si., M. Si., Lahir di Palembang, tanggal 29 Januari 1971, menyelesaikan pendidikan pada jenjang Sarjana tahun 1994 di Program Studi Matematika Fakultas MIPA, Universitas Sriwijaya dan melanjutkan studi di Program Magister Institut Teknologi Bandung pada Program Studi Aktuaria, Jurusan Matematika ITB serta lulus tahun 2001. Saat ini, sebagai dosen tetap di Program Studi Sistem Komputer Fakultas Ilmu Komputer Universitas Sriwijaya dengan bidang riset Image Processing dan Computer Vision dan sebagai salah seorang pendiri **Sriwijaya Imaging Science Research Group**

Muhammad Fachrurrozi Lahir di Palembang, tanggal 22 Mei 1980, menyelesaikan pendidikan pada jenjang Sarjana tahun 2002 di Program Studi Matematika Fakultas MIPA, Universitas Sriwijaya dan pada tahun 2006 melanjutkan studi di Program Magister Institut Teknologi Bandung pada Program Studi Informatika, Sekolah Teknik Elektro dan Informatika (STEI) Institut Teknologi Bandung (ITB) serta lulus tahun 2008. Saat ini, sebagai dosen tetap di Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Sriwijaya dengan bidang riset Image Processing dan Computer Vision dan sebagai salah seorang pendiri **Sriwijaya Imaging Science Research Group**

PemrosesanCitraBerwarnaAplikasidenganJava.pdf

ORIGINALITY REPORT

12%

SIMILARITY INDEX

PRIMARY SOURCES

1	margi.staff.gunadarma.ac.id Internet	363 words — 3%
2	www.inf.ufrgs.br Internet	156 words — 1%
3	www.megenep.com Internet	83 words — 1%
4	meandmyheart.files.wordpress.com Internet	65 words — 1%
5	Steeb, . "Fuzzy Sets and Fuzzy Logic", The Nonlinear Workbook, 2015. Crossref	60 words — 1%
6	faizabdullz.blogspot.com Internet	45 words — < 1%
7	fst.unair.ac.id Internet	41 words — < 1%
8	kenalanwar.blogspot.com Internet	39 words — < 1%
9	stopumts.nl Internet	34 words — < 1%
10	EunJin Kim, Hyeon-Jeong Suk. "Key Color Generation for Affective Multimedia Production", Proceedings of the 2016 ACM on Multimedia Conference - MM '16, 2016 Crossref	28 words — < 1%
11	etheses.uin-malang.ac.id Internet	27 words — < 1%
12	eprints.uwe.ac.uk Internet	26 words — < 1%
13	Sajal Pahariya, Sandeep Tiwari. "Image segmentation using snake model with nosie adaptive fuzzy switching median filter and MSRM method", 2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), 2015	25 words — < 1%

Crossref

14	Lebedeva, A.V.. "On algorithmic methods of analysis of two-colorings of hypergraphs.", Journal of Mathematical Sciences, Feb 9 2016 Issue Publications	24 words — < 1%
15	pdfs.semanticscholar.org Internet	23 words — < 1%
16	Mahmoodi, Mohammad Reza, and Sayed Masoud Sayedi. "A face detector based on color and texture", 2014 6th International Conference on Information Technology and Electrical Engineering (ICITEE), 2014. Crossref	22 words — < 1%
17	oru.diva-portal.org Internet	22 words — < 1%
18	event.cs.usm.my Internet	21 words — < 1%
19	Wang, Gaihua, Yang Liu, and Tongzhou Zhao. "A quaternion-based switching filter for colour image denoising", Signal Processing, 2014. Crossref	21 words — < 1%
20	www.javadocexamples.com Internet	19 words — < 1%
21	ejournal.unikama.ac.id Internet	16 words — < 1%
22	www.mdpi.com Internet	15 words — < 1%
23	skripsi-skripsiun.blogspot.com Internet	13 words — < 1%
24	id.scribd.com Internet	12 words — < 1%
25	arizona.pure.elsevier.com Internet	12 words — < 1%
26	dblp1.uni-trier.de Internet	11 words — < 1%
27	research-information.bristol.ac.uk Internet	10 words — < 1%
28	scholars.cityu.edu.hk Internet	10 words — < 1%

"Table of contents", 2016 Sixth International Conference on

29	Information Science and Technology (ICIST), 2016. Crossref	10 words — < 1%
30	Learn Java for Android Development, 2013. Crossref	10 words — < 1%
31	www.akyokus.com Internet	10 words — < 1%
32	www.readbag.com Internet	10 words — < 1%
33	edocs.ilkom.unsri.ac.id Internet	9 words — < 1%
34	dblp.uni-trier.de Internet	9 words — < 1%
35	Lezoray, O.. "Color image segmentation using morphological clustering and fusion with automatic scale selection", Pattern Recognition Letters, 20090301 Crossref	9 words — < 1%
36	Shin-Yu Chen. "Cross-view object identification using principal color transformation", 2010 International Conference on Machine Learning and Cybernetics, 07/2010 Crossref	9 words — < 1%
37	es.scribd.com Internet	9 words — < 1%
38	repository.its.ac.id Internet	9 words — < 1%
39	mitlab.csie.ndhu.edu.tw Internet	8 words — < 1%
40	trevelista.blogspot.com Internet	8 words — < 1%
41	budiyono.staff.fkip.uns.ac.id Internet	8 words — < 1%
42	www.dikti.go.id Internet	8 words — < 1%
43	www.docstoc.com Internet	8 words — < 1%
44	marcelenjose.nl Internet	8 words — < 1%

45	www.neliti.com Internet	8 words — < 1%
46	grietzzc-unity.blogspot.com Internet	8 words — < 1%
47	journal.portalgaruda.org Internet	8 words — < 1%
48	Kushwaha, Ajay Kumar, and Sajal K. Paul. "Current Mode Universal Filter Using Single Current Controlled Differential Difference Current Conveyor Transconductance Amplifier", Circuits and Systems, 2015. Crossref	6 words — < 1%

EXCLUDE QUOTES	OFF	EXCLUDE MATCHES	OFF
EXCLUDE BIBLIOGRAPHY	ON		