**Laporan Praktikum Pengolahan Citra**

****

**Mata Kuliah : [TIF19] Pengolahan Citra**

**Dosen : Teady Matius Surya Mulyana, S.Kom., M.Kom**

**Dibuat Oleh:**

**Nama : Antoni K. Wijaya**

**NIM : 32130026**

Daftar Isi

[Pertemuan I dan II: PixelGrabbing dan Pemisahan Warna 1](#_Toc438317692)

[Pertemuan III: Bitmap 7](#_Toc438317693)

[Pertemuan IV: Brightness, Citra Negatif dan Citra Biner 11](#_Toc438317694)

[Pertemuan V: Kontras dan Histogram 17](#_Toc438317695)

[Pertemuan VI: Filtering Spasial Non-Linier 22](#_Toc438317696)

[Pertemuan VII: Filtering Spasial Linier – Konvolusi 29](#_Toc438317697)

[Pertemuan VIII: Deteksi Tepi 33](#_Toc438317698)

[Pertemuan IX: Kontur 38](#_Toc438317699)

[Pertemuan X: Kontur – Kode Rantai 45](#_Toc438317700)

[Pertemuan XI: Transformasi – Flip, Translasi dan Rotasi 52](#_Toc438317701)

[Pertemuan XII: Transformasi – Scaling 60](#_Toc438317702)

[Pertemuan XIII: Steganografi 68](#_Toc438317703)

LAPORAN PRAKTIKUM PENGOLAHAN CITRA

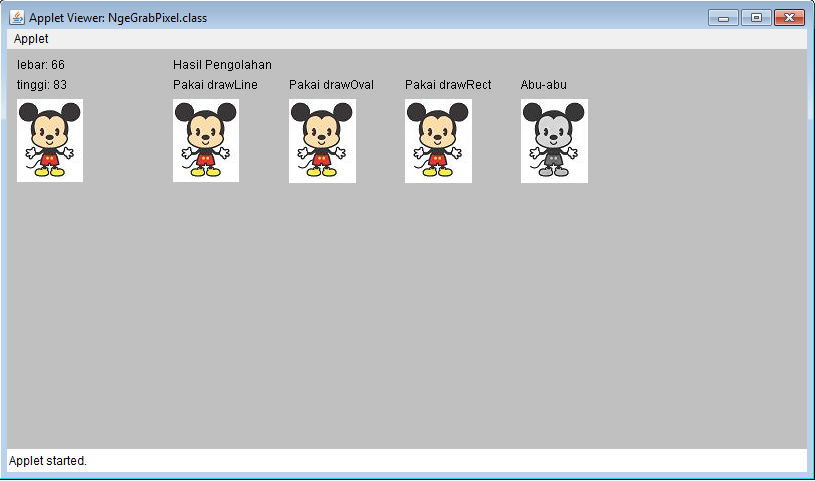
Pertemuan I dan II: PixelGrabbing dan Pemisahan Warna

Pada pertemuan pertama dan kedua di mata kuliah ini, kita mempelajari mengenai teknik-teknik dasar yang akan digunakan dalam pengolahan citra, yakni PixelGrabbing dan pemisahan warna pada tiap titik (pixel) yang ada di dalam gambar tertentu. Sebelum suatu gambar akan diolah menggunakan komputer, pertama-tama kita harus mendapatkan dan menyimpan atribut-atribut dasar pembentuk gambar tersebut. Pada dasarnya, suatu gambar tersusun dari banyak titik (pixel) yang memiliki warna tertentu. Untuk mendapatkan informasi mengenai warna-warna setiap titik, maka gambar tersebut pertama-tama perlu untuk dipecah menjadi titik-titik pembentuknya. Di dalam bahasa pemrograman Java, method yang digunakan untuk memecah gambar menjadi titik-titik ini disebut dengan PixelGrabber. Sesuai namanya, PixelGrabber akan mengambil (grab) informasi titik (pixel) yang ada pada sebuah gambar.

Sebuah warna pada intinya dapat merupakan percampuran dari beberapa warna dasar dengan intensitas yang berbeda-beda. Kombinasi dari warna dasar dan intensitasnya tersebut digunakan untuk membuat suatu model warna. Terdapat berbagai jenis model warna, dimana model warna yang paling umum digunakan contohnya adalah RGB (Red-Green-Blue) dan CMYK (Cyan Magenta Yellow Key). Pada praktik pengolahan citra yang akan kita pelajari, model warna yang akan kita gunakan adalah RGB. Dengan demikian, untuk setiap titik berwarna, maka kita dapat menarik informasi mengenai intensitas warna-warna dasar (merah, hijau dan biru) pembentuk titik berwarna tersebut. Istilah pemisahan warna mengacu pada tindakan untuk mendapatkan informasi warna dasar tersebut dari sebuah titik berwarna.

Pada praktik kali ini, kita akan menggunakan applet Java. Applet Java merupakan sebuah program kecil berbasis Java yang dapat dijalankan pada halaman web (html) yang bersifat statis. Penulisan kode program pada applet Java dapat menggunakan IDE berbasis Java apapun. Namun pada praktik kali ini, kita tidak akan menggunakan IDE berbasis Java tetapi menggunakan sebuah text editor, yaitu Notepad++. Kode program disimpan dengan menggunakan ekstensi .java dan, sama dengan program Java lain, disimpan dengan filename yang sama dengan nama class yang ada dalam program Java tersebut.

Hasil praktik pada pertemuan pertama adalah sebagai berikut:



**Gambar 1.1. Hasil Latihan PixelGrabbing dan Pemisahan Warna 1**

Pada praktik yang pertama ini, kita akan mencoba untuk membuat komputer membaca suatu gambar (image), memecah gambar tersebut dengan menggunakan method PixelGrabber, dan memisahkan warna setiap titik sambil menggambar ulang gambar tersebut. Proses penggambaran ulang disini dilakukan titik demi titik, bukannya secara langsung. Karena applet Java tidak mengizinkan penggambaran titik (pixel) secara langsung, maka untuk mengakalinya digunakan berbagai method lain yang sebenarnya fungsinya bukan untuk menggambar titik. Method tersebut adalah drawLine (untuk menggambar garis), drawOval (untuk menggambar oval/lingkaran), dan drawRect (untuk menggambar segi empat). Method tersebut akan diatur parameternya sehingga gambar hasil cetakan method tersebut menjadi sebuah titik.

Berikut kode program (source code applet Java) praktik tersebut:

|  |
| --- |
| import java.applet.Applet;  import java.awt.Graphics;  import java.awt.Color;  import java.awt.Image;  import java.awt.image.PixelGrabber;  public class NgeGrabPixel extends Applet  {  Image img;  int lebar, tinggi;  int warna, red, green, blue, alpha, abuabu, tempwarna;  String tampil;    public void init()  {  img=this.getImage(this.getDocumentBase(), "PB1MAT+mickey0.jpg");  lebar = img.getWidth(this);  }    public void paint(Graphics g)  {  //beri warna pada panel  g.setColor(Color.lightGray);  g.fillRect(0,0,2800,800);    //ambil lebar dan tinggi citra  lebar = img.getWidth(this);  tinggi = img.getHeight(this);    //Tampilkan informasi lebar dan tinggi citra  g.setColor(Color.black);  tampil = String.valueOf(lebar);  g.drawString("lebar: "+tampil,10,20);  tampil = String.valueOf(tinggi);  g.drawString("tinggi: "+tampil,10,40);    g.drawImage(img,10,50,this); //Tampilkan citra dengan drawImage    int[] pixels = new int[lebar\*tinggi]; //tentukan panjang array pixels    PixelGrabber pg = new PixelGrabber(img,0,0,lebar,tinggi,pixels,0,lebar);  try  {  pg.grabPixels(); //grab piksel citra ke object pg.  }  catch (InterruptedException ie)  {  System.out.println("Terjadi kesalahan saat mengambil data pixels");  ie.printStackTrace();  return;  }    g.drawString("Hasil Pengolahan",lebar+100,20);  g.drawString("Pakai drawLine",lebar+100,40);  g.drawString("Pakai drawOval",lebar+100+lebar+50,40);  g.drawString("Pakai drawRect",lebar+100+lebar+50+lebar+50,40);  g.drawString("Abu-abu",lebar+100+lebar+50+lebar+50+lebar+50,40);    g.setColor(Color.black);  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  warna = pixels[j\*lebar+i];  alpha = (warna >> 24) & 0xff;  red = (warna >> 16) & 0xff;  green = (warna >> 8) & 0xff;  blue = (warna) & 0xff;    g.setColor(new Color(red,green,blue));  g.drawLine(i+lebar+100,j+50,i+lebar+100,j+50);  g.drawOval(i+lebar+100+lebar+50,j+50,1,1);  g.drawRect(i+lebar+100+lebar+50+lebar+50,j+50,1,1);  abuabu = (red+green+blue) / 3;  g.setColor(new Color(abuabu,abuabu,abuabu));  g.drawOval(i+lebar+100+lebar+50+lebar+50+lebar+50,j+50,1,1);  }  }  }    public void repaint(Graphics g)  {  //kosongkan saja  }  } |

* Untuk membuat program ini, beberapa header (import) library yang akan kita gunakan adalah:
  + java.applet.Applet; = library dasar applet Java
  + java.awt.Graphics; = untuk menangani penggambaran menggunakan komputer
  + java.awt.Color; = untuk pengaturan warna
  + java.awt.Image; = untuk pembuatan objek Image, pembacaan file gambar
  + java.awt.image.PixelGrabber; = untuk penggunaan method PixelGrabber
* Pertama-tama, deklarasikan public class yang mewarisi (inherit) kelas Applet. Pada kode diatas, kelas ini bernama NgeGrabPixel (public class NgeGrabPixel extends Applet)
* Selanjutnya, siapkan variable-variable global (atribut kelas) yang diperlukan:
  + Objek “Image” yang akan digunakan untuk menampung file gambar (Image img;)
  + Variable lebar dan tinggi bertipe integer yang akan digunakan untuk menampung informasi lebar dan tinggi gambar (int lebar, tinggi;)
  + Sejumlah variable bertipe integer untuk menampung warna (int warna, red, green, blue, alpha, abuabu, tempwarna;). Diantara variable tersebut, variable yang akan digunakan pada pertemuan ini hanyalah variable “warna”, “red”, “green”, “blue” dan “abu-abu”
  + Sebuah variable bertipe String yang akan digunakan untuk menampilkan text (String tampil;)
* Setelah deklarasi atribut, buatlah method untuk inisialisasi class Java (public void init()). Isi method tersebut adalah:
  + Ambil file gambar dan masukkan ke dalam objek Image yang telah kita deklarasikan sebelumnya dengan menggunakan method getImage() -> (img=this.getImage(this.getDocumentBase(), "PB1MAT+mickey0.jpg");)

Method getImage() menerima dua buah parameter: parameter yang pertama digunakan untuk merujuk alamat lokasi file gambar di komputer, perintah “this.getDocumentBase()” diatas digunakan untuk langsung mengambil alamat dimana file class Java program ini disimpan; parameter yang kedua adalah nama file gambar yang akan dimuat dalam String.

* + Selanjutnya, coba ambil lebar gambar tersebut dan simpan dalam variable (lebar = img.getWidth(this);)
* Setelah inisialisasi, barulah kita membuat method untuk menangani penggambaran ulang citra. Method yang menangani penggambaran dalam applet Java adalah method paint(). Method ini sendiri memiliki sebuah parameter grafis yang mendefinisikan kanvas penggambaran (public void paint(Graphics g))
* Di dalam method paint(), lakukan hal-hal sebagai berikut:
  + Beri warna background pada kanvas, dengan pertama-tama mengubah warna “kuas” penggambaran menjadi abu-abu (g.setColor(Color.lightGray);) dan kemudian menggambar persegi panjang besar dengan menggunakan kuas abu-abu tersebut (g.fillRect(0,0,2800,800);)
  + Berikutnya, ambil lebar dan tinggi citra (lebar = img.getWidth(this); dan tinggi = img.getHeight(this);)
  + Tampilkan informasi lebar dan tinggi citra tersebut pada kanvas:
    - Ubah warna kuas menjadi hitam (g.setColor(Color.black);)
    - Isi String tampil dengan lebar citra (tampil = String.valueOf(lebar);)
    - Tulis lebar citra dengan method drawString (g.drawString("lebar: "+tampil,10,20);)
    - Isi String tampil dengan tinggi citra (tampil = String.valueOf(tinggi);)
    - Tulis tinggi citra dengan method drawString (g.drawString("tinggi: "+tampil,10,40);)
    - Ctt: method drawstring() menerima tiga buah parameter: parameter pertama adalah String yang akan ditampilkan, parameter kedua dan ketiga adalah posisi penggambaran String dalam koordinat (x dan y)
  + Kemudian, cobalah untuk menggambar citra tersebut secara langsung (g.drawImage(img,10,50,this);)
  + Setelah semua hal diatas, barulah proses pemecahan gambar dimulai. Mula-mula deklarasikan array satu dimensi bertipe integer (int[] pixels = new int[lebar\*tinggi];). Array ini akan digunakan untuk menampung semua titik-titik yang ada pada gambar, oleh karena itu jumlah elemen pada array ini harus sama dengan jumlah titik pada gambar tersebut. Jumlah titik pada sebuah gambar adalah lebar x tinggi pixel pada gambar tersebut
  + Selanjutnya, deklarasikan objek PixelGrabber (PixelGrabber pg = new PixelGrabber(img,0,0,lebar,tinggi,pixels,0,lebar);). Objek PixelGrabber merupakan objek yang digunakan untuk memecah gambar menjadi titik-titik pembentuknya. Objek ini menerima 8 buah parameter:
    - Parameter pertama adalah objek Image yang menampung citra yang akan dipecah (img)
    - Parameter kedua dan ketiga mendefinisikan koordinat awal pengambilan titik pada gambar (0,0)
    - Parameter keempat dan kelima mendefinisikan koordinat akhir pengambilan titik (lebar,tinggi)
    - Parameter keenam merupakan array satu dimensi penampung informasi titik-titik hasil pemecahan (pixels)
    - Parameter ketujuh dan kedelapan mendefinisikan titik-titik mana yang akan diambil per baris titik pada gambar, parameter ini juga mendefinisikan jumlah titik yang akan diambil per baris titik pada gambar (0,lebar) -> artinya, ambil titik dari 0 (paling kiri) hingga lebar gambar untuk tiap baris
  + Setelah deklarasi objek PixelGrabber, pecahkan gambar tersebut dengan method grabPixels() (pg.grabPixels();). Bungkus instruksinya dalam try-catch agar apabila terjadi kesalahan (exception) pengambilan titik, maka program dapat menanggulanginya dengan menampilkan pesan
  + Berikutnya, tampilkan sejumlah string sebagai header (judul) gambar yang akan kita buat dan posisikan string-string tersebut (g.drawString("Hasil Pengolahan",lebar+100,20);hingga g.drawString("Abu-abu",lebar+100+lebar+50+lebar+50+lebar+50,40);)
  + Set warna kuas menjadi hitam (g.setColor(Color.black);)
  + Buat perulangan, dari posisi atas-kiri hingga bawah-kanan citra (for(int j=0;j<tinggi;j++) dan for(int i=0;i<lebar;i++)) , lakukan:
    - Simpan informasi warna titik tersebut (warna = pixels[j\*lebar+i];)
    - Sebuah warna RGB di komputer disimpan dengan ukuran 4 byte. 4 byte tersebut dibagi-bagi untuk menampung informasi alpha (informasi ketransparanan warna), red (intensitas warna merah), green (intensitas warna hijau), dan blue (intensitas warna biru) yang ada pada warna RGB tersebut. Secara urutan bit, informasi tersebut tersusun berurutan, dengan informasi byte alpha berada pada MSB dan informasi byte blue berada pada LSB
    - Berdasarkan struktur tersebut, maka untuk memecah / mendapatkan informasi mengenai warna dasar (RGB dan alpha) dari sebuah warna dapat dilakukan dengan cara:
      * Untuk Alpha, geser bit warna titik sebesar 24 bit ke kanan dan AND-kan dengan 0xff (atau 8-bit 11111111) (alpha = (warna >> 24) & 0xff;)
      * Untuk Red, geser bit warna titik sebesar 16 bit ke kanan dan AND-kan dengan 0xff (red = (warna >> 16) & 0xff;)
      * Untuk Green, geser bit warna titik sebesar 8 bit ke kanan dan AND-kan dengan 0xff (green = (warna >> 8) & 0xff;)
      * Untuk Blue, langsung AND-kan warna dengan 0xff (blue = (warna) & 0xff;)
    - Set warna kuas dengan mencampurkan ketiga warna dasar RGB (g.setColor(new Color(red,green,blue));), hasilnya adalah kuas dengan warna yang sama dengan warna titik
    - Selanjutnya, gambar titik tersebut:
      * Dengan method drawLine() (g.drawLine(i+lebar+100,j+50,i+lebar+100,j+50);) Ctt: drawLine() menerima 4 parameter, titik awal (x1 dan y1) dan titik akhir (x2 dan y2) garis. Karena gambar yang akan kita buat adalah titik, maka x1 = x2 dan y1 = y2
      * Dengan method drawOval() (g.drawOval(i+lebar+100+lebar+50,j+50,1,1);) Ctt: drawOval() menerima 4 parameter, titik awal (x1 dan y1), ukuran horizontal dan ukuran vertikal (dalam pixel). Karena gambar yang akan kita buat adalah titik, ukuran horizontal dan vertikal keduanya adalah 1
      * Dengan method drawRect() (g.drawRect(i+lebar+100+lebar+50+lebar+50, j+50,1,1);) Ctt: method drawRect() menerima 4 parameter yang berfungsi sama seperti parameter drawOval()
    - Selanjutnya, untuk gambar terakhir kita akan mencoba untuk mengubah warna gambar menjadi abu-abu (grayscale) sebelum digambar ulang. Untuk itu, kita perlu untuk mencari warna grayscale gambar tersebut dengan menggunakan rumus (abuabu = (red+green+blue) / 3;)
    - Berikutnya, set warna kuas dengan memasukkan warna tersebut ke tiga warna dasar (g.setColor(new Color(abuabu,abuabu,abuabu));)
    - Terakhir, gambar titik grayscale tersebut di posisinya (g.drawOval(i+lebar+100+lebar+50+lebar+50+lebar+50,j+50,1,1);)

Berbeda dengan IDE khusus Java, Notepad++ tidak memiliki compiler Java sendiri. Untuk itu, file source code (.java) hasil penyimpanan perlu untuk di-compile secara manual oleh mesin Java. Proses compile dipanggil dengan menggunakan command prompt, dengan tahap-tahap sebagai berikut:

* Cek apakah command prompt sudah secara otomatis membaca java compiler dengan cara memasukkan perintah “javac”, Jika javac tidak terbaca oleh command prompt ('javac' is not recognized as an internal or external command operable program or batch file.), maka perlu untuk dilakukan pendeteksian secara manual dengan instruksi sebagai berikut:
  + set path=%path%;”alamat-direktori-java”
  + Isi alamat-direktori-java dengan alamat tempat java compiler, contohnya: *"c:\Program Files\Java\jdk1.7.0\_51\bin"*
* Jika javac sudah terbaca, maka tahap selanjutnya adalah untuk melakukan compile file source code yang telah kita buat. Caranya adalah:
  + Lakukan change disk (cd) ke direktori tempat dimana file source code berada. Contohnya bila file berada di desktop adalah: cd c:\Users\nama-user\Desktop\LAB02
  + Setelah command prompt berada di direktori yang kita inginkan, jalankan java compiler dengan instruksi: javac nama-file.java
  + Jika sukses, maka akan terbentuk sebuah file .class di folder yang sama dengan file source code

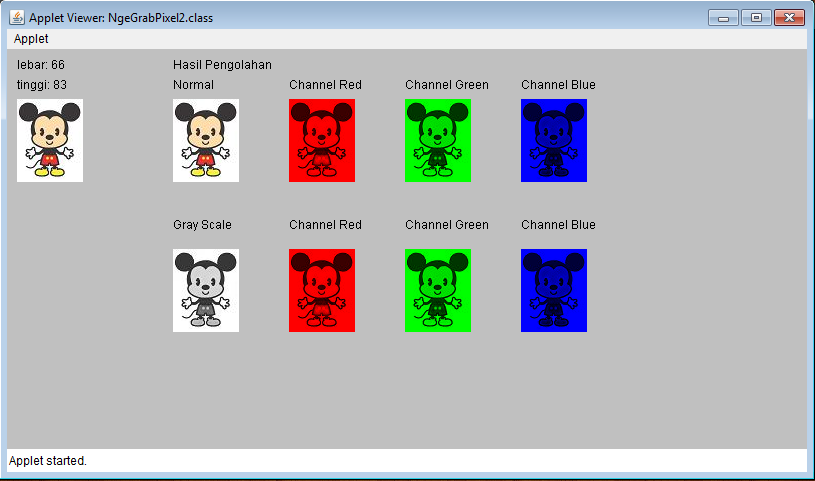
Untuk menjalankan applet Java hasil compile, maka kita perlu membuat file html sederhana untuk memanggil applet tersebut. Berikut adalah contoh isi dari dokumen html tersebut:

|  |
| --- |
| <html>  <head>  </head>  <body>  <applet code=nama-file.class width=800 height=400>  </applet>  </body>  </html> |

* Tag <applet> berfungsi untuk menjalankan java applet. Atribut code berfungsi untuk membaca file class java, atribut width dan height berfungsi untuk mengatur besar tampilan applet

Simpan dokumen html diatas pada folder yang sama dengan folder tempat file .class hasil compile. Setelah disimpan, applet dapat dijalankan menggunakan perintah appletviewer nama-file.html pada command prompt.

Setelah praktik untuk pertemuan pertama, selanjutnya adalah latihan praktik untuk pertemuan kedua. Berikut adalah hasil praktiknya:



**Gambar 2.1. Hasil Praktik Pertemuan Kedua**

Praktik pertemuan kedua ini tidak jauh berbeda dengan praktik pertemuan pertama. Praktik kali ini menekankan pada penggunaan warna dasar (Channel warna) secara tersendiri tanpa mencampurkannya. Citra yang berasal hanya dari unsur warna merahnya saja disebut Channel Red, yang berasal dari warna hijau saja disebut Channel Green, dan yang berasal dari warna biru disebut Channel Blue. Selain penggunaan channel warna tersebut, di praktik ini kita juga belajar untuk memposisikan citra agar terletak di tempat yang kita inginkan.

Oleh karena praktik kali ini mirip dengan praktik yang pertama, maka sebagian kode programnya sama dengan praktik yang pertama. Perbedaannya hanya ada di dalam method paint(), tepatnya setelah try-catch PixelGrabber.

Berikut kode program bagian yang berbeda tersebut:

|  |
| --- |
| -------------------------  g.drawString("Hasil Pengolahan",lebar+100,20);  g.drawString("Normal",lebar+100,40);  g.drawString("Channel Red",lebar+100+lebar+50,40);  g.drawString("Channel Green",lebar+100+lebar+50+lebar+50,40);  g.drawString("Channel Blue",lebar+100+lebar+50+lebar+50+lebar+50,40);  g.drawString("Gray Scale",lebar+100,180);  g.drawString("Channel Red",lebar+100+lebar+50,180);  g.drawString("Channel Green",lebar+100+lebar+50+lebar+50,180);  g.drawString("Channel Blue",lebar+100+lebar+50+lebar+50+lebar+50,180);    g.setColor(Color.black);  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  warna = pixels[j\*lebar+i];  alpha = (warna >> 24) & 0xff;  red = (warna >> 16) & 0xff;  green = (warna >> 8) & 0xff;  blue = (warna) & 0xff;    g.setColor(new Color(red,green,blue));  g.drawLine(i+lebar+100,j+50,i+lebar+100,j+50);  g.setColor(new Color(red,0,0));  g.drawLine(i+lebar+100+lebar+50,j+50,i+lebar+100+lebar+50,j+50);  g.setColor(new Color(0,green,0));  g.drawLine(i+lebar+100+lebar+50+lebar+50,j+50,i+lebar+100+lebar+50+lebar+50,j+50);    g.setColor(new Color(0,0,blue));  g.drawLine(i+lebar+100+lebar+50+lebar+50+lebar+50,j+50,i+lebar+100+lebar+50+lebar+50+lebar+50,j+50);    abuabu = (red+green+blue)/3;  g.setColor(new Color(abuabu,abuabu,abuabu));  g.drawLine(i+lebar+100,j+200,i+lebar+100,j+200);  g.setColor(new Color(red,0,0));  g.drawLine(i+lebar+100+lebar+50,j+200,i+lebar+100+lebar+50,j+200);  g.setColor(new Color(0,green,0));  g.drawLine(i+lebar+100+lebar+50+lebar+50,j+200,i+lebar+100+lebar+50+lebar+50,j+200);    g.setColor(new Color(0,0,blue)); g.drawLine(i+lebar+100+lebar+50+lebar+50+lebar+50,j+200,i+lebar+100+lebar+50+lebar+50+lebar+50,j+200)  ------------------------- |

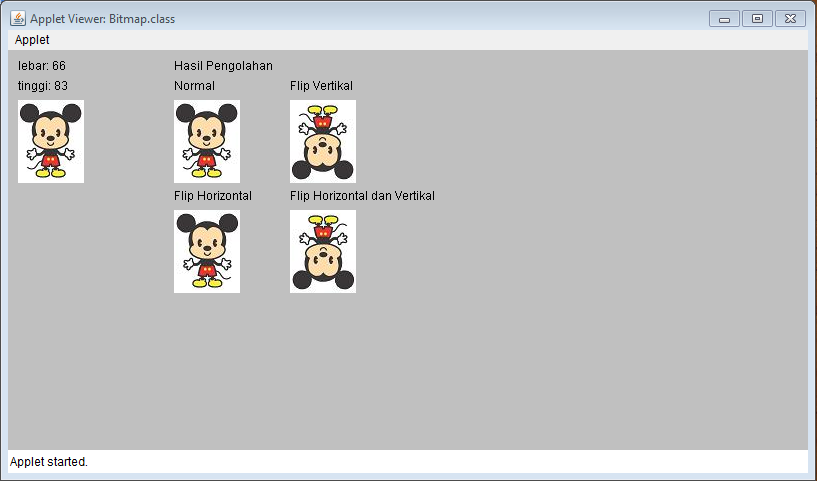
* Buat dan susun string-string judul sesuai dengan gambar 2.1 diatas (g.drawString("Hasil Pengolahan", lebar+100,20); hingga g.drawString("Channel Blue",lebar+100+lebar+50+lebar+50+lebar+50,180);). Posisi judul ini akan menentukan posisi citra juga, jadi pastikan untuk mengatur posisi judul tersebut agar citra dapat disusun dibawahnya dengan rapi
* Set warna kuas menjadi hitam, sebagai warna awal (g.setColor(Color.black);)
* Dari atas-kiri hingga bawah-kanan citra (for(int j=0;j<tinggi;j++) dan for(int i=0;i<lebar;i++)), lakukan:
  + Ambil warna titik (warna = pixels[j\*lebar+i];)
  + Pisahkan warna titik (alpha = (warna >> 24) & 0xff; hingga blue = (warna) & 0xff;)
  + Untuk gambar normal, gunakan warna kuas sama dengan warna titik (g.setColor(new Color(red,green,blue));)
  + Gambar titiknya (g.drawLine(i+lebar+100,j+50,i+lebar+100,j+50);)
  + Untuk Channel Red, campurkan warna merahnya saja pada kuas (g.setColor(new Color(red,0,0));)
  + Gambar titiknya (g.drawLine(i+lebar+100+lebar+50,j+50,i+lebar+100+lebar+50,j+50);)
  + Untuk Channel Green, campurkan warna hijaunya saja (g.setColor(new Color(0,green,0));)
  + Gambar titiknya (g.drawLine(i+lebar+100+lebar+50+lebar+50,j+50, i+lebar+100+lebar+50+lebar+50,j+50);)
  + Untuk Channel Blue, campurkan warna birunya saja (g.setColor(new Color(0,0,blue));)
  + Gambar titiknya (g.drawLine(i+lebar+100+lebar+50+lebar+50+lebar+50,j+50, i+lebar+100+lebar+50+lebar+50+lebar+50,j+50);)
  + Berikutnya, cari warna grayscale titik (abuabu = (red+green+blue)/3;)
  + Set warna grayscale pada kuas (g.setColor(new Color(abuabu,abuabu,abuabu));)
  + Gambar titiknya di bawah gambar normal (g.drawLine(i+lebar+100,j+200,i+lebar+100,j+200);)
  + Selanjutnya, gambar titik Channel RGB sama seperti sebelumnya, namun pada posisi dibawah citra RGB yang sebelumnya

LAPORAN PRAKTIKUM PENGOLAHAN CITRA

Pertemuan III: Bitmap

Di pertemuan sebelumnya, kita mempraktikkan mengenai penggunaan informasi channel warna titik secara langsung untuk membentuk ulang gambar. Berbeda dengan sebelumnya, di pertemuan kali ini kita akan belajar untuk menyimpan informasi channel-channel warna tersebut. Channel warna titik pembentuk gambar tersebut akan kita simpan di dalam sebuah bitmap, yang merupakan array dua dimensi. Setelah disimpan, kita akan mencoba untuk membentuk ulang gambar dan mem-flip gambar melalui pengaturan urutan penggambaran titik.

Berikut hasil dari praktik pada pertemuan ini:



**Gambar 3.1. Hasil Praktik Pertemuan Ketiga**

Seperti yang terlihat pada gambar 3.1, hasil praktik pertemuan kali ini tidak jauh berbeda dengan praktik pertemuan sebelumnya. Secara singkat, operasi yang dilakukan pada program kali ini adalah: memecah gambar/mengambil informasi titik, menyimpan informasi channel warna dalam bitmap dan membentuk ulang citra. Untuk pembentukan citra flip, perlu dilakukan pengaturan urutan penggambaran titik sehingga titik kiri citra digambar pada posisi kanan, titik atas citra menjadi digambar pada posisi bawah dan sebaliknya.

Kode program yang digunakan pada pertemuan kali ini juga sebagian menggunakan kode program pada pertemuan sebelumnya. Bagian yang berbeda pada pertemuan kali ini terletak di dalam method paint(), setelah try-catch PixelGrabber.

Berikut bagian yang berbeda tersebut:

|  |
| --- |
| -------------------------  g.drawString("Hasil Pengolahan",lebar+100,20);  g.drawString("Normal",lebar+100,40);  g.drawString("Flip Vertikal",lebar+100+lebar+50,40);  g.drawString("Flip Horizontal",lebar+100,150);  g.drawString("Flip Horizontal dan Vertikal",lebar+100+lebar+50,150);    g.setColor(Color.black);  int[][] red = new int[tinggi][lebar];  int[][] green = new int[tinggi][lebar];  int[][] blue = new int[tinggi][lebar];  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  warna = pixels[j\*lebar+i];  red[j][i] = (warna >> 16) & 0xff;  green[j][i] = (warna >> 8) & 0xff;  blue[j][i] = (warna) & 0xff;  }  }  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  g.setColor(new Color(red[j][i],green[j][i],blue[j][i]));  g.drawLine(i+lebar+100,j+50,i+lebar+100,j+50);    g.setColor(new Color(red[tinggi-1-j][i],green[tinggi-1-j][i],blue[tinggi-1-j][i]));  g.drawLine(i+lebar+100+lebar+50,j+50,i+lebar+100+lebar+50,j+50);    g.setColor(new Color(red[j][lebar-1-i],green[j][lebar-1-i],blue[j][lebar-1-i]));  g.drawLine(i+lebar+100,j+160,i+lebar+100,j+160);    g.setColor(new Color(red[tinggi-1-j][lebar-1-i],green[tinggi-1-j][lebar-1-i],blue[tinggi-1-j][lebar-1-i]));  g.drawLine(i+lebar+100+lebar+50,j+160,i+lebar+100+lebar+50,j+160);  }  }  ------------------------- |

* Setelah PixelGrabber, buat dan atur posisi judul untuk gambar-gambar yang akan kita buat nanti (g.drawString("Hasil Pengolahan",lebar+100,20); hingga g.drawString("Flip Horizontal dan Vertikal",lebar+100+lebar+50,150);)
* Atur warna kuas menjadi default/hitam (g.setColor(Color.black);)
* Buatlah bitmap, array dua dimensi untuk menampung channel warna merah. Ukuran bitmap sama dengan ukuran gambar (int[][] red = new int[tinggi][lebar];)
* Buatlah bitmap, array dua dimensi untuk menampung channel warna hijau (int[][] green = new int[tinggi][lebar];)
* Buatlah bitmap, array dua dimensi untuk menampung channel warna biru (int[][] blue = new int[tinggi][lebar];)
* Selanjutnya, lakukan pengisian bitmap. Dari atas-kiri hingga bawah-kanan citra (for(int j=0;j<tinggi;j++) dan for(int i=0;i<lebar;i++)) lakukan:
  + Ambil informasi warna titik hasil pemecahan (warna = pixels[j\*lebar+i];)
  + Saring channel warna merahnya dan simpan di bitmap (red[j][i] = (warna >> 16) & 0xff;)
  + Saring channel warna hijaunya dan simpan di bitmap (green[j][i] = (warna >> 8) & 0xff;)
  + Saring channel warna birunya dan simpan di bitmap (blue[j][i] = (warna) & 0xff;)
* Berikutnya, lakukan proses pembentukan ulang citra. Dari atas-kiri hingga bawah-kanan citra lakukan:
  + Untuk gambar normal, set warna kuas menjadi warna titik pada posisi normal (g.setColor(new Color(red[j][i],green[j][i],blue[j][i]));)
  + Gambar titik pada tempatnya (g.drawLine(i+lebar+100,j+50,i+lebar+100,j+50);)
  + Untuk flip vertikal, set warna titik yang berada di bagian bawah citra terlebih dahulu (g.setColor(new Color(red[tinggi-1-j][i],green[tinggi-1-j][i],blue[tinggi-1-j][i]));)
  + Gambar titik tersebut pada tempatnya (g.drawLine(i+lebar+100+lebar+50,j+50,i+lebar+100+lebar+50,j+50);)
  + Untuk flip horizontal, set warna titik yang berada di bagian kanan citra terlebih dahulu (g.setColor(new Color(red[j][lebar-1-i],green[j][lebar-1-i],blue[j][lebar-1-i]));)
  + Gambar titik tersebut pada tempatnya (g.drawLine(i+lebar+100,j+160,i+lebar+100,j+160);)
  + Untuk flip horizontal dan vertikal, set warna titik yang berada di kanan-bawah citra terlebih dahulu (g.setColor(new Color(red[tinggi-1-j][lebar-1-i],green[tinggi-1-j][lebar-1-i],blue[tinggi-1-j][lebar-1-i]));)
  + Gambar titik tersebut pada tempatnya (g.drawLine(i+lebar+100+lebar+50,j+160,i+lebar+100+lebar+50,j+160);)

Setelah selesai membuat kode program yang baru, simpanlah (pada contoh ini saya simpan dengan nama Bitmap.java), lakukan compile program melalui javac dan buat dokumen html untuk memanggilnya. Ini isi dokumen html yang saya gunakan:

|  |
| --- |
| <html>  <head>  </head>  <body>  <applet code=Bitmap.class width=800 height=400>  </applet>  </body>  <html> |

Setelah semuanya selesai, buka dokumen html melalui appletviewer dan lihat hasilnya.

LAPORAN PRAKTIKUM PENGOLAHAN CITRA

Pertemuan IV: Brightness, Citra Negatif dan Citra Biner

Pada pertemuan kali ini, kita mempraktikkan mengenai pengaturan kecerahan (brightness) citra, pengubahan citra warna dan grayscale menjadi citra negatif dan pengubahan citra warna dan grayscale menjadi citra biner. Kecerahan citra terkait dengan gelap-terangnya citra tersebut. Secara sederhana, semakin terang suatu citra, maka semakin dekat citra tersebut ke warna putih. Pada model warna RGB, warna putih didapat dengan nilai channel R, G, dan B maksimum, yaitu 255. Karena terangnya suatu warna berarti warna tersebut dekat dengan warna putih, maka untuk menerangkan suatu titik berwarna cukup dilakukan penambahan suatu nilai (konstanta/k) kecerahan terhadap channel warna/grayscale mula-mula. Rumusnya adalah sebagai berikut:

Ket: f0 = nilai channel grayscale atau R/G/B hasil brightness, fi = nilai channel mula-mula, k = nilai kecerahan

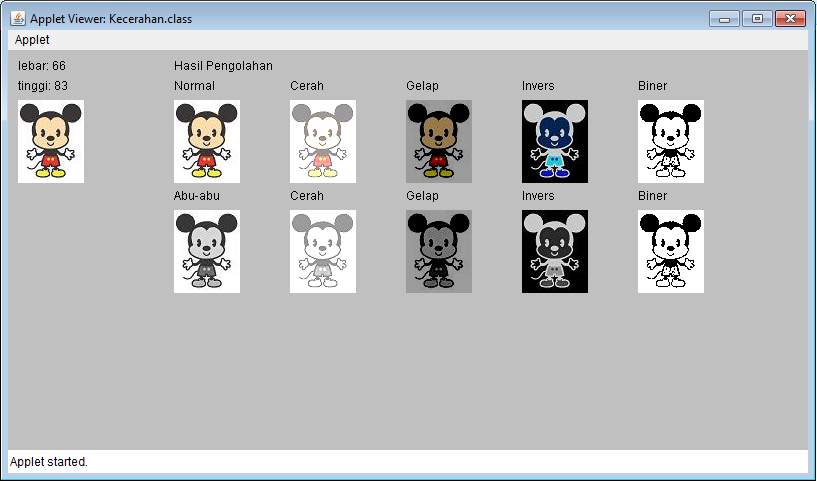
Untuk mencerahkan citra, buat nilai k menjadi positif, sedangkan untuk menggelapkan citra, buat nilai k menjadi negatif. Satu hal yang perlu diperhatikan adalah nilai channel hasil tidak boleh melebihi 255 (nilai maksimum) dan tidak boleh kurang dari 0 (nilai minimum).

Citra negatif atau citra invers dapat dikatakan sebagai kebalikan dari citra mula-mula. Secara sederhana, pembalikan warna berarti membalikkan warna dari warna putih menjadi hitam dan sebaliknya. Warna negatif citra didapatkan dari mengurangi nilai maksimum (255) dengan nilai warna yang dimiliki citra. Rumusnya adalah sebagai berikut:

Ket: f0 = nilai channel grayscale atau R/G/B negatif, fi = nilai channel mula-mula

Citra biner adalah citra dengan warna yang paling sederhana. Sederhana karena citra biner hanya memiliki murni dua warna, yaitu hitam dan putih. Kesemua titik pada citra biner memiliki nilai antara 1 (atau 255 pada citra 8 bit) atau 0. Pada pertemuan kali ini, kita akan mengkonversikan citra warna yang kita miliki menjadi citra biner dengan menggunakan prinsip threshold. Threshold adalah sebutan untuk sebuah nilai batas. Nilai batas tersebut akan menjadi acuan yang dibandingkan dengan nilai awal untuk mendapatkan nilai hasil konversi. Mudahnya, jika nilai awal melebihi atau sama dengan threshold, maka nilai awal tersebut akan dibulatkan menjadi nilai maksimum (1 atau 255) dan jika kurang dari threshold maka nilai tersebut akan dibulatkan menjadi minimum (0).

Berikut hasil dari praktik pada pertemuan kali ini:



**Gambar 4.1. Hasil Praktik Pertemuan 4**

Seperti yang terlihat pada gambar 4.1, pada praktik kali ini kita akan melakukan operasi pengaturan brightness, inversi (pembalikan) warna citra dan konversi citra ke citra biner. Ketiga hal tersebut akan kita terapkan ke versi normal citra dan ke versi grayscalenya. Konstanta brightness untuk mencerahkan citra (variablenya akan kita beri nama “kc”) akan menggunakan nilai 100, sedangkan konstanta untuk menggelapkannya (variablenya kita beri nama “kg”) akan menggunakan nilai -100. Adapun nilai threshold yang kita gunakan untuk konversi citra ke citra biner adalah 100.

Kode program yang digunakan pada pertemuan kali ini juga sebagian menggunakan kode program pada pertemuan sebelumnya. Bagian yang berbeda pada pertemuan kali ini terletak di dalam method paint(), setelah try-catch PixelGrabber.

Berikut kode program tersebut:

|  |
| --- |
| -------------------------  g.drawString("Hasil Pengolahan",lebar+100,20);  g.drawString("Normal",lebar+100,40);  g.drawString("Cerah",lebar+100+lebar+50,40);  g.drawString("Gelap",lebar+100+lebar+50+lebar+50,40);  g.drawString("Invers",lebar+100+lebar+50+lebar+50+lebar+50,40);  g.drawString("Biner",lebar+100+lebar+50+lebar+50+lebar+50+lebar+50,40);  g.drawString("Abu-abu",lebar+100,150);  g.drawString("Cerah",lebar+100+lebar+50,150);  g.drawString("Gelap",lebar+100+lebar+50+lebar+50,150);  g.drawString("Invers",lebar+100+lebar+50+lebar+50+lebar+50,150);  g.drawString("Biner",lebar+100+lebar+50+lebar+50+lebar+50+lebar+50,150);    g.setColor(Color.black);  int kc = 100; //Untuk brightness = cerah  int kg = -100; //Untuk brightness = gelap  int threshold = 100; //Untuk biner  int[][] red = new int[tinggi][lebar];  int[][] green = new int[tinggi][lebar];  int[][] blue = new int[tinggi][lebar];  int[][] abuabu = new int[tinggi][lebar];  int tempred,tempgreen,tempblue,tempabu;  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  warna = pixels[j\*lebar+i];  red[j][i] = (warna >> 16) & 0xff;  green[j][i] = (warna >> 8) & 0xff;  blue[j][i] = (warna) & 0xff;  abuabu[j][i] = (red[j][i] + green[j][i] + blue[j][i]) / 3;  }  }    for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  //Normal  g.setColor(new Color(red[j][i],green[j][i],blue[j][i]));  g.drawLine(i+lebar+100,j+50,i+lebar+100,j+50);    //Normal Cerah  if (red[j][i]+kc > 255)  {  tempred = 255;  }  else  {  tempred = red[j][i]+kc;  }    if (green[j][i]+kc > 255)  {  tempgreen = 255;  }  else  {  tempgreen = green[j][i]+kc;  }    if (blue[j][i]+kc > 255)  {  tempblue = 255;  }  else  {  tempblue = blue[j][i]+kc;  }  g.setColor(new Color(tempred,tempgreen,tempblue));  g.drawLine(i+lebar+100+lebar+50,j+50,i+lebar+100+lebar+50,j+50);    //Normal Gelap  if (red[j][i]+kg < 0)  {  tempred = 0;  }  else  {  tempred = red[j][i]+kg;  }    if (green[j][i]+kg < 0)  {  tempgreen = 0;  }  else  {  tempgreen = green[j][i]+kg;  }    if (blue[j][i]+kg < 0)  {  tempblue = 0;  }  else  {  tempblue = blue[j][i]+kg;  }  g.setColor(new Color(tempred,tempgreen,tempblue));  g.drawLine(i+lebar+100+lebar+50+lebar+50,j+50,i+lebar+100+lebar+50+lebar+50,j+50);    //Normal Negatif  g.setColor(new Color(255-red[j][i],255-green[j][i],255-blue[j][i]));  g.drawLine(i+lebar+100+lebar+50+lebar+50+lebar+50,j+50,i+lebar+100+lebar+50+lebar+50+lebar+50,j+50);    //Normal Biner  if((red[j][i]+green[j][i]+blue[j][i]) / 3 >= threshold)  {  tempabu = 255;  }  else  {  tempabu = 0;  }  g.setColor(new Color(tempabu,tempabu,tempabu)); g.drawLine(i+lebar+100+lebar+50+lebar+50+lebar+50+lebar+50,j+50,i+lebar+100+lebar+50+lebar+50+lebar+50+lebar+50,j+50);    //Abu-abu  g.setColor(new Color(abuabu[j][i],abuabu[j][i],abuabu[j][i]));  g.drawLine(i+lebar+100,j+160,i+lebar+100,j+160);    //Abu-abu Cerah  if (abuabu[j][i]+kc > 255)  {  tempabu = 255;  }  else  {  tempabu = abuabu[j][i]+kc;  }  g.setColor(new Color(tempabu,tempabu,tempabu));  g.drawLine(i+lebar+100+lebar+50,j+160,i+lebar+100+lebar+50,j+160);    //Abu-abu Gelap  if (abuabu[j][i]+kg < 0)  {  tempabu = 0;  }  else  {  tempabu = abuabu[j][i]+kg;  }  g.setColor(new Color(tempabu,tempabu,tempabu));  g.drawLine(i+lebar+100+lebar+50+lebar+50,j+160,i+lebar+100+lebar+50+lebar+50,j+160);    //Abu-abu Negatif  g.setColor(new Color(255-abuabu[j][i],255-abuabu[j][i],255-abuabu[j][i]));  g.drawLine(i+lebar+100+lebar+50+lebar+50+lebar+50,j+160,i+lebar+100+lebar+50+lebar+50+lebar+50,j+160);    //Abu-abu Biner  if(abuabu[j][i] >= threshold)  {  tempabu = 255;  }  else  {  tempabu = 0;  }  g.setColor(new Color(tempabu,tempabu,tempabu));  g.drawLine(i+lebar+100+lebar+50+lebar+50+lebar+50+lebar+50,j+160,i+lebar+100+lebar+50+lebar+50+lebar+50+lebar+50,j+160);  }  }  ------------------------- |

* Pertama-tama, tulis dan atur posisi header untuk setiap gambar (g.drawString("Hasil Pengolahan",lebar+100,20); hingga g.drawString("Biner",lebar+100+lebar+50+lebar+50+lebar+50 +lebar+50,150);)
* Set warna kuas menjadi default/hitam (g.setColor(Color.black);)
* Deklarasi variable yang dibutuhkan:
  + Konstanta brightness untuk mencerahkan citra (int kc = 100;)
  + Konstanta brightness untuk menggelapkan citra (int kg = -100;)
  + Threshold untuk citra biner (int threshold = 100;)
  + Array dua dimensi untuk bitmap (int[][] red hingga int[][] abuabu)
  + Variable sementara untuk menampung hasil perhitungan dan menggambarkan citra (int tempred,tempgreen,tempblue,tempabu;)
* Selanjutnya, pisahkan warna citra (for(int j=0;j<tinggi;j++), for(int i=0;i<lebar;i++), warna = pixels[j\*lebar+i]; hingga abuabu[j][i] = (red[j][i] + green[j][i] + blue[j][i]) / 3;)
* Setelah itu, dari titik atas-kiri hingga bawah-kanan citra (for(int j=0;j<tinggi;j++), for(int i=0;i<lebar;i++)) lakukan:
  + Buat citra normal, set warna kuas menjadi warna asli citra dan gambar titiknya (g.setColor(new Color(red[j][i],green[j][i],blue[j][i])); dan g.drawLine(i+lebar+100,j+50, i+lebar+100,j+50);)
  + Untuk citra normal cerah, pertama-tama lakukan perbandingan:
    - Jika nilai channel red setelah ditambah nilai kecerahan lebih dari nilai maksimum (if(red[j][i]+kc > 255)), maka bulatkan nilai tersebut ke nilai maksimum dan masukkan ke tempred untuk selanjutnya digambarkan (tempred = 255;). Jika tidak, maka nilai hasil penambahan tersebutlah yang dimasukkan ke tempred (tempred = red[j][i]+kc;)
    - Jika nilai channel green setelah ditambah nilai kecerahan lebih dari nilai maksimum (if(green[j][i]+kc > 255)), maka bulatkan nilai tersebut ke nilai maksimum dan masukkan ke tempgreen (tempgreen = 255;). Jika tidak, maka nilai hasil penambahan tersebutlah yang dimasukkan ke tempgreen (tempgreen = green[j][i]+kc;)
    - Jika nilai channel blue setelah ditambah nilai kecerahan lebih dari nilai maksimum (if(blue[j][i]+kc > 255)), maka bulatkan nilai tersebut ke nilai maksimum dan masukkan ke tempblue untuk selanjutnya digambarkan (tempblue = 255;). Jika tidak, maka nilai hasil penambahan tersebutlah yang dimasukkan ke tempblue (tempblue = blue[j][i]+kc;)
    - Setelah ketiga channel warna baru didapat, masukkan ke kuas (g.setColor(new Color(tempred,tempgreen,tempblue));) dan gambarkan titiknya (g.drawLine(i+lebar+100+ lebar+50,j+50,i+lebar+100+lebar+50,j+50);)
  + Untuk citra normal gelap, lakukan perbandingan:
    - Jika nilai channel red setelah dikurangi nilai kecerahan kurang dari nilai minimum (if(red[j][i]+kg < 0)), maka bulatkan nilai tersebut ke nilai minimum dan masukkan ke tempred (tempred = 0;). Jika tidak, maka nilai hasil pengurangan tersebutlah yang dimasukkan ke tempred (tempred = red[j][i]+kg;)
    - Jika nilai channel green setelah dikurangi nilai kecerahan kurang dari nilai minimum (if(green[j][i]+kg < 0)), maka bulatkan nilai tersebut ke nilai minimum dan masukkan ke tempgreen (tempgreen = 0;). Jika tidak, maka nilai hasil pengurangan tersebutlah yang dimasukkan ke tempgreen (tempgreen = green[j][i]+kg;)
    - Jika nilai channel blue setelah dikurangi nilai kecerahan kurang dari nilai minimum (if(blue[j][i]+kg < 0)), maka bulatkan nilai tersebut ke nilai minimum dan masukkan ke tempblue (tempblue = 0;). Jika tidak, maka nilai hasil pengurangan tersebutlah yang dimasukkan ke tempblue (tempblue = blue[j][i]+kg;)
    - Setelah ketiga channel warna baru didapat, masukkan ke kuas (g.setColor(new Color(tempred,tempgreen,tempblue));) dan gambarkan titiknya (g.drawLine(i+lebar+100+ lebar+50+lebar+50,j+50,i+lebar+100+lebar+50+lebar+50,j+50);)
  + Untuk mendapatkan warna negatif dari citra normal, terapkan rumus pada setiap channel warna dan masukkan warna tersebut ke kuas (g.setColor(new Color(255-red[j][i],255-green[j][i],255-blue[j][i]));) kemudian gambar titiknya (g.drawLine(i+lebar+100+lebar+50+lebar+50 +lebar+50,j+50,i+lebar+100+lebar+50+lebar+50+lebar+50,j+50);)
  + Untuk citra biner, bandingkan nilai grayscale titik ((channel R+G+B)/3) dengan nilai threshold. Jika nilai grayscale titik lebih besar atau sama dengan threshold (if((red[j][i]+green[j][i]+blue[j][i]) / 3 >= threshold)), maka bulatkan nilai tersebut ke nilai maksimum (tempabu = 255;). Jika tidak maka bulatkan ke nilai minimum (tempabu = 0;). Ctt: tempabu digunakan karena citra biner sesungguhnya mirip dengan grayscale, dimana ketiga channel RGB-nya diisi dengan nilai yang sama
  + Setelah didapat hasil konversinya, set warna kuas (g.setColor(new Color(tempabu,tempabu, tempabu));) dan gambar titiknya (g.drawLine(i+lebar+100+lebar+50+lebar+50+lebar+50+ lebar+50,j+50,i+lebar+100+lebar+50+lebar+50+lebar+50+lebar+50,j+50);)
  + Selanjutnya untuk citra abu-abu (grayscale), pertama-tama buat citra grayscalenya (g.setColor(new Color(abuabu[j][i],abuabu[j][i],abuabu[j][i])); dan g.drawLine(i+lebar+100,j+160, i+lebar+100,j+160);)
  + Untuk citra abu-abu cerah, lakukan perbandingan. Jika nilai abu-abu hasil pencerahan lebih dari nilai maksimum (if (abuabu[j][i]+kc > 255)), maka bulatkan nilai tersebut ke maksimum dan masukkan ke variable tempabu (tempabu = 255;). Jika tidak, masukkan nilai hasil pencerahan tersebut ke tempabu (tempabu = abuabu[j][i]+kc;)
  + Setelah didapat nilai hasil pencerahan, masukkan ke kuas (g.setColor(new Color(tempabu,tempabu, tempabu));) dan gambar titiknya (g.drawLine(i+lebar+100+lebar+50,j+160,i+lebar+100 +lebar+50,j+160);)
  + Untuk citra abu-abu gelap, lakukan perbandingan. Jika nilai abu-abu hasil penggelapan kurang dari nilai minimum (if (abuabu[j][i]+kg < 0)), maka bulatkan nilai tersebut ke minimum dan masukkan ke variable tempabu (tempabu = 0;). Jika tidak, masukkan nilai hasil penggelapan tersebut ke tempabu (tempabu = abuabu[j][i]+kg;)
  + Setelah didapat nilainya, masukkan ke kuas (g.setColor(new Color(tempabu,tempabu,tempabu));) dan gambar titiknya (g.drawLine(i+lebar+100+lebar+50+lebar+50,j+160, i+lebar+100+lebar+50+lebar+50,j+160);)
  + Untuk citra abu-abu negatif, terapkan rumus ke nilai abu-abu titik dan masukkan nilainya ke kuas (g.setColor(new Color(255-abuabu[j][i],255-abuabu[j][i],255-abuabu[j][i]));) kemudian gambar titiknya (g.drawLine(i+lebar+100+lebar+50+lebar+50+lebar+50,j+160, i+lebar+100+lebar+50+lebar+50+lebar+50,j+160);)
  + Untuk citra biner, bandingkan nilai abu-abu titik dengan threshold. Jika nilai abu-abu titik lebih besar atau sama dengan threshold (if(abuabu[j][i] >= threshold)), maka bulatkan nilai tersebut ke maksimum (tempabu = 255;). Jika tidak, maka bulatkan ke minimum (tempabu = 0;)
  + Setelah didapat nilainya, masukkan ke kuas (g.setColor(new Color(tempabu,tempabu,tempabu));) dan gambar titiknya (g.drawLine(i+lebar+100+lebar+50+lebar+50+lebar+50+lebar+50,j+160, i+lebar+100+lebar+50+lebar+50+lebar+50+lebar+50,j+160);)

Setelah kode program yang baru selesai, compile dengan javac, buat html pemanggilnya dan lihat hasilnya melalui appletviewer.

LAPORAN PRAKTIKUM PENGOLAHAN CITRA

Pertemuan V: Kontras dan Histogram

Pada pertemuan kali ini, kita mempraktikkan mengenai pengaturan kontras pada citra. Secara singkat, operasi pengaturan kontras merupakan operasi pengaturan terhadap distribusi pixel-pixel ke dalam intensitas warna tertentu. Sebuah citra dapat memiliki banyak titik (pixel) dengan intensitas warna masing-masing titik yang beragam. Frekuensi kemunculan pixel dengan intensitas warna tertentu pada suatu citra menunjukkan seberapa besar dominansi dari intensitas warna tersebut terhadap citra secara keseluruhan. Operasi pengaturan kontras bertujuan untuk memperbaiki citra dengan mengurangi dominansi intensitas warna tertentu serta meningkatkan dominansi dari intensitas warna lain.

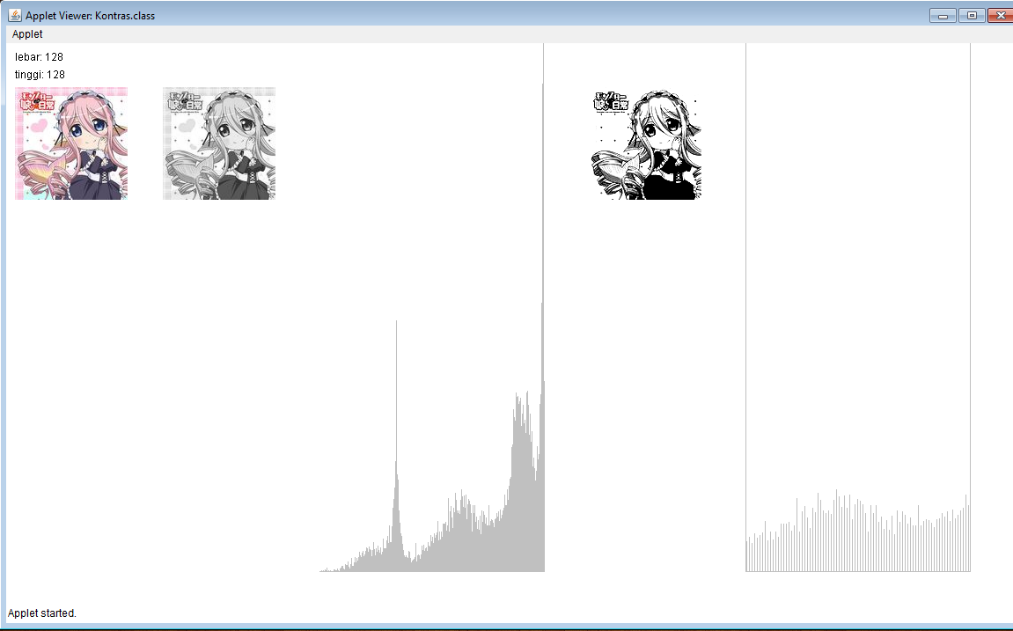
Operasi pengaturan kontras tidak lepas dari adanya perubahan terhadap frekuensi kemunculan intensitas warna tertentu. Untuk melihat pengaruh dari penerapan operasi pengaturan kontras, kita perlu untuk dapat melihat frekuensi kemunculan intensitas warna yang dimiliki oleh suatu citra. Alat bantu yang digunakan untuk melihat hal tersebut adalah histogram. Histogram disini merupakan sebuah grafik yang menggambarkan banyaknya (frekuensi) kemunculan dari seluruh intensitas warna, dari nilai 0 hingga 255, yang dimiliki oleh sebuah citra.

Terdapat banyak fungsi untuk menguatkan kontras pada suatu citra. Pada praktik kali ini, kita akan menggunakan salah satu fungsi yang paling sederhana:

Ket: f0 = intensitas warna pixel setelah penguatan kontras, fi = intensitas warna pixel sebelum penguatan kontras, G = koefisien penguatan kontras, P = nilai intensitas yang dipakai sebagai pusat pengontrasan

Pada rumus diatas, nilai koefisien G dan pusat pengontrasan P merupakan nilai yang kita masukkan sendiri untuk mencapai tingkat pengontrasan yang diharapkan. Di praktik kali ini, kita akan mencoba untuk mengambil kedua nilai tersebut melalui tag <param> pada html pemanggil.

Berikut hasil dari praktik pada pertemuan kali ini:



**Gambar 5.1. Hasil Praktik Pertemuan 5**

Pada praktik kali ini, kita akan melakukan operasi pengaturan kontras pada citra grayscalenya saja. Secara singkat, yang akan kita lakukan pada pertemuan kali ini adalah gambar citra grayscalenya, buat histogram dari citra grayscalenya, lakukan operasi kontras pada citra grayscalenya, gambar citra hasil kontras dan buat histogram dari citra hasil kontrasnya.

Praktik kali ini memerlukan variabel tambahan untuk menampung nilai koefisien G dan pusat pengontrasan P yang diambil dari dokumen html pemanggil applet. Proses pengambilan parameternya dilakukan pada inisialisasi class, tepatnya di dalam method init(). Karena akan diambil menggunakan method init() serta digunakan pada method paint(), maka deklarasi variabelnya perlu untuk dibuat global untuk kemudahan. Secara detil, variable global yang akan digunakan pada pertemuan ini adalah sebagai berikut:

|  |
| --- |
| Image img; //Untuk menampung citra  int lebar, tinggi; //Untuk menampung lebar dan tinggi citra  int parG,parP; //Untuk menampung nilai koefisien G dan pusat pengontrasan P  int warna; //Untuk menampung nilai pixel warna  String tampil; //Untuk menampilkan text |

Method init() juga perlu untuk dimodifikasi agar dapat mengambil nilai parameter dari dokumen html pemanggil applet. Berikut kode program method init() setelah modifikasi tersebut:

|  |
| --- |
| public void init()  {  img=this.getImage(this.getDocumentBase(),getParameter("filegambar"));  parG = Integer.parseInt(getParameter("parg"));  parP = Integer.parseInt(getParameter("parp"));  lebar = img.getWidth(this);  } |

* Pertama-tama, ambil file gambar dan masukkan ke variable “img” (img=this.getImage( this.getDocumentBase(),getParameter("filegambar"));). Berbeda dengan praktik pertemuan sebelumnya, kali ini nama file gambar diambil dari parameter pada dokumen html. Fungsi getParameter("filegambar") berfungsi untuk mengambil nilai parameter yang bernama “filegambar” pada dokumen html. Untuk lebih jelasnya, lihat deklarasi tag html terkait pada bagian kode program html pemanggil dibawah.
* Selanjutnya, ambil nilai parameter G dari dokumen html (parG = Integer.parseInt(getParameter("parg"));). Karena parG bertipe integer, maka nilai parameter yang diambil dari dokumen html perlu untuk dikonversi menjadi integer dengan perintah Integer.parseInt()
* Berikutnya, ambil nilai parameter P dari dokumen html (parP = Integer.parseInt(getParameter("parp"));)
* Terakhir, ambil lebar dari citra dan masukkan ke variablenya (lebar = img.getWidth(this);)

Isi method paint() pada pertemuan kali ini juga sebagian menggunakan kode program yang ada pada pertemuan sebelumnya. Bagian yang berbeda pada kode program kali ini terutama terletak setelah try-catch PixelGrabber. Berikut bagian yang berbeda tersebut:

|  |
| --- |
| -------------------------  g.setColor(Color.black);  int[][] red = new int[tinggi][lebar];  int[][] green = new int[tinggi][lebar];  int[][] blue = new int[tinggi][lebar];  int[][] abuabu = new int[tinggi][lebar];  int[] frek = new int[256];  int tempabu;  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  warna = pixels[j\*lebar+i];  red[j][i] = (warna >> 16) & 0xff;  green[j][i] = (warna >> 8) & 0xff;  blue[j][i] = (warna) & 0xff;  abuabu[j][i] = (red[j][i] + green[j][i] + blue[j][i]) / 3;  }  }    for(int i=0;i<256;i++)  {  frek[i] = 0;  }    for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  //Abu-abu  g.setColor(new Color(abuabu[j][i],abuabu[j][i],abuabu[j][i]));  g.drawLine(i+lebar+50,j+50,i+lebar+50,j+50);    //Simpan frekuensi  frek[abuabu[j][i]] = frek[abuabu[j][i]] + 1;  }  }    g.setColor(Color.lightGray);  for(int i=0;i<256;i++)  {  g.drawLine(lebar+lebar+100+i,600,lebar+lebar+100+i,600-frek[i]);  }    for(int i=0;i<256;i++)  {  frek[i] = 0;  }    for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  tempabu = parG \* (abuabu[j][i] - parP) + parP;  if(tempabu > 255)  {  tempabu = 255;  }  else if (tempabu < 0)  {  tempabu = 0;  }  frek[tempabu] = frek[tempabu]+1;  g.setColor(new Color(tempabu,tempabu,tempabu));  g.drawLine(i+lebar+lebar+256+150,j+50,i+lebar+lebar+256+150,j+50);  }  }    g.setColor(Color.lightGray);  for(int i=0;i<256;i++)  {  g.drawLine(lebar+lebar+lebar+256+200+i,600,lebar+lebar+lebar+256+200+i,600-frek[i]);  }  ------------------------- |

* Pertama, set warna kuas menjadi default/hitam (g.setColor(Color.black);)
* Selanjutnya, deklarasi bitmap untuk menampung channel warna red, green, blue serta abu-abu (int[][] red, int[][] green, int[][] blue dan int[][] abuabu)
* Berikutnya, deklarasi sebuah array satu dimensi bertipe integer untuk menampung frekuensi intensitas warna. Variabel ini memiliki 256 elemen, yang berfungsi untuk menyimpan frekuensi dari intensitas warna 0 hingga 255 (int[] frek = new int[256];)
* Deklarasikan pula variable sementara untuk menampung nilai warna hasil pengontrasan sebelum pencetakannya (int tempabu;)
* Setelah semua deklarasi selesai, lakukan pemisahan warna titik (for(int j=0;j<tinggi;j++), for(int i=0;i<lebar;i++), warna = pixels[j\*lebar+i]; hingga abuabu[j][i] = (red[j][i] + green[j][i] + blue[j][i]) / 3;)
* Berikutnya, kosongkan nilai array frekuensi (for(int i=0;i<256;i++), frek[i] = 0;)
* Setelah itu, dari atas-kiri hingga bawah-kanan citra (for(int j=0;j<tinggi;j++), for(int i=0;i<lebar;i++)) lakukan:
  + Cetak citra grayscalenya (g.setColor(new Color(abuabu[j][i],abuabu[j][i],abuabu[j][i])); dan g.drawLine(i+lebar+50,j+50,i+lebar+50,j+50);)
  + Tambah dan simpan frekuensi dari intensitas warna titik tersebut di array frekuensi (frek[abuabu[j][i]] = frek[abuabu[j][i]] + 1;)
* Selanjutnya, untuk penggambaran histogram, set warna kuas menjadi abu-abu terang (g.setColor(Color.lightGray);)
* Dari elemen pertama hingga terakhir array frekuensi (atau dari intensitas warna 0 hingga 255) (for(int i=0;i<256;i++)) lakukan penggambaran garis vertikal setinggi banyaknya frekuensi intensitas tersebut (g.drawLine(lebar+lebar+100+i,600,lebar+lebar+100+i,600-frek[i]);). Kumpulan dari garis-garis vertikal yang terbentuk dari frekuensi intensitas warna inilah yang akan menjadi histogram kita.
* Untuk selanjutnya, kosongkan lagi array frekuensi (for(int i=0;i<256;i++), frek[i] = 0;)
* Dari atas-kiri hingga bawah-kanan citra (for(int j=0;j<tinggi;j++), for(int i=0;i<lebar;i++)) lakukan:
  + Terapkan rumus penguatan kontras pada pixel tersebut, lalu simpan di variable “tempabu” (tempabu = parG \* (abuabu[j][i] - parP) + parP;)
  + Batasi nilai hasilnya, jika hasilnya diatas 255 maka bulatkan ke 255 (if(tempabu > 255), tempabu = 255;). Sedangkan jika hasilnya dibawah 0 maka bulatkan ke 0 (else if (tempabu < 0), tempabu = 0;)
  + Tambah dan simpan frekuensi dari intensitas warna titik tersebut di array frekuensi (frek[tempabu] = frek[tempabu]+1;)
  + Gambar citra hasil pengontrasan tersebut (g.setColor(new Color(tempabu,tempabu,tempabu)); dan g.drawLine(i+lebar+lebar+256+150,j+50,i+lebar+lebar+256+150,j+50);)
* Untuk histogram dari citra hasil pengontrasan, set warna kuas menjadi abu-abu terang (g.setColor(Color.lightGray);)
* Dan gambar histogramnya (for(int i=0;i<256;i++), g.drawLine(lebar+lebar+lebar+256+200+i ,600,lebar+lebar+lebar+256+200+i,600-frek[i]);)

Isi dari dokumen html pemanggil applet java ini mengalami perubahan karena penggunaan parameter untuk memasukkan nilai koefisien G dan pusat pengontrasan P. Perubahan tersebut terletak pada adanya penambahan tag <param> yang berfungsi untuk menampung nilai parameter tersebut. Berikut isi dari dokumen html tersebut:

|  |
| --- |
| <html>  <head>  </head>  <body>  <applet code=Kontras.class width=1200 height=600>  <param name = filegambar value = "NEOGDS-153345.jpg">  <param name = parp value = 190>  <param name = parg value = 3>  </applet>  </body>  <html> |

* Di antara tag pembuka dan penutup applet, tambahkan tiga buah tag <param>. Beri nama masing-masing tag tersebut dengan “filegambar”, “parp” dan “parg”. Nama tersebutlah yang akan digunakan untuk mengidentifikasi parameter di dalam method getParameter()
* Selanjutnya tentukan nilai (value) dari parameter tersebut. Parameter “filegambar” akan digunakan untuk mengambil file gambar, oleh karena itu nilai parameter ini adalah nama dari gambar tersebut. Parameter “parp” akan digunakan untuk menarik nilai pusat pengontrasan P, oleh karena itu parameter ini diisi dengan nilai intensitas warna yang akan digunakan sebagai pusat pengontrasan. Parameter “parg” akan digunakan untuk menarik nilai koefisien pengontrasan G, oleh karena itu parameter ini diisi dengan nilai koefisien pengontrasan yang diinginkan.

LAPORAN PRAKTIKUM PENGOLAHAN CITRA

Pertemuan VI: Filtering Spasial Non-Linier

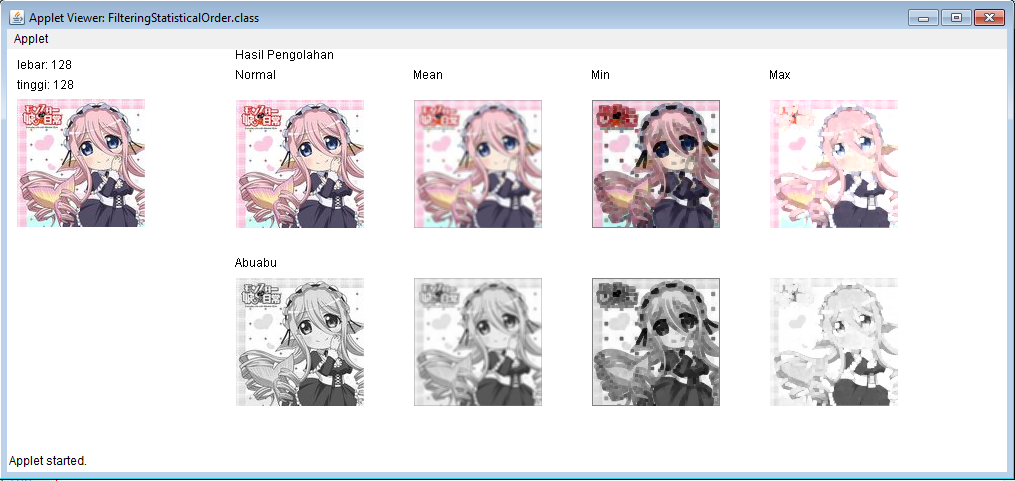
Pada pertemuan kali ini, kita mempraktikkan mengenai filtering, yaitu sebuah metode pengolahan citra untuk meloloskan suatu nilai titik tertentu dan menghilangkan (menolak) nilai lain berdasarkan hubungan nilai titik tersebut dengan nilai titik-titik lain (titik tetangga) pada citra yang diatur menggunakan filter. Terdapat dua jenis filtering, yaitu filtering spasial linier dan spasial non-linier. Pertemuan kali ini mempraktikkan mengenai filtering spasial non-linier, yaitu teknik untuk meloloskan suatu nilai yang didapat dari proses non-linier.

Filtering spasial non-linier merupakan operasi pengolahan citra yang bersifat lokal, artinya nilai hasil pengolahan tersebut terkait juga dengan nilai titik di area sekitar titik yang diolah. Filtering spasial non-linier disebut juga sebagai Statistical-Order Filtering karena operasi-operasinya terkait dengan pengurutan secara statistik intensitas antara piksel yang diolah dan piksel-piksel tetangganya. Contoh Statistical-Order Filtering misalnya adalah Mean Filtering, Median Filtering, Maximum dan Minimum Filtering.

Dalam melakukan filtering, kita perlu untuk menentukan suatu filter. Filter (disebut juga sebagai mask, kernel dan window) merupakan suatu matriks berukuran m x n (biasanya 3x3) yang digunakan untuk memfiltrasi citra. Setiap filter memiliki sel pusat (biasanya di tengah matriks) dimana sel bitmap yang akan di proses terdapat pada sel tersebut. Sel-sel disekitar sel pusat akan diisikan dengan sel-sel di sekitar sel bitmap yang akan di proses, sehingga ukuran matriks akan menentukan seberapa banyak sel di sekeliling sel yang akan diproses yang diambil dan berpengaruh terhadap hasil filter. Sel matriks juga dapat memiliki bobot yang akan menentukan hasil akhir proses filtrasi.

Filter akan mengambil nilai intensitas sel pusat (piksel yang akan diproses) dan sel-sel di sekitarnya dalam perhitungan untuk menentukan hasil filtrasi. Namun, apa yang terjadi apabila piksel yang akan diproses tersebut adalah piksel di pinggiran citra? Piksel pinggiran citra tidak memiliki piksel sekitar yang lengkap, oleh karena itu digunakanlah suatu kebijakan yang dinamakan kebijakan piksel tepi. Secara sederhana, kebijakan piksel tepi bermaksud untuk mengisi (menambahkan) piksel-piksel tambahan di pinggiran citra sehingga proses filter tidak menemui adanya piksel yang kosong. Nilai piksel tepi tersebut dapat ditentukan sesuai dengan kebutuhan filtrasi.

Berikut hasil dari praktik pada pertemuan kali ini:



**Gambar 6.1. Hasil Praktik Pertemuan 6**

Pada praktik kali ini, kita akan menerapkan filtering spasial non-linier terhadap citra dengan menggunakan kernel berukuran 3 x 3. Adapun operasi filter yang akan kita lakukan adalah Mean Filtering, Minimum Filtering dan Maximum Filtering. Filter akan kita terapkan pada citra warna dan abu-abu.

Program pada pertemuan kali ini memiliki sejumlah tambahan dibandingkan dengan pertemuan sebelumnya. Pertama-tama, pada program kali ini terdapat sebuah tambahan atribut (variable global), yaitu variable “m” dan “n“ yang bertipe integer (int m, n;). Variable ini berfungsi untuk menampung ukuran filter (kernel) yang akan digunakan dalam filtering dan diinisialisasi nilainya pada saat inisialisasi class (method init()). Karena ukuran kernel pada praktik kali ini adalah 3 x 3, maka inisialisasi nilai “m” dan “n” tersebut menjadi 3 pada saat inisialisasi class (tambahkan perintah -> m = 3; dan n = 3; di dalam method init()).

Selain itu, pada program kali ini juga ditambahkan sebuah method yang berfungsi untuk menambahkan dan mengeset nilai tepi pada bitmap citra mula-mula. Method tersebut adalah sebagai berikut:

|  |
| --- |
| public void setNilaiTepi(int processedBitmap[][], int nilai)  {  for(int i=0;i<tinggi+2;i++)  for(int j=0;j<lebar+2;j++)  if(j == 0 || i == 0 || j == lebar+1 || i == tinggi+1)  processedBitmap[i][j] = nilai;  } |

* Method setNilaiTepi() menerima dua buah parameter, yaitu “processedBitmap[][]” yang merupakan array dua dimensi yang menampung bitmap citra yang akan ditambahkan nilai tepi dan “nilai” yang digunakan untuk menentukan nilai intensitas piksel tepi yang akan ditambahkan
* Untuk prosesnya, lakukanlah pengulangan secara menyeluruh dari awal bitmap hingga akhir bitmap tersebut (for(int i=0;i<tinggi+2;i++), for(int j=0;j<lebar+2;j++)):
  + Untuk setiap pengulangan, lakukan pemeriksaan: jika piksel sekarang berada di tepi kiri bitmap (j == 0) atau di tepi atas bitmap (i == 0) atau di tepi kanan bitmap (j == lebar+1) atau di tepi bawah bitmap (i == tinggi+1), maka masukkan nilai piksel tepi pada piksel tersebut (processedBitmap[i][j] = nilai;)
* Ket: “i” dan “j” diulang dari 0 hingga tinggi + 2 dan lebar + 2 dilakukan dengan asumsi bahwa kernel yang digunakan berukuran 3 x 3, sehingga diperlukan masing-masing satu lapisan piksel tepi di sebelah kiri dan kanan (lebar + 2) serta atas dan bawah (tinggi + 2) dari citra. Untuk menampung nilai piksel tepi ini, maka ukuran bitmap penampung citra mula-mula jugalah harus berukuran sedikit lebih besar (lebar+2, tinggi+2). Lihatlah deklarasi bitmap mula-mula ini serta bagaimana pemasukkan nilai-nilai intensitas citra mula-mula pada bitmap ini di bagian method paint().

Selanjutnya, proses-proses untuk filtrasi kita masukkan di dalam method paint(). Sebagian kode program method ini menggunakan kode program yang ada di pertemuan sebelumnya. Bagian yang berbeda (bagian filtrasi) pada pertemuan kali ini terletak setelah try catch PixelGrabber. Berikut bagian yang berbeda tersebut:

|  |
| --- |
| -------------------------  int[][] red = new int[tinggi+2][lebar+2];  int[][] green = new int[tinggi+2][lebar+2];  int[][] blue = new int[tinggi+2][lebar+2];  int[][] abuabu = new int[tinggi+2][lebar+2];  int[][] bitmaphasilred = new int[tinggi][lebar];  int[][] bitmaphasilgreen = new int[tinggi][lebar];  int[][] bitmaphasilblue = new int[tinggi][lebar];  int[][] bitmaphasilabuabu = new int[tinggi][lebar];    g.drawString("Hasil Pengolahan",lebar+100,10);  g.drawString("Normal",lebar+100,30);  g.drawString("Mean",lebar+100+lebar+50,30);  g.drawString("Min",lebar+100+lebar+50+lebar+50,30);  g.drawString("Max",lebar+100+lebar+50+lebar+50+lebar+50,30);  g.drawString("Abuabu",lebar+100,tinggi+90);  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  warna = pixels[j\*lebar+i];  red[j+1][i+1] = (warna >> 16) & 0xff;  green[j+1][i+1] = (warna >> 8) & 0xff;  blue[j+1][i+1] = (warna) & 0xff;  abuabu[j+1][i+1] = (red[j+1][i+1] + green[j+1][i+1] + blue[j+1][i+1]) / 3;  }  }  //isi nilai tepi  setNilaiTepi(red, 128);  setNilaiTepi(green, 128);  setNilaiTepi(blue, 128);  setNilaiTepi(abuabu, 128);  //Filtering: Mean  for(int j=1;j<=tinggi;j++)  {  for(int i=1;i<=lebar;i++)  {  bitmaphasilred[j-1][i-1] = 0;  bitmaphasilgreen[j-1][i-1] = 0;  bitmaphasilblue[j-1][i-1] = 0;  bitmaphasilabuabu[j-1][i-1] = 0;    for(int y=j-(n/2);y<=j+(n/2);y++)  for(int x=i-(m/2);x<=i+(m/2);x++)  {  bitmaphasilred[j-1][i-1] = bitmaphasilred[j-1][i-1] + red[y][x];  bitmaphasilgreen[j-1][i-1] = bitmaphasilgreen[j-1][i-1] + green[y][x];  bitmaphasilblue[j-1][i-1] = bitmaphasilblue[j-1][i-1] + blue[y][x];  bitmaphasilabuabu[j-1][i-1] = bitmaphasilabuabu[j-1][i-1] + abuabu[y][x];  }  bitmaphasilred[j-1][i-1] = bitmaphasilred[j-1][i-1] / (m \* n);  bitmaphasilgreen[j-1][i-1] = bitmaphasilgreen[j-1][i-1] / (m \* n);  bitmaphasilblue[j-1][i-1] = bitmaphasilblue[j-1][i-1] / (m \* n);  bitmaphasilabuabu[j-1][i-1] = bitmaphasilabuabu[j-1][i-1] / (m \* n);  }  }    //Tampilkan hasil  for(int j=1;j<=tinggi;j++)  {  for(int i=1;i<=lebar;i++)  {  //Normal  g.setColor(new Color(red[j][i],green[j][i],blue[j][i]));  g.drawLine(i+lebar+100,j+50,i+lebar+100,j+50);    //Abu-abu  g.setColor(new Color(abuabu[j][i],abuabu[j][i],abuabu[j][i]));  g.drawLine(i+lebar+100,j+tinggi+100,i+lebar+100,j+tinggi+100);    //Hasil filter mean normal  g.setColor(new Color(bitmaphasilred[j-1][i-1],bitmaphasilgreen[j-1][i-1],bitmaphasilblue[j-1][i-1]));  g.drawLine(i+lebar+100+lebar+50,j+50,i+lebar+100+lebar+50,j+50);    //Hasil filter mean abu-abu  g.setColor(new Color(bitmaphasilabuabu[j-1][i-1],bitmaphasilabuabu[j-1][i-1],bitmaphasilabuabu[j-1][i-1]));  g.drawLine(i+lebar+100+lebar+50,j+tinggi+100,i+lebar+100+lebar+50,j+tinggi+100);  }  }    //Filtering: Min  for(int j=1;j<=tinggi;j++)  {  for(int i=1;i<=lebar;i++)  {  bitmaphasilred[j-1][i-1] = 255;  bitmaphasilgreen[j-1][i-1] = 255;  bitmaphasilblue[j-1][i-1] = 255;  bitmaphasilabuabu[j-1][i-1] = 255;    for(int y=j-(n/2);y<=j+(n/2);y++)  for(int x=i-(m/2);x<=i+(m/2);x++)  {  if(bitmaphasilred[j-1][i-1] > red[y][x]) bitmaphasilred[j-1][i-1] = red[y][x];  if(bitmaphasilgreen[j-1][i-1] > green[y][x]) bitmaphasilgreen[j-1][i-1] = green[y][x];  if(bitmaphasilblue[j-1][i-1] > blue[y][x]) bitmaphasilblue[j-1][i-1] = blue[y][x];  if(bitmaphasilabuabu[j-1][i-1] > abuabu[y][x]) bitmaphasilabuabu[j-1][i-1] = abuabu[y][x];  }  }  }    //Tampilkan hasil  for(int j=1;j<=tinggi;j++)  {  for(int i=1;i<=lebar;i++)  {  //Hasil filter min normal  g.setColor(new Color(bitmaphasilred[j-1][i-1],bitmaphasilgreen[j-1][i-1],bitmaphasilblue[j-1][i-1]));  g.drawLine(i+lebar+100+lebar+50+lebar+50,j+50,i+lebar+100+lebar+50+lebar+50,j+50);    //Hasil filter min abu-abu  g.setColor(new Color(bitmaphasilabuabu[j-1][i-1],bitmaphasilabuabu[j-1][i-1],bitmaphasilabuabu[j-1][i-1]));  g.drawLine(i+lebar+100+lebar+50+lebar+50,j+tinggi+100,i+lebar+100+lebar+50+lebar+50,j+tinggi+100);  }  }    //Filtering: Max  for(int j=1;j<=tinggi;j++)  {  for(int i=1;i<=lebar;i++)  {  bitmaphasilred[j-1][i-1] = 0;  bitmaphasilgreen[j-1][i-1] = 0;  bitmaphasilblue[j-1][i-1] = 0;  bitmaphasilabuabu[j-1][i-1] = 0;    for(int y=j-(n/2);y<=j+(n/2);y++)  for(int x=i-(m/2);x<=i+(m/2);x++)  {  if(bitmaphasilred[j-1][i-1] < red[y][x]) bitmaphasilred[j-1][i-1] = red[y][x];  if(bitmaphasilgreen[j-1][i-1] < green[y][x]) bitmaphasilgreen[j-1][i-1] = green[y][x];  if(bitmaphasilblue[j-1][i-1] < blue[y][x]) bitmaphasilblue[j-1][i-1] = blue[y][x];  if(bitmaphasilabuabu[j-1][i-1] < abuabu[y][x]) bitmaphasilabuabu[j-1][i-1] = abuabu[y][x];  }  }  }    //Tampilkan hasil  for(int j=1;j<=tinggi;j++)  {  for(int i=1;i<=lebar;i++)  {  //Hasil filter max normal  g.setColor(new Color(bitmaphasilred[j-1][i-1],bitmaphasilgreen[j-1][i-1],bitmaphasilblue[j-1][i-1]));  g.drawLine(i+lebar+100+lebar+50+lebar+50+lebar+50,j+50,i+lebar+100+lebar+50+lebar+50+lebar+50,j+50);    //Hasil filter max abu-abu  g.setColor(new Color(bitmaphasilabuabu[j-1][i-1],bitmaphasilabuabu[j-1][i-1],bitmaphasilabuabu[j-1][i-1]));  g.drawLine(i+lebar+100+lebar+50+lebar+50+lebar+50,j+tinggi+100,i+lebar+100+lebar+50+lebar+50+lebar+50,j+tinggi+100);  }  }  ------------------------- |

* Pertama-tama, deklarasi array bitmap untuk menampung channel warna (int[][] red = new int[tinggi+2][lebar+2]; hingga int[][] abuabu = new int[tinggi+2][lebar+2];). Perhatikan bahwa ukuran bitmap tersebut dibuat lebih besar untuk menampung piksel tepi di bagian atas, bawah, kiri dan kanan citra
* Selanjutnya, deklarasikan pula array bitmap untuk menampung hasil filtrasi (int[][] bitmaphasilred = new int[tinggi][lebar]; hingga int[][] bitmaphasilabuabu = new int[tinggi][lebar];). Ukuran bitmap hasil dibuat sama dengan ukuran citra dikarenakan pada bitmap hasil tidak perlu dilakukan filtering lagi
* Berikutnya, buat header (judul) masing-masing citra yang akan dicetak dan posisikan header tersebut (g.drawString("Hasil Pengolahan",lebar+100,10); hingga g.drawString("Abuabu",lebar+100, tinggi+90);)
* Selanjutnya, lakukan pengulangan, dari atas-kiri hingga bawah kanan citra (for(int j=0;j<tinggi;j++), for(int i=0;i<lebar;i++)):
  + Ambil informasi warna titik hasil pemecahan (warna = pixels[j\*lebar+i];)
  + Saring channel warna merahnya dan simpan di bitmap, kosongkan tepi bitmapnya (red[j+1][i+1] = (warna >> 16) & 0xff;)
  + Saring channel warna hijaunya dan simpan di bitmap, kosongkan tepi bitmapnya (green[j+1][i+1] = (warna >> 8) & 0xff;)
  + Saring channel warna birunya dan simpan di bitmap, kosongkan tepi bitmapnya (blue[j+1][i+1] = (warna) & 0xff;)
  + Hitung intensitas abu-abunya dan simpan hasilnya di bitmap, kosongkan tepi bitmapnya (abuabu[j+1][i+1] = (red[j+1][i+1] + green[j+1][i+1] + blue[j+1][i+1]) / 3;)
* Isi nilai tepi masing-masing bitmap dengan menggunakan method yang telah dibuat sebelumnya. Nilai tepi yang digunakan untuk filtering kali ini adalah 128 (setNilaiTepi(red, 128); hingga setNilaiTepi(abuabu, 128);)
* Untuk Mean Filtering, pertama-tama lakukan perulangan. Dari bagian atas-kiri hingga bawah-kanan bitmap (tidak termasuk tepi bitmap), lakukan (for(int j=1;j<=tinggi;j++), for(int i=1;i<=lebar;i++)):
  + Kosongkan bitmap hasil pada posisi piksel yang sama dengan posisi piksel yang diperiksa pada citra (bitmaphasilred[j-1][i-1] = 0; hingga bitmaphasilabuabu[j-1][i-1] = 0;)
  + Selanjutnya, terapkan filter pada titik yang diperiksa. Dari piksel paling atas-kiri hingga piksel paling bawah-kanan filter (for(int y=j-(n/2);y<=j+(n/2);y++), for(int x=i-(m/2);x<=i+(m/2);x++)) lakukan:
    - Akumulasi nilai titik-titik yang diperiksa pada filter dan simpan hasilnya pada bitmap hasil (bitmaphasilred[j-1][i-1] = bitmaphasilred[j-1][i-1] + red[y][x]; hingga bitmaphasilabuabu[j-1][i-1] = bitmaphasilabuabu[j-1][i-1] + abuabu[y][x];)
  + Setelah selesai, bagi nilai akumulasi tersebut dengan jumlah titik-titik (sel) pada filter untuk mendapatkan nilai intensitas rata-rata (bitmaphasilred[j-1][i-1] = bitmaphasilred[j-1][i-1] / (m \* n); hingga bitmaphasilabuabu[j-1][i-1] = bitmaphasilabuabu[j-1][i-1] / (m \* n);)
* Setelah hasil mean filtering didapat, tampilkan hasilnya. Dari atas-kiri hingga bawah-kanan bitmap (tidak termasuk tepi), lakukan (for(int j=1;j<=tinggi;j++), for(int i=1;i<=lebar;i++)):
  + Tampilkan citra normal (sebelum filter) pada posisinya (g.setColor(new Color(red[j][i], green[j][i],blue[j][i])); dan g.drawLine(i+lebar+100,j+50,i+lebar+100, j+50);)
  + Tampilkan citra abu-abu (sebelum filter) pada posisinya (g.setColor(new Color(abuabu[j][i], abuabu[j][i],abuabu[j][i])); dan g.drawLine(i+lebar+100,j+tinggi+100,i+lebar+100, j+tinggi+100);)
  + Tampilkan hasil mean filtering citra normal pada posisinya (g.setColor(new Color(bitmaphasilred[j-1][i-1],bitmaphasilgreen[j-1][i-1],bitmaphasilblue[j-1][i-1])); dan g.drawLine(i+lebar+ 100+lebar+50,j+50,i+lebar+100+lebar+50,j+50);)
  + Tampilkan hasil mean filtering citra grayscale pada posisinya (g.setColor(new Color(bitmaphasilabuabu[j-1][i-1],bitmaphasilabuabu[j-1][i-1],bitmaphasilabuabu[j-1][i-1])); dan g.drawLine(i+lebar+100+lebar+50,j+tinggi+100,i+lebar+100+lebar+50, j+tinggi+100);)
* Untuk Minimum Filtering, lakukan perulangan. Dari atas-kiri hingga bawah-kanan bitmap (tidak termasuk tepi bitmap), lakukan (for(int j=1;j<=tinggi;j++), for(int i=1;i<=lebar;i++)):
  + Isi bitmap hasil dengan nilai intensitas maksimum (bitmaphasilred[j-1][i-1] = 255; hingga bitmaphasilabuabu[j-1][i-1] = 255;)
  + Selanjutnya, terapkan filter pada titik yang diperiksa. Dari piksel paling atas-kiri hingga piksel paling bawah-kanan filter (for(int y=j-(n/2);y<=j+(n/2);y++), for(int x=i-(m/2);x<=i+(m/2);x++)) lakukan:
    - Jika nilai intensitas piksel yang diperiksa pada filter lebih kecil daripada nilai intensitas yang disimpan bitmap hasil, maka ganti nilai yang disimpan bitmap hasil tersebut dengan nilai intensitas piksel yang diperiksa (if(bitmaphasilred[j-1][i-1] > red[y][x]) bitmaphasilred[j-1][i-1] = red[y][x]; hingga if(bitmaphasilabuabu[j-1][i-1] > abuabu[y][x]) bitmaphasilabuabu[j-1][i-1] = abuabu[y][x];)
* Setelah hasil minimum filtering didapat, tampilkan hasilnya. Dari atas-kiri hingga bawah-kanan bitmap (tidak termasuk tepi), lakukan (for(int j=1;j<=tinggi;j++), for(int i=1;i<=lebar;i++)):
  + Tampilkan hasil minimum filtering citra normal pada posisinya (g.setColor(new Color(bitmaphasilred[j-1][i-1],bitmaphasilgreen[j-1][i-1],bitmaphasilblue[j-1][i-1])); dan g.drawLine(i+lebar+100+lebar+50+lebar+50,j+50,i+lebar+100+lebar+50+lebar+50,j+50);)
  + Tampilkan hasil minimum filtering citra grayscale pada posisinya (g.setColor(new Color(bitmaphasilabuabu[j-1][i-1],bitmaphasilabuabu[j-1][i-1],bitmaphasilabuabu[j-1][i-1])); dan g.drawLine(i+lebar+100+lebar+50+lebar+50,j+tinggi+100,i+lebar+100+ lebar+50+lebar+50,j+tinggi+100);)
* Untuk Maximum Filtering, lakukan perulangan. Dari atas-kiri hingga bawah-kanan bitmap (tidak termasuk tepi bitmap), lakukan (for(int j=1;j<=tinggi;j++), for(int i=1;i<=lebar;i++)):
  + Isi bitmap hasil dengan nilai intensitas minimum (bitmaphasilred[j-1][i-1] = 0; hingga bitmaphasilabuabu[j-1][i-1] = 0;)
  + Selanjutnya, terapkan filter pada titik yang diperiksa. Dari piksel paling atas-kiri hingga piksel paling bawah-kanan filter (for(int y=j-(n/2);y<=j+(n/2);y++), for(int x=i-(m/2);x<=i+(m/2);x++)) lakukan:
    - Jika nilai intensitas piksel yang diperiksa pada filter lebih besar daripada nilai intensitas yang disimpan bitmap hasil, maka ganti nilai yang disimpan bitmap hasil tersebut dengan nilai intensitas piksel yang diperiksa (if(bitmaphasilred[j-1][i-1] < red[y][x]) bitmaphasilred[j-1][i-1] = red[y][x]; hingga if(bitmaphasilabuabu[j-1][i-1] < abuabu[y][x]) bitmaphasilabuabu[j-1][i-1] = abuabu[y][x];)
* Setelah hasil maximum filtering didapat, tampilkan hasilnya. Dari atas-kiri hingga bawah-kanan bitmap (tidak termasuk tepi), lakukan (for(int j=1;j<=tinggi;j++), for(int i=1;i<=lebar;i++)):
  + Tampilkan hasil maximum filtering citra normal pada posisinya (g.setColor(new Color(bitmaphasilred[j-1][i-1],bitmaphasilgreen[j-1][i-1],bitmaphasilblue[j-1][i-1])); dan g.drawLine(i+lebar+100+lebar+50+lebar+50+lebar+50,j+50,i+lebar+100+lebar+50+lebar+ 50+lebar+50,j+50);)
  + Tampilkan hasil maximum filtering citra grayscale pada posisinya (g.setColor(new Color(bitmaphasilabuabu[j-1][i-1],bitmaphasilabuabu[j-1][i-1],bitmaphasilabuabu[j-1][i-1])); dan g.drawLine(i+lebar+100+lebar+50+lebar+50+lebar+50,j+tinggi+100,i+lebar+ 100+lebar+50+lebar+50+lebar+50,j+tinggi+100);)

Setelah selesai, compile programnya dan lihat hasilnya melalui appletviewer.

LAPORAN PRAKTIKUM PENGOLAHAN CITRA

Pertemuan VII: Filtering Spasial Linier – Konvolusi

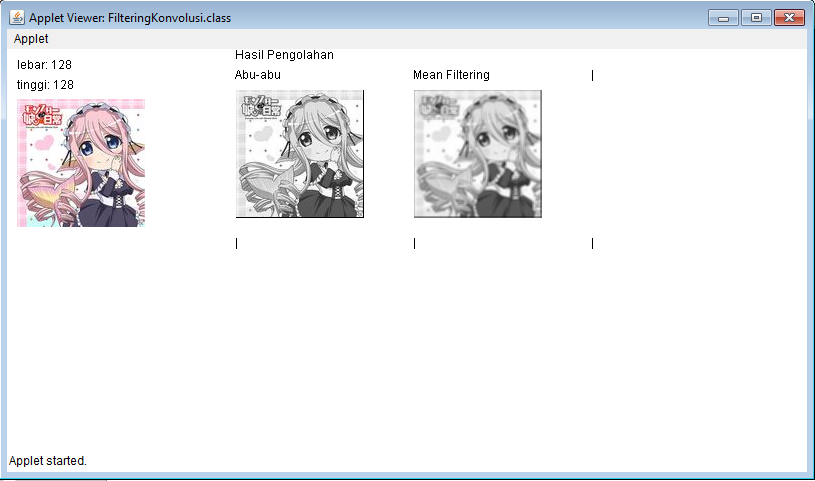
Pada pertemuan kali ini, kita mempraktikkan mengenai filtering jenis yang kedua, yakni filtering spasial linier. Filtering spasial linier merupakan filtering yang didapat dari proses perhitungan secara linier. Pada filtering spasial linier, filter/mask/kernel memiliki peran yang lebih penting dibandingkan pada filtering spasial non-linier. Hal ini dikarenakan sel-sel matriks filter/mask/kernel pada filtering spasial linier akan memiliki nilai yang kemudian akan dikalikan dengan nilai dari pixel gambar yang akan difilter beserta pixel-pixel tetangganya.

Proses perhitungan linier antara matriks filter dengan matriks sel yang akan difilter (yang terdiri dari sel pixel yang akan difilter beserta tetangganya) dibagi menjadi dua, yaitu korelasi dan konvolusi. Korelasi merupakan perhitungan (perkalian dan kemudian penjumlahan) dari setiap sel-sel matriks pertama dengan sel-sel matriks kedua pada lokasi yang bersesuaian. Jadi, misalnya filter yang digunakan berukuran 3 x 3, maka proses filter yang terjadi pada sebuah titik piksel pada gambar akan terjadi dengan cara sebagai berikut: nilai sel filter pada posisi kolom pertama dan baris pertama dikalikan dengan nilai sel citra pada posisi baris - 1 dan kolom - 1 dari sel yang akan difilter, nilai sel filter pada posisi kolom kedua baris pertama dikalikan dengan nilai sel citra pada posisi baris – 1 dan kolom dari sel yang akan difilter dan seterusnya hingga piksel yang akan difilter dan seluruh tetangganya telah dikalikan dengan filter. Terakhir, semua hasil perhitungan tersebut dijumlahkan untuk mendapatkan nilai sel baru hasil filter.

Konvolusi dapat dikatakan sebagai kebalikan dari korelasi, dimana perhitungan bukan terjadi pada sel-sel yang posisinya bersesuaian, melainkan pada posisi yang berkebalikan secara simetris. Jadi, misalnya filter yang digunakan berukuran 3 x 3, maka proses filter yang terjadi pada sebuah titik piksel pada gambar akan terjadi dengan cara sebagai berikut: nilai sel filter pada posisi kolom pertama dan baris pertama akan dikalikan dengan nilai sel citra pada posisi baris + 1 dan kolom + 1 dari sel yang akan difilter, nilai sel filter pada posisi kolom kedua dan baris pertama dikalikan dengan nilai sel citra pada posisi baris + 1 dan kolom dari sel yang akan difilter dan seterusnya hingga piksel yang akan difilter dan seluruh tetangganya telah dikalikan dengan filter. Terakhir, semua hasil perhitungan tersebut dijumlahkan untuk mendapatkan nilai sel baru hasil filter.

Pada pertemuan kali ini, kita akan mempraktikkan filtering spasial linier dengan perhitungan konvolusi. Adapun konsep filter yang akan kita terapkan adalah filter rata-rata. Filter rata-rata pada pertemuan kali ini berbeda dengan pertemuan sebelumnya. Pada pertemuan sebelumnya, nilai sel citra beserta tetangganya dijumlahkan dan kemudian dibagi dengan jumlah sel-sel yang dijumlahkan (atau dibagi dengan ukuran filter), sehingga perhitungan pada pertemuan sebelumnya tidak perlu menggunakan matriks filter. Pada pertemuan kali ini, nilai sel citra tersebut akan dikalikan dengan nilai sel matriks filter secara konvolusi, dimana setiap sel akan diisikan dengan nilai 1/ukuran matriks. Seluruh hasil perkalian tersebut akan dijumlahkan sehingga didapatkan nilai rata-rata antara piksel yang difilter dengan tetangganya.

Berikut hasil dari praktik pada pertemuan kali ini:



**Gambar 7.1. Hasil Praktik Filtering Spasial Linier – Konvolusi**

Sesuai dengan gambar diatas, di pertemuan kali ini kita akan menerapkan filtering rata-rata (mean) dengan konvolusi terhadap citra abu-abunya saja. Beberapa hal yang harus diperhatikan adalah pada pertemuan kali ini kita akan menggunakan sebagian konsep pada pertemuan sebelumnya, seperti ukuran filter dengan m x n dan kebijakan nilai tepi, dan proses filtering dengan konvolusi akan kita masukkan ke dalam sebuah method khusus, sehingga method tersebut dapat dipanggil untuk setiap kali filtering konvolusi dilakukan.

Sebagian kode program pada pertemuan kali ini menggunakan kode program pada pertemuan sebelumnya. Beberapa penambahan yang dilakukan antara lain adalah: pembuatan sebuah atribut (variable array dua dimensi) global bernama kernelMean[][] yang, sesuai namanya, berfungsi sebagai matriks kernel (filter) untuk proses filtering konvolusi dan sebuah method bernama konvolusi() yang juga sesuai namanya berfungsi untuk melakukan konvolusi. Array kernelMean[][] bertipe double dan deklarasi ukurannya dilakukan di dalam method initialization. Lengkapnya, isi dari method init() pada program kali ini adalah:

|  |
| --- |
| public void init()  {  img=this.getImage(this.getDocumentBase(),getParameter("filegambar"));  //tentukan ukuran kernel  m = 3;  n = 3;  //deklarasi kernel  kernelMean = new double[m+1][n+1];  //isi kernelMean  for(int y=1;y<=n;y++)  for(int x=1;x<=m;x++)  kernelMean[x][y] = 1 / (double)(m \* n);  lebar = img.getWidth(this);  } |

* Pertama-tama, ambil file citra dan masukkan ke variable img (img=this.getImage(this.getDocumentBase() ,getParameter("filegambar"));)
* Selanjutnya, tentukan ukuran kernel (m = 3; dan n = 3;)
* Lalu, lakukan deklarasi ukuran kernel (kernelMean = new double[m+1][n+1];). Ukuran dibuat + 1 untuk memudahkan dalam logika pemakaiannya.
* Berikutnya, isilah kernelMean tersebut, dari elemen ke-1 hingga ukuran kernel (m dan n) (for(int y=1;y<=n;y++) dan for(int x=1;x<=m;x++)), lakukan:
  + Isi sel kernel dengan nilai 1/ukuran kernel (kernelMean[x][y] = 1 / (double)(m \* n);)
* Terakhir, ambil lebar citra untuk pengujian (lebar = img.getWidth(this);)

Sementara itu, isi dari method konvolusi() adalah sebagai berikut:

|  |
| --- |
| public double konvolusi(int x, int y, int m, int n, int tempbitmap[][], double kernel[][])  {  double hasil;  hasil = 0;    for(int j=n;j>=1;j--)  for(int i=m;i>=1;i--)  hasil = hasil + kernel[i][j] \* tempbitmap[y-(j-n+(n/2))][x-(i-m+(m/2))];  //korelasi : hasil = hasil + kernel[i][j] \* tempbitmap[y+(j-1-(n/2))][x+(i-1-(m/2))];    //Thresholding  if(hasil > 255) hasil = 255;  if(hasil < 0) hasil = 0;    return hasil;  } |

* Method konvolusi() akan mengembalikan (*return*) nilai yang bertipe double, dimana nilai yang dikembalikan tersebut adalah nilai hasil filtering konvolusi sebuah pixel. Method ini menerima sejumlah parameter, yakni:
  + Posisi pixel citra yang akan di-filter (x dan y yang bertipe integer)
  + Ukuran kernel yang digunakan (m dan n yang bertipe integer)
  + Bitmap citra yang di-filter (tempbitmap yang berupa array dua dimensi bertipe integer)
  + Kernel/filter yang digunakan (kernel yang merupakan array dua dimensi bertipe double)
* Untuk cara kerja method ini, pertama-tama deklarasikan sebuah variable “hasil” yang bertipe double (double hasil;). Variable ini berfungsi untuk menampung nilai hasil filter yang akan dikembalikan (*return*) oleh method.
* Berikutnya, inisialisasi nilai variable tersebut (hasil = 0;)
* Lalu, lakukanlah perulangan, dari kanan-bawah hingga kiri-atas filter (for(int j=n;j>=1;j--) dan for(int i=m;i>=1;i--)) (Ket: perulangan dilakukan mulai dari m atau n hingga 1 karena array kernel terisi mulai dari indeks 1 hingga m atau n, bukan dari indeks 0 hingga m – 1 atau n – 1):
  + Perhitungan konvolusi terhadap pixel di posisi x dan y (hasil = hasil + kernel[i][j] \* tempbitmap[y-(j-n+(n/2))][x-(i-m+(m/2))];). Logikanya adalah sebagai berikut:
    - Kalikan isi sel filter pada posisi i dan j (kernel[i][j]) dengan sel citra pada posisi x-(i-m+(m/2)) dan y-(j-n+(n/2)) (tempbitmap[y-(j-n+(n/2))][x-(i-m+(m/2))]) dan akumulasikan hasilnya
  + Ctt: rumus untuk korelasi ditulis dalam komentar dibawahnya
* Selanjutnya, lakukanlah thresholding untuk hasil filter yang didapat. Jika hasil diatas 255 maka bulatkan ke 255 (if(hasil > 255) hasil = 255;)
* Dan jika hasil dibawah 0 maka bulatkan ke 0 (if(hasil < 0) hasil = 0;)
* Terakhir, kembalikan nilai hasil tersebut (return hasil;)

Selanjutnya, proses filtering akan dipanggil dalam method paint(). Isi sebagian method ini sama dengan pertemuan sebelumnya. Perbedaannya terdapat setelah try-catch PixelGrabber. Berikut bagian yang berbeda tersebut:

|  |
| --- |
| -------------------------  int[][] red = new int[tinggi+2][lebar+2];  int[][] green = new int[tinggi+2][lebar+2];  int[][] blue = new int[tinggi+2][lebar+2];  int[][] abuabu = new int[tinggi+2][lebar+2];  int[][] bitmaphasil = new int[tinggi+1][lebar+1];    g.drawString("Hasil Pengolahan",lebar+100,10);  g.drawString("Abu-abu",lebar+100,30);  g.drawString("Mean Filtering",lebar+100+lebar+50,30);  g.drawString("|",lebar+100+2\*lebar+2\*50,30);  g.drawString("|",lebar+100,tinggi+70);  g.drawString("|",lebar+100+lebar+50,tinggi+70);  g.drawString("|",lebar+100+2\*lebar+2\*50,tinggi+70);  for(int j=0;j<tinggi-1;j++)  {  for(int i=0;i<lebar-1;i++)  {  warna = pixels[j\*lebar+i];  red[j+1][i+1] = (warna >> 16) & 0xff;  green[j+1][i+1] = (warna >> 8) & 0xff;  blue[j+1][i+1] = (warna) & 0xff;  abuabu[j+1][i+1] = (red[j+1][i+1] + green[j+1][i+1] + blue[j+1][i+1]) / 3;  }  }  //Mean filtering  //isi filter tepi  setNilaiTepi(abuabu, 128);  //konvolusi  for(int j=1;j<=tinggi;j++)  {  for(int i=1;i<=lebar;i++)  {  bitmaphasil[j][i] = (int) konvolusi(i, j, m, n, abuabu, kernelMean);  }  }    //Tampilkan hasil proses  for(int j=1;j<=tinggi;j++)  {  for(int i=1;i<=lebar;i++)  {  //Abu-abu  g.setColor(new Color(abuabu[j][i],abuabu[j][i],abuabu[j][i]));  g.drawLine(i+lebar+100,j+40,i+lebar+100,j+40);    //Hasil konvolusi dengan filter mean  g.setColor(new Color(bitmaphasil[j][i],bitmaphasil[j][i],bitmaphasil[j][i]));  g.drawLine(i+2\*lebar+100+50,j+40,i+2\*lebar+100+50,j+40);  }  }  ------------------------- |

* Pertama-tama, deklarasikan bitmap citra (int[][] red = new int[tinggi+2][lebar+2]; hingga int[][] abuabu = new int[tinggi+2][lebar+2];) lalu deklarasikan bitmap hasil yang berfungsi menampung bitmap citra abu-abu hasil filtering (int[][] bitmaphasil = new int[tinggi+1][lebar+1];). Bitmap hasil ukurannya dibuat + 1 untuk kemudahan dalam pemakaiannya.
* Selanjutnya, cetak header (judul) tampilan citra dan beberapa garis pembatasnya (g.drawString("Hasil Pengolahan",lebar+100,10); hingga g.drawString("|",lebar+100+2\*lebar+2\*50,tinggi+70);)
* Lalu, lakukan perulangan, dari atas-kiri hingga bawah-kanan citra (for(int j=0;j<tinggi-1;j++) dan for(int i=0;i<lebar-1;i++)), lakukan:
  + Ambil pixel warna hasil pemecahan (warna = pixels[j\*lebar+i];), saring masing-masing channel warnanya (red[j+1][i+1] = (warna >> 16) & 0xff; hingga blue[j+1][i+1] = (warna) & 0xff;) dan hitung pixel abu-abunya (abuabu[j+1][i+1] = (red[j+1][i+1] + green[j+1][i+1] + blue[j+1][i+1]) / 3;)
* Setelah itu, lakukan pengisian pixel tepi, nilai yang dipakai adalah 128 (setNilaiTepi(abuabu, 128);)
* Berikutnya, lakukan perulangan, dari atas-kiri hingga bawah-kanan citra (minus tepi) (for(int j=1;j<=tinggi;j++), for(int i=1;i<=lebar;i++)), lakukan:
  + Filter pixel pada posisi i dan j (panggil method konvolusi) dan simpan hasilnya pada bitmap hasil (bitmaphasil[j][i] = (int) konvolusi(i, j, m, n, abuabu, kernelMean);)
* Terakhir, cetak hasilnya, dari atas-kiri hingga bawah-kanan citra (minus tepi) (for(int j=1;j<=tinggi;j++), for(int i=1;i<=lebar;i++)), lakukan:
  + Cetak citra grayscale sebelum filtering (g.setColor(new Color(abuabu[j][i], abuabu[j][i],abuabu[j][i])); dan g.drawLine(i+lebar+100,j+40,i+lebar+100,j+40);)
  + Cetak citra grayscale setelah filtering (g.setColor(new Color(bitmaphasil[j][i], bitmaphasil[j][i],bitmaphasil[j][i])); dan g.drawLine(i+2\*lebar+100+50,j+40, i+2\*lebar+100+50,j+40);)

Setelah selesai, compile programnya dan lihat hasilnya melalui appletviewer.

LAPORAN PRAKTIKUM PENGOLAHAN CITRA

Pertemuan VIII: Deteksi Tepi

Pada pertemuan kali ini, kita mempraktikkan mengenai deteksi tepi, yaitu suatu metode untuk mendapatkan tepi-tepi dari sebuah objek yang terdapat di dalam citra. Deteksi tepi merupakan salah satu konsep yang digunakan pada computer vision, dimana pada computer vision, komputer akan dibuat untuk mampu mengenali objek-objek yang terdapat dalam citra. Deteksi tepi sendiri diimplementasikan dengan menggunakan filtering, dengan suatu filter khusus.

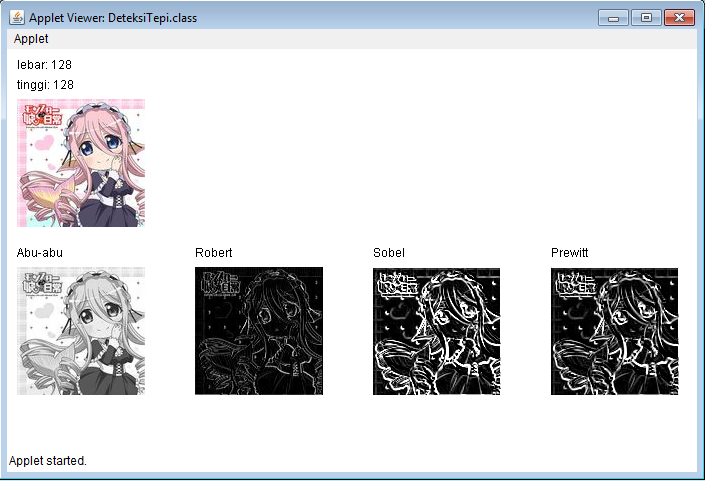
Deteksi tepi dapat dibagi menjadi dua, yaitu deteksi titik dan deteksi garis. Untuk melakukan deteksi titik, dapat digunakan high pass filtering dengan total koefisien nol. Untuk deteksi garis, filter yang digunakan memiliki total koefisien nol juga, namun susunan koefisien beserta nilainya telah disesuaikan sedemikian rupa untuk mendeteksi garis. Terdapat berbagai macam filter yang dibuat oleh para ahli yang dapat digunakan untuk melakukan deteksi garis. Pada praktik kali ini, kita akan menerapkan deteksi garis dengan menggunakan filter Robert, Sobel dan Prewitt.

Gx Gy Sx Sy Px Py

**Gambar 8.1. Filter Robert (Gx = Filter Garis Horizontal; Gy = Filter Garis Vertikal), Filter Sobel (Sx = Filter Garis Horizontal; Sy = Filter Garis Vertikal), Filter Prewitt (Px = Filter Garis Horizontal; Py = Filter Garis Vertikal)**

Untuk menerapkan filtering dengan deteksi tepi, maka kita perlu untuk melakukan konvolusi terhadap filter-filter deteksi tepi tersebut. Method konvolusi() yang dibuat pada pertemuan sebelumnya akan membantu dalam melakukan konvolusi terhadap filter-filter tersebut.

Berikut hasil dari praktik pada pertemuan kali ini:



**Gambar 8.2. Hasil Praktik Deteksi Tepi**

Di pertemuan kali ini, kita akan melakukan filtering terhadap citra abu-abu dengan menggunakan filter Robert, Sobel dan Prewitt dan kemudian menampilkan citra hasil filtering tersebut dalam applet.

Kode program pada pertemuan kali ini menggunakan sebagian kode program pada pertemuan sebelumnya. Beberapa bagian yang berbeda/ditambahkan diantaranya adalah:

* + - * + Deklarasi atribut (variable) global untuk kernel Robert, Sobel dan Prewitt

|  |
| --- |
| -------------------------  double Gx[][]; //Robert  double Gy[][];  double Sx[][]; //Sobel  double Sy[][];  double Px[][]; //Prewitt  double Py[][];  ------------------------- |

* + - * + Isi dari method init()

|  |
| --- |
| public void init()  {  img=this.getImage(this.getDocumentBase(),getParameter("filegambar"));  //deklarasi kernel  Gx = new double[2][2];  Gy = new double[2][2];  Sx = new double[3][3];  Sy = new double[3][3];  Px = new double[3][3];  Py = new double[3][3];  //isi kernel  Gx[0][0] = 1;  Gx[0][1] = 0;  Gx[1][0] = 0;  Gx[1][1] = -1;  Gy[0][0] = 0;  Gy[0][1] = -1;  Gy[1][0] = 1;  Gy[1][1] = 0;  //x  Sx[0][0] = -1;  Sx[0][1] = 0;  Sx[0][2] = 1;  Sx[1][0] = -2;  Sx[1][1] = 0;  Sx[1][2] = 2;  Sx[2][0] = -1;  Sx[2][1] = 0;  Sx[2][2] = 1;  //y  Sy[0][0] = 1;  Sy[0][1] = 2;  Sy[0][2] = 1;  Sy[1][0] = 0;  Sy[1][1] = 0;  Sy[1][2] = 0;  Sy[2][0] = -1;  Sy[2][1] = -2;  Sy[2][2] = -1;  //x  Px[0][0] = -1;  Px[0][1] = 0;  Px[0][2] = 1;  Px[1][0] = -1;  Px[1][1] = 0;  Px[1][2] = 1;  Px[2][0] = -1;  Px[2][1] = 0;  Px[2][2] = 1;  //y  Py[0][0] = 1;  Py[0][1] = 1;  Py[0][2] = 1;  Py[1][0] = 0;  Py[1][1] = 0;  Py[1][2] = 0;  Py[2][0] = -1;  Py[2][1] = -1;  Py[2][2] = -1;    lebar = img.getWidth(this);  } |

Pertama-tama, ambil file citra dan masukkan ke dalam variable img (img=this.getImage(this.getDocumentBase(),getParameter("filegambar"));)

Berikutnya, lakukan deklarasi ukuran masing-masing kernel (Gx = new double[2][2]; hingga Py = new double[3][3];)

Selanjutnya, lakukan pengisian masing-masing kernel (Gx[0][0] = 1; hingga Py[2][2] = -1;)

Terakhir, ambil lebar citra dan masukkan ke variable untuk pengetesan

Proses filteringnya akan dilakukan di dalam method paint(). Sebagian isi dari method paint() pada pertemuan kali ini sama dengan yang ada pada pertemuan sebelumnya. Bagian yang berbeda pada pertemuan kali terletak setelah try-catch PixelGrabber. Berikut bagian yang berbeda tersebut:

|  |
| --- |
| -------------------------  int[][] red = new int[tinggi+2][lebar+2];  int[][] green = new int[tinggi+2][lebar+2];  int[][] blue = new int[tinggi+2][lebar+2];  int[][] abuabu = new int[tinggi+2][lebar+2];  int[][] robertX = new int[tinggi][lebar];  int[][] robertY = new int[tinggi][lebar];  int[][] sobelX = new int[tinggi][lebar];  int[][] sobelY = new int[tinggi][lebar];  int[][] prewittX = new int[tinggi][lebar];  int[][] prewittY = new int[tinggi][lebar];  g.drawString("Abu-abu",10,tinggi+80);  g.drawString("Robert",10+lebar+50,tinggi+80);  g.drawString("Sobel",10+lebar+50+lebar+50,tinggi+80);  g.drawString("Prewitt",10+lebar+50+lebar+50+lebar+50,tinggi+80);  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  warna = pixels[j\*lebar+i];  red[j+1][i+1] = (warna >> 16) & 0xff;  green[j+1][i+1] = (warna >> 8) & 0xff;  blue[j+1][i+1] = (warna) & 0xff;  abuabu[j+1][i+1] = (red[j+1][i+1] + green[j+1][i+1] + blue[j+1][i+1]) / 3;  }  }  //isi filter tepi  setNilaiTepi(abuabu, 128);  //konvolusi  for(int j=1;j<=tinggi;j++)  {  for(int i=1;i<=lebar;i++)  {  robertX[j-1][i-1] = (int) konvolusi(i, j, 2, 2, abuabu, Gx);  robertY[j-1][i-1] = (int) konvolusi(i, j, 2, 2, abuabu, Gy);  sobelX[j-1][i-1] = (int) konvolusi(i, j, 3, 3, abuabu, Sx);  sobelY[j-1][i-1] = (int) konvolusi(i, j, 3, 3, abuabu, Sy);  prewittX[j-1][i-1] = (int) konvolusi(i, j, 3, 3, abuabu, Px);  prewittY[j-1][i-1] = (int) konvolusi(i, j, 3, 3, abuabu, Py);  }  }    //Tampilkan hasil proses  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  //Abu-abu  g.setColor(new Color(abuabu[j+1][i+1],abuabu[j+1][i+1],abuabu[j+1][i+1]));  g.drawLine(10+i,tinggi+50+40+j,10+i,tinggi+50+40+j);  //robert  if (robertX[j][i] > robertY[j][i])  {  g.setColor(new Color(robertX[j][i],robertX[j][i],robertX[j][i]));  }  else  {  g.setColor(new Color(robertY[j][i],robertY[j][i],robertY[j][i]));  }  g.drawLine(10+i+lebar+50,tinggi+50+40+j,10+i+lebar+50,tinggi+50+40+j);  //sobel  if (sobelX[j][i] > sobelY[j][i])  {  g.setColor(new Color(sobelX[j][i],sobelX[j][i],sobelX[j][i]));  }  else  {  g.setColor(new Color(sobelY[j][i],sobelY[j][i],sobelY[j][i]));  }  g.drawLine(10+i+lebar+50+lebar+50,tinggi+50+40+j,10+i+lebar+50+lebar+50,tinggi+50+40+j);  //prewitt  if (prewittX[j][i] > prewittY[j][i])  {  g.setColor(new Color(prewittX[j][i],prewittX[j][i],prewittX[j][i]));  }  else  {  g.setColor(new Color(prewittY[j][i],prewittY[j][i],prewittY[j][i]));  }  g.drawLine(10+i+lebar+50+lebar+50+lebar+50,tinggi+50+40+j,10+i+lebar+50+lebar+50+lebar+50,tinggi+50+40+j);  }  }  ------------------------- |

* Pertama-tama, deklarasikan bitmap untuk menampung channel warna (int[][] red = new int[tinggi+2][lebar+2]; hingga int[][] abuabu = new int[tinggi+2][lebar+2];)
* Kedua, deklarasikan bitmap untuk menampung hasil filtering (int[][] robertX = new int[tinggi][lebar]; hingga int[][] prewittY = new int[tinggi][lebar];)
* Ketiga, cetak header (judul) dari masing-masing citra yang akan ditampilkan (g.drawString("Abu-abu",10,tinggi+80); hingga g.drawString("Prewitt",10+lebar+50+lebar+50+lebar+50,tinggi+80);)
* Selanjutnya, lakukan pemisahan channel warna
* Setelah selesai, isi nilai tepi bitmap (setNilaiTepi(abuabu, 128);)
* Lalu lakukan filtering, dari atas-kiri hingga bawah kanan citra (for(int j=1;j<=tinggi;j++), for(int i=1;i<=lebar;i++)), lakukan:
  + Filtering (konvolusi) dengan filter Robert Gx (robertX[j-1][i-1] = (int) konvolusi(i, j, 2, 2, abuabu, Gx);)
  + Filtering (konvolusi) dengan filter Robert Gy (robertY[j-1][i-1] = (int) konvolusi(i, j, 2, 2, abuabu, Gy);)
  + Filtering (konvolusi) dengan filter Sobel Sx (sobelX[j-1][i-1] = (int) konvolusi(i, j, 3, 3, abuabu, Sx);)
  + Filtering (konvolusi) dengan filter Sobel Sy (sobelY[j-1][i-1] = (int) konvolusi(i, j, 3, 3, abuabu, Sy);)
  + Filtering (konvolusi) dengan filter Prewitt Px (prewittX[j-1][i-1] = (int) konvolusi(i, j, 3, 3, abuabu, Px);)
  + Filtering (konvolusi) dengan filter Prewitt Py (prewittY[j-1][i-1] = (int) konvolusi(i, j, 3, 3, abuabu, Py);)
* Setelah semuanya selesai, tampilkan hasilnya, dari atas-kiri hingga bawah-kanan citra (for(int j=0;j<tinggi;j++), for(int i=0;i<lebar;i++)), lakukan:
  + Cetak citra grayscale sebelum filtering (g.setColor(new Color(abuabu[j+1][i+1], abuabu[j+1][i+1],abuabu[j+1][i+1]));, g.drawLine(10+i,tinggi+50+40+j,10+i,tinggi+50+40 +j);)
  + Untuk hasil filtering dengan kernel Robert, Sobel dan Prewitt, pertama-tama tentukan:
    - Pada posisi pixel yang sama, apakah hasil filtering dengan filter Robert Gx lebih besar dari hasil filtering dengan filter Robert Gy (if (robertX[j][i] > robertY[j][i]))? Jika ya, maka gunakan hasil filtering Robert Gx sebagai warna kuas (g.setColor(new Color(robertX[j][i],robertX[j][i],robertX[j][i]));). Jika tidak (else), maka gunakan hasil filtering Robert Gy sebagai warna kuas (g.setColor(new Color(robertY[j][i],robertY[j][i],robertY[j][i]));). Setelah warna kuas ditentukan, cetak hasilnya (g.drawLine(10+i+lebar+50,tinggi+50+40+j,10+i+lebar+50, tinggi+50+40+j););
    - Pada posisi pixel yang sama, apakah hasil filtering dengan filter Sobel Sx lebih besar dari hasil filtering dengan filter Sobel Sy (if (sobelX[j][i] > sobelY[j][i]))? Jika ya, maka gunakan hasil filtering Sobel Sx sebagai warna kuas (g.setColor(new Color(sobelX[j][i],sobelX[j][i],sobelX[j][i]));). Jika tidak (else), maka gunakan hasil filtering Sobel Sy sebagai warna kuas (g.setColor(new Color(sobelY[j][i],sobelY[j][i],sobelY[j][i]));). Setelah warna kuas ditentukan, cetak hasilnya (g.drawLine(10+i+lebar+50+lebar+50,tinggi+50+40+j,10+i+lebar+50+lebar+50, tinggi+50+40+j););
    - Pada posisi pixel yang sama, apakah hasil filtering dengan filter Prewitt Px lebih besar dari hasil filtering dengan filter Prewitt Py (if (prewittX[j][i] > prewittY[j][i]))? Jika ya, maka gunakan hasil filtering Prewitt Px sebagai warna kuas (g.setColor(new Color(prewittX[j][i],prewittX[j][i],prewittX[j][i]));). Jika tidak (else), maka gunakan hasil filtering Prewitt Py sebagai warna kuas (g.setColor(new Color(prewittY[j][i],prewittY[j][i],prewittY[j][i]));). Setelah warna kuas ditentukan, cetak hasilnya (g.drawLine(10+i+lebar+50+lebar+50+lebar+50, tinggi+50+40+j,10+i+lebar+50+lebar+50+lebar+50,tinggi+50+40+j););

Setelah selesai, compile programnya dan lihat hasilnya melalui appletviewer.

LAPORAN PRAKTIKUM PENGOLAHAN CITRA

Pertemuan IX: Kontur

Pada pertemuan kali ini, kita mempelajari mengenai pendeteksian kontur. Kontur adalah rangkaian pixel-pixel yang membentuk suatu batas daerah. Melalui kontur, dapat diperoleh informasi-informasi yang berkaitan dengan ciri dari suatu objek dalam citra.

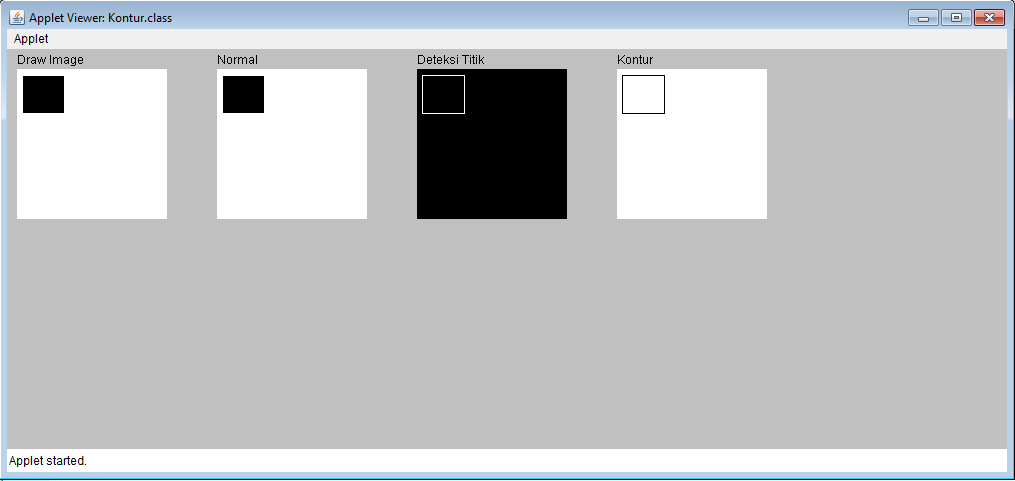
Kontur dapat diperoleh dengan melakukan sejumlah proses pada citra. Proses-proses yang dimaksud adalah:

1. Mengubah citra menjadi citra biner (dengan threshold tertentu)
2. Melakukan deteksi titik (filter dengan high pass filtering total koefisien 0)
3. Invers citra hasil deteksi titik

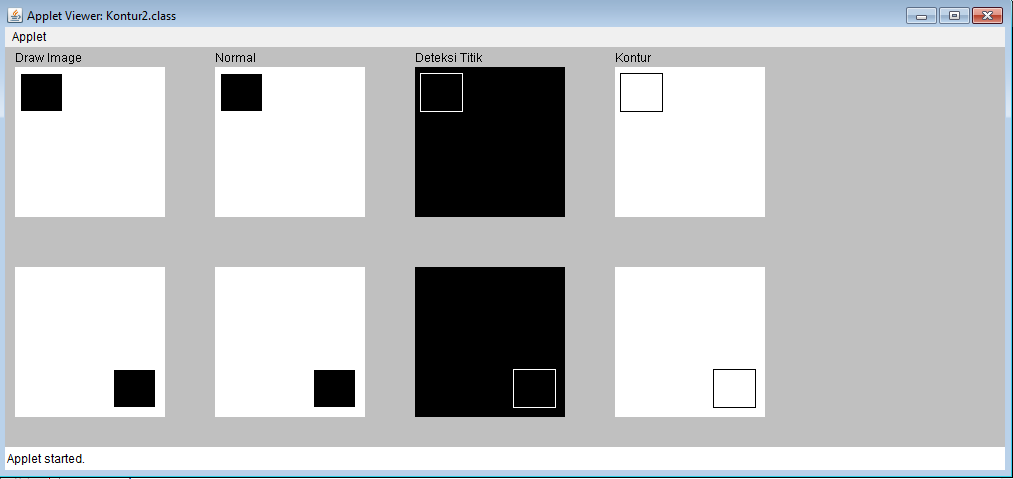
Pendeteksian titik dapat dilakukan dengan menggunakan high pass filter apa saja selama total koefisiennya 0. Namun di praktik kali ini, kita akan menggunakan high pass filter yang seperti berikut:

Pada praktik kali ini, kita akan mencoba pendeteksian kontur terhadap sejumlah citra yang memiliki objek yang berbeda-beda di dalamnya, baik dalam bentuk maupun posisinya. Citra yang digunakan adalah citra yang berlatar belakang berwarna putih dan dengan objek yang berwarna hitam. Citra yang dibutuhkan tersebut kita buat sendiri dengan menggunakan program pengolah citra yang ada.

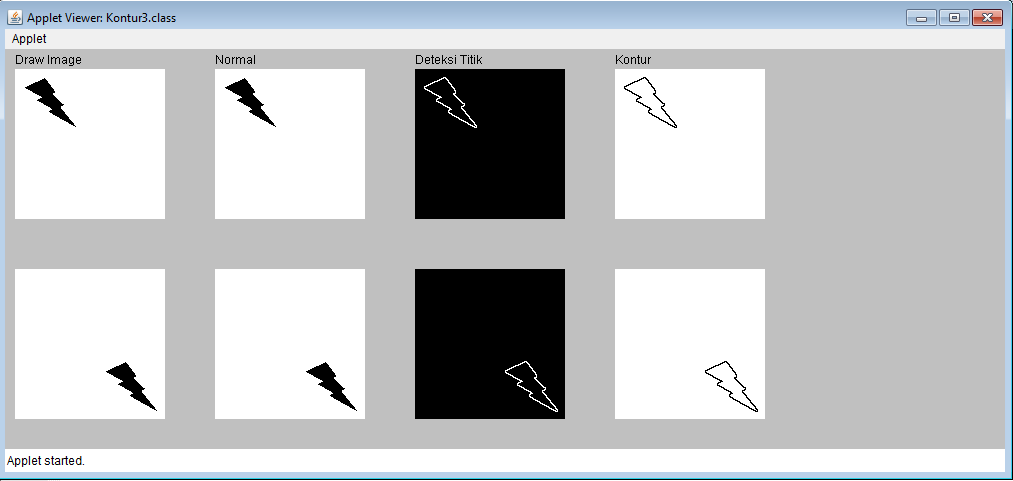
Kita akan melakukan tiga kali praktik, hasil dari masing-masing praktik tersebut adalah:



**Gambar 9.1. Hasil Praktik Kontur 1 – Objek Persegi**

****

**Gambar 9.2. Hasil Praktik Kontur 2 – Objek Persegi, berbeda posisi**

****

**Gambar 9.3. Hasil Praktik Kontur 3 – Objek Petir**

Sesuai dengan gambar-gambar diatas, di praktik kali ini kita akan melakukan thresholding citra, filtering dengan high pass filtering (deteksi titik) dan kemudian mencari konturnya melalui invers citra. Beberapa fungsi dari praktik sebelumnya seperti setNilaiTepi() dan konvolusi() juga digunakan kembali pada praktik kali ini.

**Praktik 1 – Kontur Persegi**

Kode program pada praktik kali ini sebagiannya menggunakan kode program pada pertemuan sebelumnya. Beberapa bagian yang berbeda diantaranya adalah:

1. Deklarasi variable array dua dimensi global untuk kernel high pass filtering (double K\_titik[][];)
2. Inisialisasi variable tersebut di dalam method init(), lengkapnya, isi dari method init() sekarang adalah:

|  |
| --- |
| public void init()  {  img=this.getImage(this.getDocumentBase(),"square1.png");  //deklarasi kernel  K\_titik = new double[3][3];  K\_titik[0][0] = -1;  K\_titik[0][1] = -1;  K\_titik[0][2] = -1;  K\_titik[1][0] = -1;  K\_titik[1][1] = 8;  K\_titik[1][2] = -1;  K\_titik[2][0] = -1;  K\_titik[2][1] = -1;  K\_titik[2][2] = -1;  lebar = img.getWidth(this);  } |

* Pertama-tama, ambil file citra
* Kemudian, deklarasi ukuran kernel (K\_titik = new double[3][3];)
* Lalu, isi kernel tersebut (K\_titik[0][0] = -1; hingga K\_titik[2][2] = -1;)
* Terakhir, ambil lebar citra (lebar = img.getWidth(this);)

1. Buat method khusus (thresholding()) yang berfungsi untuk mengubah citra menjadi citra biner. Isi dari method tersebut adalah:

|  |
| --- |
| public int thresholding(int value,int threshold)  {  if (value > threshold) return 255; else return 0;  } |

* Method thresholding() berfungsi untuk melakukan konversi citra ke citra biner. Method ini menerima dua buah parameter, yakni “value” yang berisi nilai intensitas pixel yang akan di-threshold dan “threshold” yang berisi nilai ambang batas konversi
* Jika value lebih besar dari threshold (if (value > threshold)), maka kembalikan nilai maksimum (return 255;), jika tidak (else), maka kembalikan nilai minimum (return 0;)

Proses utama pendeteksian kontur dilakukan di dalam method paint(). Sebagian isi dari method ini sama dengan yang ada pada pertemuan sebelumnya. Bagian yang berbeda hanya terletak setelah try-catch PixelGrabber. Berikut bagian yang berbeda tersebut:

|  |
| --- |
| -------------------------  int[][] red = new int[tinggi+2][lebar+2];  int[][] green = new int[tinggi+2][lebar+2];  int[][] blue = new int[tinggi+2][lebar+2];  int[][] abuabu = new int[tinggi+2][lebar+2];  int[][] temp = new int[tinggi+2][lebar+2];  int[][] titik = new int[tinggi][lebar];  int[][] kontur = new int[tinggi][lebar];  g.setColor(Color.black);  g.drawString("Draw Image",10,15);  g.drawString("Normal",10+lebar+50,15);  g.drawString("Deteksi Titik",10+lebar+50+lebar+50,15);  g.drawString("Kontur",10+lebar+50+lebar+50+lebar+50,15);  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  warna = pixels[j\*lebar+i];  red[j+1][i+1] = (warna >> 16) & 0xff;  green[j+1][i+1] = (warna >> 8) & 0xff;  blue[j+1][i+1] = (warna) & 0xff;  abuabu[j+1][i+1] = (red[j+1][i+1] + green[j+1][i+1] + blue[j+1][i+1]) / 3;  }  }  //isi filter tepi  setNilaiTepi(abuabu, 255);  setNilaiTepi(temp, 255);  //thresholding  for(int j=1;j<=tinggi;j++)  {  for(int i=1;i<=lebar;i++)  {  temp[j][i] = thresholding(abuabu[j][i],127);  }  }    //Filtering  for(int j=1;j<=tinggi;j++)  {  for(int i=1;i<=lebar;i++)  {  titik[j-1][i-1] = (int) konvolusi(i, j, 3, 3, temp, K\_titik);  }  }  //thresholding 2  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  titik[j][i] = thresholding(titik[j][i],127);  }  }  //kontur (invers)  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  kontur[j][i] = 255-titik[j][i];  }  }  //Cetak  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  //Normal  g.setColor(new Color(red[j+1][i+1],green[j+1][i+1],blue[j+1][i+1]));  g.drawLine(10+lebar+50+i,20+j,10+lebar+50+i,20+j);    //Deteksi Titik  g.setColor(new Color(titik[j][i],titik[j][i],titik[j][i]));  g.drawLine(10+i+lebar+50+lebar+50,20+j,10+i+lebar+50+lebar+50,20+j);    //Kontur  g.setColor(new Color(kontur[j][i],kontur[j][i],kontur[j][i]));  g.drawLine(10+i+lebar+50+lebar+50+lebar+50,20+j,10+i+lebar+50+lebar+50+lebar+50,20+j);  }  }  ------------------------- |

* Pertama-tama, deklarasi bitmap-bitmap channel warna (int[][] red = new int[tinggi+2][lebar+2]; hingga int[][] abuabu = new int[tinggi+2][lebar+2];)
* Selanjutnya, deklarasi bitmap untuk menampung citra hasil thresholding (int[][] temp = new int[tinggi+2][lebar+2];)
* Deklarasikan pula bitmap untuk menampung citra setelah deteksi titik (int[][] titik = new int[tinggi][lebar];) dan bitmap untuk menampung citra kontur (int[][] kontur = new int[tinggi][lebar];)
* Kemudian, cetak header (judul) dari masing-masing citra (g.setColor(Color.black); hingga g.drawString("Kontur",10+lebar+50+lebar+50+lebar+50,15);)
* Setelah itu, pisahkan channel warna masing-masing pixel pada citra (warna = pixels[j\*lebar+i]; hingga abuabu[j+1][i+1] = (red[j+1][i+1] + green[j+1][i+1] + blue[j+1][i+1]) / 3;)
* Berikutnya, set nilai tepi untuk bitmap abu-abu dan temp (setNilaiTepi(abuabu, 255); dan setNilaiTepi(temp, 255);)
* Kemudian, lakukan thresholding. Dari atas-kiri hingga bawah-kanan citra (for(int j=1;j<=tinggi;j++), for(int i=1;i<=lebar;i++)), lakukan:
  + Tresholding pixel pada bitmap abu-abu di posisi i dan j dengan nilai 127 dan kemudian simpan hasilnya di bitmap temp (temp[j][i] = thresholding(abuabu[j][i],127);)
* Setelah selesai, lakukanlah deteksi titik. Dari atas-kiri hingga bawah-kanan citra (for(int j=1;j<=tinggi;j++), for(int i=1;i<=lebar;i++)), lakukan:
  + Filtering konvolusi bitmap temp dengan kernel high pass filtering dan kemudian simpan hasilnya di bitmap titik (titik[j-1][i-1] = (int) konvolusi(i, j, 3, 3, temp, K\_titik);)
* Setelah itu, lakukanlah thresholding untuk kedua kalinya demi menghindari adanya nilai hasil filter yang bukan 0/255. Dari atas-kiri hingga bawah-kanan citra (for(int j=0;j<tinggi;j++), for(int i=0;i<lebar;i++)), lakukan:
  + Thresholding bitmap titik dengan nilai 127 (titik[j][i] = thresholding(titik[j][i],127);)
* Berikutnya, temukan kontur citra. Dari atas-kiri hingga bawah kanan citra (for(int j=0;j<tinggi;j++), for(int i=0;i<lebar;i++)), lakukan:
  + Invers semua pixel pada bitmap titik dan simpan hasilnya pada bitmap kontur (kontur[j][i] = 255-titik[j][i];)
* Setelah semuanya selesai, cetak hasilnya

**Praktik 2 – Kontur Persegi – Berbeda Posisi**

Di praktik yang kedua ini, kita akan menambahkan citra dengan objek persegi yang berbeda posisi tepat di bawah citra yang pertama dan kemudian melakukan operasi pendeteksian kontur yang sama terhadap citra kedua tersebut. Karena sebagian besar operasinya sama, praktik kedua ini lebih menekankan pada penambahan variable-variable baru serta penggunaan variable tersebut untuk menampung dan memproses citra yang kedua. Berikut sejumlah penambahan yang dilakukan pada praktik yang kedua ini:

1. Deklarasi atribut global

|  |
| --- |
| Image img,img2;  int lebar, tinggi;  int lebar2, tinggi2;  int warna, warna2;  double K\_titik[][]; |

1. Method init()

|  |
| --- |
| public void init()  {  img=this.getImage(this.getDocumentBase(),"square1.png");  img2=this.getImage(this.getDocumentBase(),"square2.png");  //deklarasi kernel  K\_titik = new double[3][3];  K\_titik[0][0] = -1;  K\_titik[0][1] = -1;  K\_titik[0][2] = -1;  K\_titik[1][0] = -1;  K\_titik[1][1] = 8;  K\_titik[1][2] = -1;  K\_titik[2][0] = -1;  K\_titik[2][1] = -1;  K\_titik[2][2] = -1;  lebar = img.getWidth(this);  lebar2 = img2.getWidth(this);  } |

1. Method paint()

|  |
| --- |
| public void paint(Graphics g)  {  //Set warna background  g.setColor(Color.lightGray);  g.fillRect(0,0,2800,800);  //ambil lebar dan tinggi citra  lebar = img.getWidth(this);  tinggi = img.getHeight(this);  lebar2 = img2.getWidth(this);  tinggi2 = img2.getHeight(this);    g.drawImage(img,10,20,this); //Tampilkan citra dengan drawImage  g.drawImage(img2,10,20+tinggi2+50,this); //Tampilkan citra dengan drawImage    int[] pixels = new int[lebar\*tinggi]; //tentukan panjang array pixels  int[] pixels2 = new int[lebar2\*tinggi2];  PixelGrabber pg = new PixelGrabber(img,0,0,lebar,tinggi,pixels,0,lebar);  PixelGrabber pg2 = new PixelGrabber(img2,0,0,lebar2,tinggi2,pixels2,0,lebar2);  try  {  pg.grabPixels(); //grab piksel citra ke object pg.  pg2.grabPixels();  }  catch (InterruptedException ie)  {  System.out.println("Terjadi kesalahan saat mengambil data pixels");  ie.printStackTrace();  return;  }    int[][] red = new int[tinggi+2][lebar+2];  int[][] green = new int[tinggi+2][lebar+2];  int[][] blue = new int[tinggi+2][lebar+2];  int[][] abuabu = new int[tinggi+2][lebar+2];  int[][] temp = new int[tinggi+2][lebar+2];  int[][] titik = new int[tinggi][lebar];  int[][] kontur = new int[tinggi][lebar];  int[][] red2 = new int[tinggi+2][lebar+2];  int[][] green2 = new int[tinggi+2][lebar+2];  int[][] blue2 = new int[tinggi+2][lebar+2];  int[][] abuabu2 = new int[tinggi+2][lebar+2];  int[][] temp2 = new int[tinggi+2][lebar+2];  int[][] titik2 = new int[tinggi][lebar];  int[][] kontur2 = new int[tinggi][lebar];    g.setColor(Color.black);  g.drawString("Draw Image",10,15);  g.drawString("Normal",10+lebar+50,15);  g.drawString("Deteksi Titik",10+lebar+50+lebar+50,15);  g.drawString("Kontur",10+lebar+50+lebar+50+lebar+50,15);  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  warna = pixels[j\*lebar+i];  warna2 = pixels2[j\*lebar2+i];  red[j+1][i+1] = (warna >> 16) & 0xff;  green[j+1][i+1] = (warna >> 8) & 0xff;  blue[j+1][i+1] = (warna) & 0xff;  abuabu[j+1][i+1] = (red[j+1][i+1] + green[j+1][i+1] + blue[j+1][i+1]) / 3;    red2[j+1][i+1] = (warna2 >> 16) & 0xff;  green2[j+1][i+1] = (warna2 >> 8) & 0xff;  blue2[j+1][i+1] = (warna2) & 0xff;  abuabu2[j+1][i+1] = (red2[j+1][i+1] + green2[j+1][i+1] + blue2[j+1][i+1]) / 3;  }  }  //isi filter tepi  setNilaiTepi(abuabu, 255);  setNilaiTepi(temp, 255);    setNilaiTepi(abuabu2, 255);  setNilaiTepi(temp2, 255);  //thresholding  for(int j=1;j<=tinggi;j++)  {  for(int i=1;i<=lebar;i++)  {  temp[j][i] = thresholding(abuabu[j][i],127);  temp2[j][i] = thresholding(abuabu2[j][i],127);  }  }    //Filtering  for(int j=1;j<=tinggi;j++)  {  for(int i=1;i<=lebar;i++)  {  titik[j-1][i-1] = (int) konvolusi(i, j, 3, 3, temp, K\_titik);  titik2[j-1][i-1] = (int) konvolusi(i, j, 3, 3, temp2, K\_titik);  }  }  //thresholding 2  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  titik[j][i] = thresholding(titik[j][i],127);  titik2[j][i] = thresholding(titik2[j][i],127);  }  }  //kontur (invers)  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  kontur[j][i] = 255-titik[j][i];  kontur2[j][i] = 255-titik2[j][i];  }  }  //Cetak  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  //Normal  g.setColor(new Color(red[j+1][i+1],green[j+1][i+1],blue[j+1][i+1]));  g.drawLine(10+lebar+50+i,20+j,10+lebar+50+i,20+j);  g.setColor(new Color(red2[j+1][i+1],green2[j+1][i+1],blue2[j+1][i+1]));  g.drawLine(10+lebar+50+i,20+j+tinggi2+50,10+lebar+50+i,20+j+tinggi2+50);    //Deteksi Titik  g.setColor(new Color(titik[j][i],titik[j][i],titik[j][i]));  g.drawLine(10+i+lebar+50+lebar+50,20+j,10+i+lebar+50+lebar+50,20+j);  g.setColor(new Color(titik2[j][i],titik2[j][i],titik2[j][i]));  g.drawLine(10+i+lebar+50+lebar+50,20+j+tinggi2+50,10+i+lebar+50+lebar+50 ,20+j+tinggi2+50);    //Kontur  g.setColor(new Color(kontur[j][i],kontur[j][i],kontur[j][i]));  g.drawLine(10+i+lebar+50+lebar+50+lebar+50,20+j,10+i+lebar+50+lebar+50 +lebar+50,20+j);  g.setColor(new Color(kontur2[j][i],kontur2[j][i],kontur2[j][i]));  g.drawLine(10+i+lebar+50+lebar+50+lebar+50,20+j+tinggi2+50,10+i+lebar+50 +lebar+50+lebar+50,20+j+tinggi2+50);  }  }  } |

**Praktik 3 – Kontur Petir**

Praktik yang ini menggunakan kode program dari praktik 2. Hanya saja, file citra yang dipakai bukan citra dengan objek persegi, melainkan citra dengan objek petir di dalamnya.

LAPORAN PRAKTIKUM PENGOLAHAN CITRA

Pertemuan X: Kontur – Kode Rantai

Di pertemuan sebelumnya kita telah mempelajari mengenai kontur serta metode untuk mengolah citra sedemikian rupa sehingga diperoleh kontur objek dalam citra. Setelah kita mendapatkan citra dengan kontur objek, bagaimanakah cara untuk mengubah kontur tersebut menjadi informasi yang mengenali objek? Salah satu caranya adalah dengan menggunakan kode rantai.

Sederhananya, kode rantai merupakan rangkaian angka-angka (kode) yang diurutkan sesuai dengan arah pixel-pixel kontur yang membentuk citra. Pembacaan kode rantai dimulai dari sejak pixel kontur (objek) ditemukan dan diteruskan hingga seluruh pixel kontur dimasukkan ke dalam kode rantai. Dari pixel kontur awal, diperiksa pixel-pixel tetangganya untuk menemukan pixel dengan warna yang sama (atau mendekati) dengan pixel tersebut. Setelah ditemukan, catat arahnya dan mulai pemeriksaan yang sama di pixel tersebut. Ulangi proses tersebut sehingga seluruh pixel kontur tercatat di dalam kode rantai.

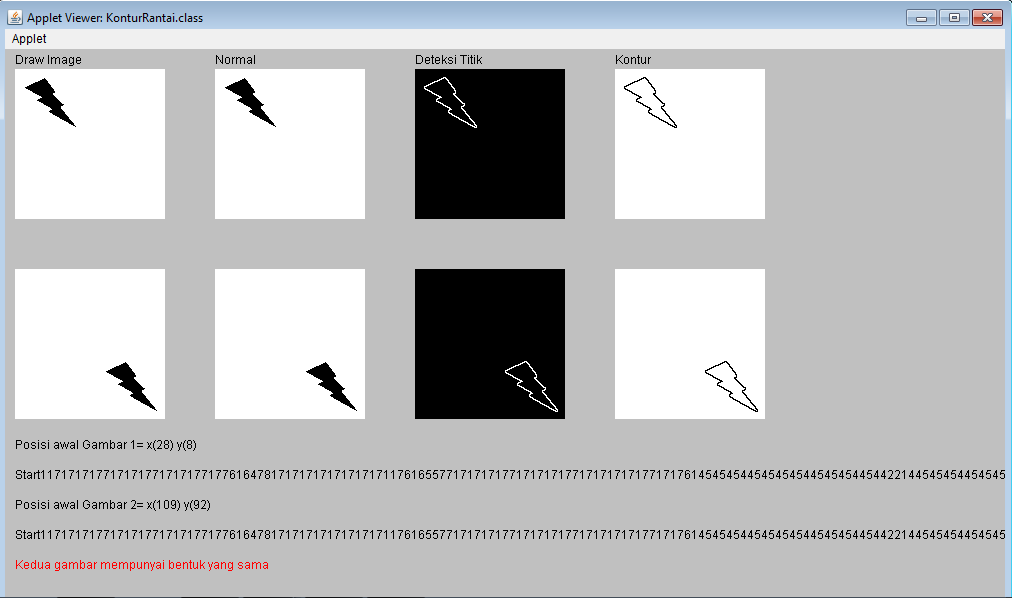
Arah pixel dapat direpresentasikan dengan menggunakan angka, misalnya dari angka 1 sampai 8. Masing-masing angka dapat merepresentasikan arah mana saja. Namun, di praktik kali ini, kita akan menggunakan representasi arah sebagai berikut:

|  |  |  |
| --- | --- | --- |
| 4 | 3 | 2 |
| 5 | x | 1 |
| 6 | 7 | 8 |

**Gambar 10.1. Representasi Arah:**

**1 = kanan, 2 = kanan-atas, 3 = atas, 4 = kiri-atas, 5 = kiri, 6 = kiri-bawah, 7 = bawah, 8 = kanan-bawah**

Berikut hasil dari praktik pada pertemuan kali ini:



**Gambar 10.2. Hasil Praktik Kode Rantai**

Sesuai dengan gambar diatas, pertemuan kali ini akan mengambil sebagian besar kode program pada praktik pertemuan sebelumnya (tepatnya kontur-praktik 2). Setelah pembacaan kontur, selanjutnya kita akan mencari kode rantai untuk tiap-tiap kontur objek pada citra yang pertama dan kedua. Setelah kode rantai kedua objek didapat, selanjutnya lakukan perbandingan kedua kode rantai tersebut. Jika kode rantainya sama maka kedua objek tersebut adalah objek yang sama, namun jika berbeda, maka kedua objek tersebut tidak sama.

Program kali ini sebagian besar menggunakan bagian dari program pada pertemuan sebelumnya. Berikut adalah bagian-bagian yang diubah/ditambahkan untuk program yang baru:

1. HTML pemanggil

Program kali ini menggunakan sejumlah parameter untuk mengambil file gambar serta nilai tepi (untuk kebijakan nilai tepi). Isi dari HTML pemanggil ini adalah:

|  |
| --- |
| <html>  <head>  </head>  <body>  <applet code=KonturRantai.class width=1000 height=600>  <PARAM name=filegambar VALUE="lightning1.png">  <PARAM name=filegambar2 VALUE="lightning2.png">  <PARAM name=tepi VALUE=255>  <PARAM name=tepi2 VALUE=255>  </applet>  </body>  </html> |

Sesuai dengan isi dokumen HTML diatas, kebijakan nilai tepi yang akan kita gunakan pada program kali ini adalah 255.

1. Deklarasi variable/atribut global untuk menampung kode rantai (String stackrantai,rantai1,rantai2;). “stackrantai” digunakan dalam method untuk mencari kode rantai, “rantai1” digunakan untuk menampung kode rantai dari objek yang pertama, sedangkan “rantai2” digunakan untuk menampung kode rantai dari objek yang kedua.
2. Method init()

Di dalam method ini, kita mengubah pengambilan file citra menjadi dengan menggunakan parameter (img=this.getImage(this.getDocumentBase(),getParameter("filegambar")); dan img2=this.getImage(this.getDocumentBase(),getParameter("filegambar2"));)

1. Pembuatan method koderantai() untuk mencari kode rantai. Berikut isi dari method tersebut:

|  |
| --- |
| public void koderantai(int x, int y, int bitmap[][])  {  //1  if(bitmap[y][x+1] == 0)  {  stackrantai = stackrantai + "1";  bitmap[y][x+1] = 1;  koderantai(x+1,y,bitmap);  }  //2  if(bitmap[y-1][x+1] == 0)  {  stackrantai = stackrantai + "2";  bitmap[y-1][x+1] = 2;  koderantai(x+1,y-1,bitmap);  }  //3  if(bitmap[y-1][x] == 0)  {  stackrantai = stackrantai + "3";  bitmap[y-1][x] = 3;  koderantai(x,y-1,bitmap);  }  //4  if(bitmap[y-1][x-1] == 0)  {  stackrantai = stackrantai + "4";  bitmap[y-1][x-1] = 4;  koderantai(x-1,y-1,bitmap);  }  //5  if(bitmap[y][x-1] == 0)  {  stackrantai = stackrantai + "5";  bitmap[y][x-1] = 5;  koderantai(x-1,y,bitmap);  }  //6  if(bitmap[y+1][x-1] == 0)  {  stackrantai = stackrantai + "6";  bitmap[y+1][x-1] = 6;  koderantai(x-1,y+1,bitmap);  }  //7  if(bitmap[y+1][x] == 0)  {  stackrantai = stackrantai + "7";  bitmap[y+1][x] = 7;  koderantai(x,y+1,bitmap);  }  //8  if(bitmap[y+1][x+1] == 0)  {  stackrantai = stackrantai + "8";  bitmap[y+1][x+1] = 8;  koderantai(x+1,y+1,bitmap);  }  } |

* Method koderantai() menerima tiga buah parameter, x dan y yang bertipe integer berfungsi untuk menunjukkan koordinat (x dan y) pixel yang akan diperiksa dan bitmap[][] merupakan array yang menampung bitmap citra yang berisi kontur objek.
* Lakukan pemeriksaan pada semua tetangga pixel, jika pixel kanannya berwarna hitam (if(bitmap[y][x+1] == 0)), maka masukkan angka 1 dalam kode rantai (stackrantai = stackrantai + "1";), tandai bitmapnya (bitmap[y][x+1] = 1;), dan lakukan pemeriksaan yang sama di pixel pada posisi tersebut (koderantai(x+1,y,bitmap);)
* Jika pixel kanan-atasnya berwarna hitam (if(bitmap[y-1][x+1] == 0)), maka masukkan angka 2 dalam kode rantai (stackrantai = stackrantai + "2";), tandai bitmapnya (bitmap[y-1][x+1] = 2;), dan lakukan pemeriksaan yang sama di pixel pada posisi tersebut (koderantai(x+1,y-1,bitmap);)
* Jika pixel atasnya berwarna hitam (if(bitmap[y-1][x] == 0)), maka masukkan angka 3 dalam kode rantai (stackrantai = stackrantai + "3";), tandai bitmapnya (bitmap[y-1][x] = 3;), dan lakukan pemeriksaan yang sama di pixel pada posisi tersebut (koderantai(x,y-1,bitmap);)
* Jika pixel kiri-atasnya berwarna hitam (if(bitmap[y-1][x-1] == 0)), maka masukkan angka 4 dalam kode rantai (stackrantai = stackrantai + "4";), tandai bitmapnya (bitmap[y-1][x-1] = 4;), dan lakukan pemeriksaan yang sama di pixel pada posisi tersebut (koderantai(x-1,y-1,bitmap);)
* Jika pixel kirinya berwarna hitam (if(bitmap[y][x-1] == 0)), maka masukkan angka 5 dalam kode rantai (stackrantai = stackrantai + "5";), tandai bitmapnya (bitmap[y][x-1] = 5;), dan lakukan pemeriksaan yang sama di pixel pada posisi tersebut (koderantai(x-1,y,bitmap);)
* Jika pixel kiri-bawahnya berwarna hitam (if(bitmap[y+1][x-1] == 0)), maka masukkan angka 6 dalam kode rantai (stackrantai = stackrantai + "6";), tandai bitmapnya (bitmap[y+1][x-1] = 6;), dan lakukan pemeriksaan yang sama di pixel pada posisi tersebut (koderantai(x-1,y+1,bitmap);)
* Jika pixel bawahnya berwarna hitam (if(bitmap[y+1][x] == 0)), maka masukkan angka 7 dalam kode rantai (stackrantai = stackrantai + "7";), tandai bitmapnya (bitmap[y+1][x] = 7;), dan lakukan pemeriksaan yang sama di pixel pada posisi tersebut (koderantai(x,y+1,bitmap);)
* Jika pixel kanan-bawahnya berwarna hitam (if(bitmap[y+1][x+1] == 0)), maka masukkan angka 8 dalam kode rantai (stackrantai = stackrantai + "8";), tandai bitmapnya (bitmap[y+1][x+1] = 8;), dan lakukan pemeriksaan yang sama di pixel pada posisi tersebut (koderantai(x+1,y+1,bitmap);)

1. Method paint()

Sebagian isi method ini sama dengan yang ada pada pertemuan sebelumnya. Bagian yang berbeda pada program kali ini terletak setelah proses pemisahan channel warna. Berikut bagian yang berbeda tersebut:

|  |
| --- |
| -------------------------  //isi filter tepi  setNilaiTepi(abuabu, Integer.parseInt(getParameter("tepi")));  setNilaiTepi(temp, Integer.parseInt(getParameter("tepi")));    setNilaiTepi(abuabu2, Integer.parseInt(getParameter("tepi2")));  setNilaiTepi(temp2, Integer.parseInt(getParameter("tepi2")));  //thresholding  for(int j=1;j<=tinggi;j++)  {  for(int i=1;i<=lebar;i++)  {  temp[j][i] = thresholding(abuabu[j][i],50);  temp2[j][i] = thresholding(abuabu2[j][i],50);  }  }    //Filtering  for(int j=1;j<=tinggi;j++)  {  for(int i=1;i<=lebar;i++)  {  titik[j-1][i-1] = (int) konvolusi(i, j, 3, 3, temp, K\_titik);  titik2[j-1][i-1] = (int) konvolusi(i, j, 3, 3, temp2, K\_titik);  }  }  //thresholding 2  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  titik[j][i] = thresholding(titik[j][i],50);  titik2[j][i] = thresholding(titik2[j][i],50);  }  }  //kontur (invers)  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  kontur[j][i] = 255-titik[j][i];  kontur2[j][i] = 255-titik2[j][i];  }  }  //Cetak  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  //Grayscale  g.setColor(new Color(abuabu[j+1][i+1],abuabu[j+1][i+1],abuabu[j+1][i+1]));  g.drawLine(10+lebar+50+i,20+j,10+lebar+50+i,20+j);  g.setColor(new Color(abuabu2[j+1][i+1],abuabu2[j+1][i+1],abuabu2[j+1][i+1]));  g.drawLine(10+lebar+50+i,20+j+tinggi2+50,10+lebar+50+i,20+j+tinggi2+50);    //Deteksi Titik  g.setColor(new Color(titik[j][i],titik[j][i],titik[j][i]));  g.drawLine(10+i+lebar+50+lebar+50,20+j,10+i+lebar+50+lebar+50,20+j);  g.setColor(new Color(titik2[j][i],titik2[j][i],titik2[j][i]));  g.drawLine(10+i+lebar+50+lebar+50,20+j+tinggi2+50, 10+i+lebar+50+lebar+50,20+j+tinggi2+50);    //Kontur  g.setColor(new Color(kontur[j][i],kontur[j][i],kontur[j][i]));  g.drawLine(10+i+lebar+50+lebar+50+lebar+50,20+j, 10+i+lebar+50+lebar+50+lebar+50,20+j);  g.setColor(new Color(kontur2[j][i],kontur2[j][i],kontur2[j][i]));  g.drawLine(10+i+lebar+50+lebar+50+lebar+50,20+j+tinggi2+50, 10+i+lebar+50+lebar+50+lebar+50,20+j+tinggi2+50);  }  }  //scanline & kode rantai  int i=0;  int j=0;  while(kontur[j][i] == 255 && (i < lebar && j < tinggi))  {  if(i == lebar-1)  {  j++;  i = 0;  }  else  {  i++;  }  }    stackrantai = "Start";  koderantai(i,j,kontur);  rantai1 = stackrantai + "end";  g.setColor(Color.black);  g.drawString("Posisi awal Gambar 1= x("+i+") y("+j+")",10,2\*tinggi+100);  g.drawString(rantai1,10,2\*tinggi+100+30);    //scanline & kode rantai 2  i=0;  j=0;  while(kontur2[j][i] == 255 && (i < lebar2 && j < tinggi2))  {  if(i == lebar2-1)  {  j++;  i = 0;  }  else  {  i++;  }  }  stackrantai = "Start";  koderantai(i,j,kontur2);  rantai2 = stackrantai + "end";  g.setColor(Color.black);  g.drawString("Posisi awal Gambar 2= x("+i+") y("+j+")",10,2\*tinggi+100+60);  g.drawString(rantai2,10,2\*tinggi+100+90);    if(rantai1.equals(rantai2))  {  g.setColor(Color.red);  g.drawString("Kedua gambar mempunyai bentuk yang sama",10,2\*tinggi+100+120);  }  else  {  g.setColor(Color.blue);  g.drawString("Kedua gambar mempunyai bentuk yang berbeda",10,2\*tinggi+100+120);  }  ------------------------- |

* Pertama-tama, isi nilai tepi dengan nilai yang diambil melalui parameter (setNilaiTepi(abuabu, Integer.parseInt(getParameter("tepi"))); hingga setNilaiTepi(temp2, Integer.parseInt(getParameter("tepi2")));)
* Berikutnya, lakukan thresholding dengan nilai batas ambang 50. Simpan hasilnya di variable temp
* Lalu, lakukan deteksi titik (filtering). Simpan hasilnya di variable titik
* Kemudian, thresholding sekali lagi hasil filtering tersebut dengan nilai threshold 50
* Setelah itu, carilah konturnya (dengan invers)
* Setelah selesai, cetak hasilnya
* Untuk kode rantai, pertama-tama lakukan scanline. Deklarasi iterasi mula-mula (int i=0; dan int j=0;) dan kemudian lakukan perulangan. Selama pixel yang diperiksa berwarna latar (putih) dan pemeriksaan belum sampai ke pixel terakhir (while(kontur[j][i] == 255 && (i < lebar && j < tinggi))), maka:
  + Jika satu baris sudah diperiksa seluruhnya (if(i == lebar-1)), maka ganti baris (j++;) dan ulang pemeriksaan dari kolom pertama (i = 0;)
  + Jika satu baris belum diperiksa seluruhnya, maka lanjut ke kolom selanjutnnya (i++;)
* Setelah posisi pixel kontur awal ditemukan (iterasi selesai), masukkan kata “Start” pada kode rantai sebagai tanda pembuka (stackrantai = "Start";)
* Kemudian, lakukan pembacaan kode rantai pada posisi pixel yang telah ditemukan (koderantai(i,j,kontur);)
* Setelah selesai, tambah kode rantai dengan kata “end” sebagai penanda akhir dan masukkan hasilnya ke variable rantai1 untuk kode rantai objek yang pertama (rantai1 = stackrantai + "end";)
* Berikutnya, cetak hasilnya (g.setColor(Color.black); hingga g.drawString(rantai1,10,2\*tinggi+100+30);)
* Untuk objek yang kedua, lakukan hal yang sama. Inisialisasi iterasi mula-mula (i=0; dan j=0;) kemudian lakukan perulangan. Selama nilai pixel yang diperiksa berwarna latar (putih) dan pemeriksaan belum sampai ke pixel terakhir (while(kontur2[j][i] == 255 && (i < lebar2 && j < tinggi2))), maka:
  + Jika satu baris sudah diperiksa seluruhnya (if(i == lebar2-1)), maka ganti baris (j++;) dan ulang pemeriksaan dari kolom pertama (i = 0;)
  + Jika satu baris belum diperiksa seluruhnya, maka lanjut ke kolom selanjutnnya (i++;)
* Setelah posisi pixel kontur awal ditemukan, masukkan kata “Start” pada kode rantai sebagai tanda pembuka (stackrantai = "Start";)
* Kemudian, lakukan pembacaan kode rantai pada posisi pixel yang telah ditemukan (koderantai(i,j,kontur2);)
* Setelah selesai, tambah kode rantai dengan kata “end” sebagai penanda akhir dan masukkan hasilnya ke variable rantai2 untuk kode rantai objek yang kedua (rantai2 = stackrantai + "end";)
* Berikutnya, cetak hasilnya (g.setColor(Color.black); hingga g.drawString(rantai2,10,2\*tinggi+100+90);)
* Untuk terakhir, lakukan perbandingan kedua kode rantai. Jika kode rantai objek 1 sama dengan kode rantai objek 2 (if(rantai1.equals(rantai2))), maka:
  + Tampilkan pesan bahwa kedua objek merupakan objek yang sama (g.setColor(Color.red); dan g.drawString("Kedua gambar mempunyai bentuk yang sama",10,2\*tinggi+100+120);)
  + Jika tidak sama, maka tampilkan pesan bahwa kedua objek merupakan objek yang berbeda (g.setColor(Color.blue); dan g.drawString("Kedua gambar mempunyai bentuk yang berbeda",10,2\*tinggi+100+120);)

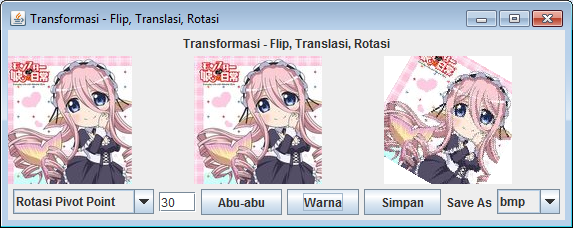
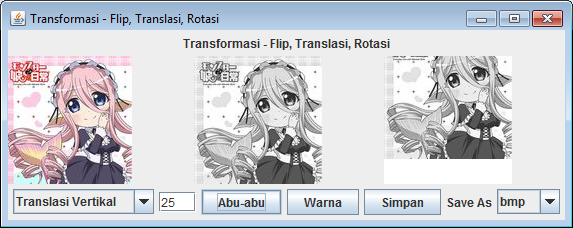
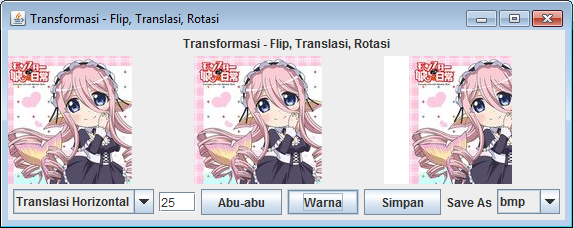
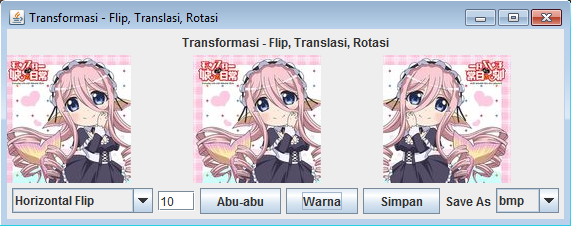
LAPORAN PRAKTIKUM PENGOLAHAN CITRA

Pertemuan XI: Transformasi – Flip, Translasi dan Rotasi

Di pertemuan ini, kita akan mempelajari untuk melakukan transformasi terhadap citra yang kita miliki. Transformasi tersebut bukan kita terapkan pada “objek” di dalam citra tersebut, melainkan kita terapkan pada seluruh titik (pixel) pada citra tersebut: ketika suatu citra digeser, maka pixel-pixel citra tersebut akan kita pindahkan posisinya sejauh besar pergeseran; ketika suatu citra dirotasi, maka pixel-pixel citra tersebut akan kita rotasi sejauh besar sudut rotasi yang telah ditentukan dan lain sebagainya.

Kali ini, program yang akan kita buat tidak akan dijalankan sebagai applet java, melainkan dijalankan sebagai class java biasa. Program ini tetap akan menggunakan java compiler (instruksi javac) namun program tersebut tidak akan dipanggil melalui appletviewer, tetapi dipanggil melalui instruksi “java”. Selain itu, program ini akan menggunakan Graphical User Interface (GUI) sederhana beserta berbagai kontrol umum (common controls) yang digunakan sebagai komponen-komponen GUI tersebut. Karena instruksi program untuk menciptakan common controls beserta inisialisasinya cukup banyak, maka pada laporan kali ini saya hanya akan menampilkan kode program yang berkaitan langsung dengan proses transformasi citra, sebagai inti dari laporan kali ini.

Berikut hasil dari praktik pada pertemuan kali ini:



**Gambar 11.1. Hasil Praktik Pertemuan 11 – Transformasi**

Sesuai dengan gambar diatas, pada program kali ini, kita akan mengambil sebuah gambar (dari parameter program) lalu kemudian, sesuai dengan pilihan user pada combo box yang ada, program akan melakukan fungsi-fungsi transformasi terhadap citra abu-abu atau warna sesuai dengan tombol yang ditekan oleh user. Besar translasi serta sudut diambil dari text box yang ada pada form tersebut. Terdapat pula sebuah tombol simpan yang akan menyimpan citra hasil transformasi dalam format yang dipilih oleh user dalam combo box.

Program di atas menggunakan dua buah class, yaitu class Gambar dan class Transformasi. Program utama kita akan kita tulis dalam class Transformasi, termasuk deklarasi variable-variable yang dibutuhkannya. Berikut variable-variable (global) yang dideklarasikan di dalam class Transformasi tersebut:

|  |
| --- |
| -------------------------  int lebar, tinggi;  int warna, red, green, blue, alpha, abuabu;  String tampil;  int m,n; //Ukuran kernel m x n  double kernelMean[][];  Color bitmap[][];  int bitmapA[][];  int bitmapR[][];  int bitmapG[][];  int bitmapB[][];  Color resultBitmap[][];  int resultBitmapA[][];  int resultBitmapR[][];  int resultBitmapG[][];  int resultBitmapB[][];  static int nilaiTepi = 0;  static int theta = 0;  int xp,yp,xn,yn;  ------------------------- |

* Variable lebar dan tinggi (int lebar, tinggi;) berfungsi untuk menyimpan lebar dan tinggi citra mula-mula
* Variable warna, red, green, blue, alpha dan abuabu berfungsi untuk memisahkan warna (int warna, red, green, blue, alpha, abuabu;)
* Variable tampil (String tampil;) digunakan untuk menampilkan string dalam kanvas/layar
* Variable m dan n (int m,n;) digunakan untuk menampung ukuran kernel (tidak digunakan dalam program ini)
* Array kernelMean (double kernelMean[][];) digunakan untuk menampung filter (juga tidak digunakan)
* Array bitmap (Color bitmap[][];) digunakan untuk menampung bitmap citra, namun hasil yang disimpan dalam array ini merupakan campuran dari channel alpha, red, green dan blue
* Array bitmapA/R/G/B (int bitmapA[][];,int bitmapR[][];, int bitmapG[][];, int bitmapB[][];) digunakan untuk menampung bitmap channel warna abu-abu, merah, hijau dan biru
* Array resultBitmap (Color resultBitmap[][];) digunakan untuk menampung bitmap citra hasil, namun hasil yang disimpan dalam array ini merupakan campuran dari channel alpha, red, green dan blue
* Array resultBitmapA/R/G/B (int resultBitmapA[][];, int resultBitmapR[][];, int resultBitmapG[][];, int resultBitmapB[][];) digunakan untuk menampung bitmap hasil channel warna abu-abu, merah, hijau dan biru
* Variable nilaiTepi (static int nilaiTepi = 0;) digunakan untuk menampung nilai tepi
* Variable theta (static int theta = 0;) digunakan untuk menampung sudut rotasi citra
* Variable xp dan yp digunakan untuk menampung titik pusat citra, sementara xn dan yn akan digunakan untuk menampung posisi pixel hasil rotasi (int xp,yp,xn,yn;)

Selanjutnya, sebagian variable tersebut digunakan/diinisialisasi di dalam konstruktor class. Berikut potongan dari isi konstruktor tersebut:

|  |
| --- |
| -------------------------  bitmap = new Color[lebar][tinggi];  bitmapA = new int[lebar][tinggi];  bitmapR = new int[lebar][tinggi];  bitmapG = new int[lebar][tinggi];  bitmapB = new int[lebar][tinggi];  resultBitmap = new Color[lebar][tinggi];  resultBitmapA = new int[lebar][tinggi];  resultBitmapR = new int[lebar][tinggi];  resultBitmapG = new int[lebar][tinggi];  resultBitmapB = new int[lebar][tinggi];  bitmap = panelGbrAsli.bitmap;  //memisahkan warna untuk diproses  for(int y=0;y<tinggi;y++)  {  for(int x=0;x<lebar;x++)  {  warna = bitmap[x][y].getRGB();  alpha = (warna >> 24) & 0xff;  red = (warna >> 16) & 0xff;  green = (warna >> 8) & 0xff;  blue = (warna) & 0xff;  abuabu = (red+green+blue) / 3;  //isi bitmap dengan nilai intensitasnya  bitmapA[x][y] = abuabu;  bitmapR[x][y] = red;  bitmapG[x][y] = green;  bitmapB[x][y] = blue;  }  }  ------------------------- |

* Pertama-tama, deklarasikan ukuran semua bitmap sesuai dengan lebar dan tinggi citra (bitmap = new Color[lebar][tinggi]; hingga resultBitmapB = new int[lebar][tinggi];)
* Masukkan citra asli yang telah disimpan dalam panelGbrAsli ke dalam bitmap (bitmap = panelGbrAsli.bitmap;)
* Selanjutnya, lakukan pemisahan warna. Dari kiri-atas hingga bawah-kanan citra (for(int y=0;y<tinggi;y++), for(int x=0;x<lebar;x++)), lakukan:
  + Ambil nilai warna dari bitmap (warna = bitmap[x][y].getRGB();)
  + Pisahkan nilai alpha, red, green dan blue-nya serta cari nilai abu-abunya (alpha = (warna >> 24) & 0xff; hingga abuabu = (red+green+blue) / 3;)
  + Lalu, isi bitmap sesuai dengan nilai intensitasnya (bitmapA[x][y] = abuabu; hingga bitmapB[x][y] = blue;)

Setelah pemisahan warna selesai dilakukan, selanjutnya adalah memproses bitmap tersebut. Pemrosesan (transformasi) bitmap dilaksanakan di dalam event yang dimiliki oleh button click (actionPerformed()). Berikut isi dari event method tersebut:

|  |
| --- |
| public void actionPerformed(ActionEvent evt)  {  Object buttonId = evt.getSource();  //inisialisasi bitmap hasil dengan putih  for(int y=0;y<tinggi;y++)  {  for(int x=0;x<lebar;x++)  {  resultBitmapA[x][y] = 255;  resultBitmapR[x][y] = 255;  resultBitmapG[x][y] = 255;  resultBitmapB[x][y] = 255;  }  }    int operasi = comboxOperasi.getSelectedIndex();    if(operasi == 0)//Normal  {  for(int y=0;y<tinggi;y++)  {  for(int x=0;x<lebar;x++)  {  resultBitmapA[x][y] = bitmapA[x][y];  resultBitmapR[x][y] = bitmapR[x][y];  resultBitmapG[x][y] = bitmapG[x][y];  resultBitmapB[x][y] = bitmapB[x][y];  }  }  }  else if(operasi == 1)//Flip Horizontal  {  for(int y=0;y<tinggi;y++)  {  for(int x=0;x<lebar;x++)  {  resultBitmapA[x][y] = bitmapA[lebar-1-x][y];  resultBitmapR[x][y] = bitmapR[lebar-1-x][y];  resultBitmapG[x][y] = bitmapG[lebar-1-x][y];  resultBitmapB[x][y] = bitmapB[lebar-1-x][y];  }  }  }  else if(operasi == 2)//Flip Vertikal  {  for(int y=0;y<tinggi;y++)  {  for(int x=0;x<lebar;x++)  {  resultBitmapA[x][y] = bitmapA[x][tinggi-1-y];  resultBitmapR[x][y] = bitmapR[x][tinggi-1-y];  resultBitmapG[x][y] = bitmapG[x][tinggi-1-y];  resultBitmapB[x][y] = bitmapB[x][tinggi-1-y];  }  }  }  else if(operasi == 3)//Geser Horizontal  {  int nilaiGeser = Integer.parseInt(this.edittextNilai.getText());  for(int y=0;y<tinggi;y++)  {  for(int x=0;x<lebar;x++)  {  if(x-nilaiGeser < lebar && x-nilaiGeser >= 0)  {  resultBitmapA[x][y] = bitmapA[x-nilaiGeser][y];  resultBitmapR[x][y] = bitmapR[x-nilaiGeser][y];  resultBitmapG[x][y] = bitmapG[x-nilaiGeser][y];  resultBitmapB[x][y] = bitmapB[x-nilaiGeser][y];  }  }  }  }  else if(operasi == 4)//Geser Vertikal  {  int nilaiGeser = Integer.parseInt(this.edittextNilai.getText());  for(int y=0;y<tinggi;y++)  {  for(int x=0;x<lebar;x++)  {  if(y+nilaiGeser < tinggi && y+nilaiGeser >= 0)  {  resultBitmapA[x][y] = bitmapA[x][y+nilaiGeser];  resultBitmapR[x][y] = bitmapR[x][y+nilaiGeser];  resultBitmapG[x][y] = bitmapG[x][y+nilaiGeser];  resultBitmapB[x][y] = bitmapB[x][y+nilaiGeser];  }  }  }  }  else if(operasi == 5)//Rotasi Pivot Point  {  theta = Integer.parseInt(this.edittextNilai.getText());  double angle = theta \* Math.PI / 180;  for(int y=0;y<tinggi;y++)  {  for(int x=0;x<lebar;x++)  {    xn = (int)Math.round(x\*Math.cos(angle)+y\*Math.sin(angle)+xp\*(1-Math.cos(angle))-yp\*Math.sin(angle));  yn = (int)Math.round(-x\*Math.sin(angle)+y\*Math.cos(angle)+yp\*(1-Math.cos(angle))+xp\*Math.sin(angle));  if(xn < lebar && xn >= 0 && yn < tinggi && yn >= 0)  {  resultBitmapA[x][y] = bitmapA[xn][yn];  resultBitmapR[x][y] = bitmapR[xn][yn];  resultBitmapG[x][y] = bitmapG[xn][yn];  resultBitmapB[x][y] = bitmapB[xn][yn];  }  }  }  }    if(buttonId == tombolAbuAbu)  {  //Proses filtering gambar Abu-abu  for(int y=0;y<tinggi;y++)  {  for(int x=0;x<lebar;x++)  {  bitmap[x][y] = new Color(bitmapA[x][y],bitmapA[x][y],bitmapA[x][y]);  resultBitmap[x][y] = new Color(resultBitmapA[x][y],resultBitmapA[x][y],resultBitmapA[x][y]);  }  }    panelGbrSumber.isiGambar(bitmap,lebar,tinggi);  panelGbrHasil.bitmap = resultBitmap;  panelGbrHasil.lebar = lebar;  panelGbrHasil.tinggi = tinggi;  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  panelGbrSumber.repaint(i,j,1,1);  }  }  panelGbrHasil.repaint();  modelWarna = "ABUABU";  }  else if(buttonId == tombolWarna)  {  for(int y=0;y<tinggi;y++)  {  for(int x=0;x<lebar;x++)  {  bitmap[x][y] = new Color(bitmapR[x][y],bitmapG[x][y],bitmapB[x][y]);  resultBitmap[x][y] = new Color(resultBitmapR[x][y],resultBitmapG[x][y],resultBitmapB[x][y]);  }  }  panelGbrSumber.isiGambar(bitmap,lebar,tinggi);  panelGbrHasil.isiGambar(resultBitmap,lebar,tinggi);  panelGbrSumber.repaint();  panelGbrHasil.repaint();  modelWarna = "WARNA";  }  -------------------------  } |

* Ketika event ini dijalankan, mula-mula cari id komponen (button) yang memanggil event ini (Object buttonId = evt.getSource();)
* Selanjutnya, inisialisasi bitmap dengan warna putih (for(int y=0;y<tinggi;y++), for(int x=0;x<lebar;x++), resultBitmapA/R/G/B[x][y] = 255;)
* Kemudian, ambil index combo box operasi yang dipilih oleh user (int operasi = comboxOperasi.getSelectedIndex();)
* Jika user memilih operasi Normal/tanpa operasi (if(operasi == 0)), maka masukkan nilai bitmapA/R/G/B ke dalam resultBitmapA/R/G/B (for(int y=0;y<tinggi;y++), for(int x=0;x<lebar;x++), resultBitmapA/R/G/B[x][y] = bitmapA/R/G/B[x][y];)
* Jika user memilih operasi Flip Horizontal (else if(operasi == 1)), maka tukar posisi kiri-kanan masing-masing pixel dalam bitmap citra (for(int y=0;y<tinggi;y++), for(int x=0;x<lebar;x++), resultBitmapA/R/G/B[x][y] = bitmapA/R/G/B[lebar-1-x][y];)
* Jika user memilih operasi Flip Vertikal (else if(operasi == 2)), maka tukar posisi atas-bawah masing-masing pixel dalam bitmap citra (for(int y=0;y<tinggi;y++), for(int x=0;x<lebar;x++), resultBitmapA/R/G/B[x][y] = bitmapA/R/G/B[x][tinggi-1-y];)
* Jika user memilih operasi Geser Horizontal (else if(operasi == 3)), maka:
  + Ambil nilai pergeseran dari text box dan masukkan ke variable (int nilaiGeser = Integer.parseInt(this.edittextNilai.getText());)
  + Untuk semua pixel bitmap (for(int y=0;y<tinggi;y++), for(int x=0;x<lebar;x++)), lakukan:
    - Jika nilai x setelah dikurangi/ditambahkan dengan nilaiGeser diatas 0 dan dibawah lebar (if(x-nilaiGeser < lebar && x-nilaiGeser >= 0)), maka ambil pixel pada posisi setelah pergeseran dan masukkan ke resultBitmap (resultBitmapA/R/G/B[x][y] = bitmapA/R/G/B[x-nilaiGeser][y];)
* Jika user memilih operasi Geser Vertikal (else if(operasi == 4)), maka:
  + Ambil nilai pergeseran dari text box dan masukkan ke variable (int nilaiGeser = Integer.parseInt(this.edittextNilai.getText());)
  + Untuk semua pixel bitmap (for(int y=0;y<tinggi;y++), for(int x=0;x<lebar;x++)), lakukan:
    - Jika nilai y setelah dikurangi/ditambahkan dengan nilaiGeser diatas 0 dan dibawah tinggi (if(y+nilaiGeser < tinggi && y+nilaiGeser >= 0)), maka ambil pixel pada posisi setelah pergeseran dan masukkan ke resultBitmap (resultBitmapA/R/G/B[x][y] = bitmapA/R/G/B[x][y+nilaiGeser];)
* Jika user memilih operasi Rotasi Pivot Point (else if(operasi == 5)), maka:
  + Ambil nilai sudut rotasi dari text box dan masukkan ke variable theta (theta = Integer.parseInt(this.edittextNilai.getText());)
  + Konversikan ke radian dan simpan dalam variable (double angle = theta \* Math.PI / 180;)
  + Untuk semua pixel bitmap (for(int y=0;y<tinggi;y++), for(int x=0;x<lebar;x++)), lakukan:
    - Rotasikan pixel tersebut dengan menggunakan rumus dan simpan hasilnya (xn = (int)Math.round(x\*Math.cos(angle)+y\*Math.sin(angle)+xp\*(1-Math.cos(angle))-yp\*Math.sin(angle)); dan yn = (int)Math.round(-x\*Math.sin(angle)+y\*Math.cos(angle)+yp\*(1-Math.cos(angle))+xp\*Math.sin(angle));)
    - Jika xn dan yn diatas 0 dan dibawah lebar dan tinggi (if(xn < lebar && xn >= 0 && yn < tinggi && yn >= 0)), maka masukkan pixel pada posisi hasil rotasi pada resultBitmap (resultBitmapA/R/G/B[x][y] = bitmapA/R/G/B[xn][yn];)
* Selanjutnya, jika user menekan tombol abu-abu (if(buttonId == tombolAbuAbu)), maka:
  + Untuk semua pixel, masukkan warna abu-abu ke dalam bitmap dan resultBitmap (bitmap[x][y] = new Color(bitmapA[x][y],bitmapA[x][y],bitmapA[x][y]); dan resultBitmap[x][y] = new Color(resultBitmapA[x][y],resultBitmapA[x][y],resultBitmapA[x][y]);)
  + Isi panelGbrSumber dengan bitmap mula-mula (sebelum transformasi) (panelGbrSumber.isiGambar(bitmap,lebar,tinggi);)
  + Isi panelGbrHasil dengan bitmap hasil transformasi (panelGbrHasil.bitmap = resultBitmap; hingga panelGbrHasil.tinggi = tinggi;)
  + Lakukan repaint untuk semua pixel pada panel Sumber (for(int j=0;j<tinggi;j++), for(int i=0;i<lebar;i++), panelGbrSumber.repaint(i,j,1,1);) dan panel Hasil (panelGbrHasil.repaint();)
  + Tetapkan bahwa citra hasil adalah citra abu-abu (modelWarna = "ABUABU";)
* Sebaliknya, jika user menekan tombol warna (else if(buttonId == tombolWarna)), maka:
  + Untuk semua pixel, masukkan warna RGB ke dalam bitmap dan resultBitmap (bitmap[x][y] = new Color(bitmapR[x][y],bitmapG[x][y],bitmapB[x][y]); dan resultBitmap[x][y] = new Color(resultBitmapR[x][y],resultBitmapG[x][y],resultBitmapB[x][y]);)
  + Isi panelGbrSumber dan panelGbrHasil dengan bitmap mula-mula dan hasil transformasi (panelGbrSumber.isiGambar(bitmap,lebar,tinggi); dan panelGbrHasil.isiGambar(resultBitmap,lebar,tinggi);)
  + Repaint kedua panel (panelGbrSumber.repaint(); dan panelGbrHasil.repaint();)
  + Tetapkan bahwa citra hasil adalah citra warna (modelWarna = "WARNA";)

LAPORAN PRAKTIKUM PENGOLAHAN CITRA

Pertemuan XII: Transformasi – Scaling

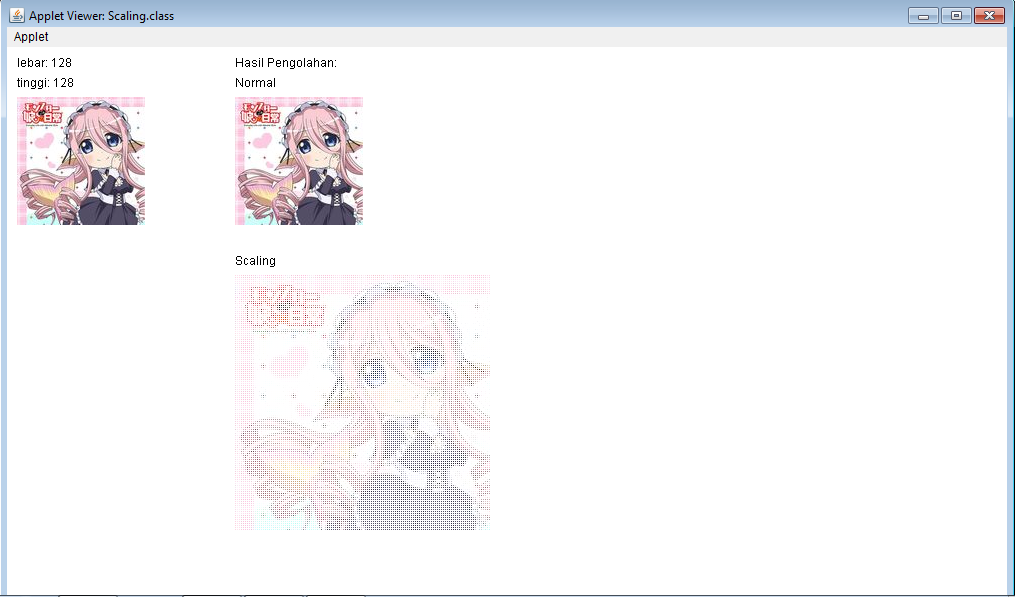
Di praktik pada pertemuan kali ini, kita masih akan menerapkan transformasi terhadap citra. Namun, kali ini kita akan fokus kepada salah satu jenis transformasi saja, yaitu scaling (mengubah ukuran) citra. Scaling pada pengolahan citra, sama seperti transformasi yang lain, juga tidak diterapkan pada sebuah “objek” yang berada di dalam citra. Scaling dilakukan kepada keseluruhan citra, atau dengan kata lain, saat kita melakukan scaling pada citra, kita juga mengubah ukuran (dimensi) dari citra tersebut. Apabila sebuah citra memiliki ukuran 10x10 pixel sebelum di scaling, maka setelah di-scaling, ukuran citra tersebut dapat berubah menjadi 5x5 (diperkecil) atau 20x20 (diperbesar) tergantung dari besar scaling yang dilakukan.

Scaling juga berkaitan dengan letak (posisi) suatu pixel di dalam citra. Apabila suatu pixel berada di posisi koordinat (1,1), maka setelah di-scaling (2x, misalnya) pixel tersebut dapat berada di posisi (2,2). Secara umum, rumus untuk menentukan posisi pixel baru hasil scaling adalah x’ = x \* sx dan y’ = y \* sy, dimana x’ dan y’ adalah posisi pixel yang baru, x dan y merupakan posisi pixel sebelum scaling, dan sx dan sy merupakan besar scaling horizontal dan vertikal.

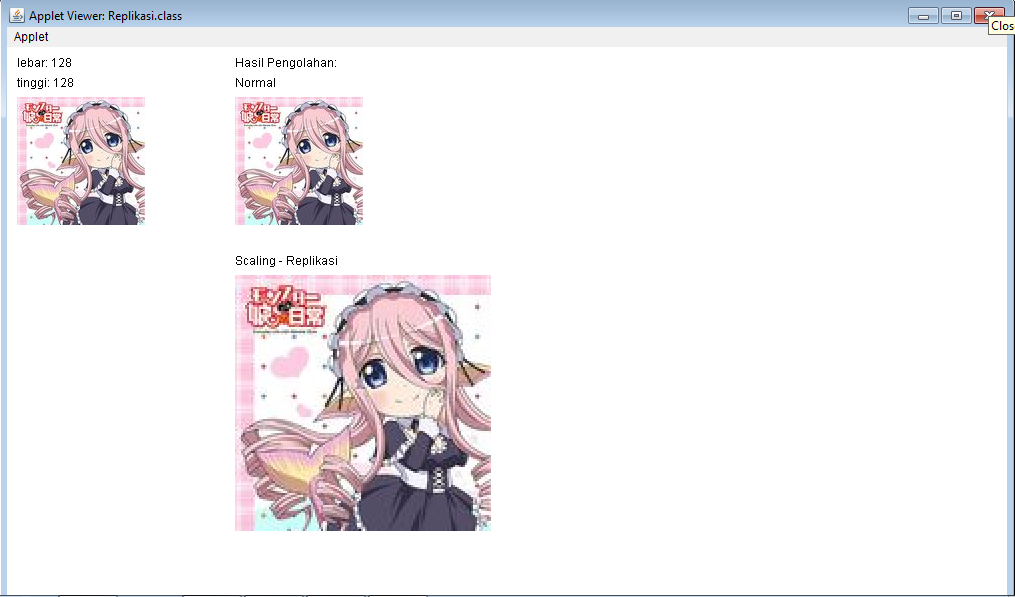
Oleh karena scaling akan mengubah ukuran citra, maka diperlukan suatu tempat (bitmap) untuk menampung citra yang baru dengan dimensi yang berbeda dari citra mula-mula. Proses scaling akan memindahkan pixel-pixel pada citra lama ke dalam bitmap baru di posisi barunya setelah scaling. Dimensi bitmap yang berbeda dari citra mula-mula menyebabkan pemetaaan pixel menjadi sulit, dimana pixel citra asal dapat berjumlah lebih banyak dari tempatnya yang tersedia pada bitmap baru (diperkecil) dan dapat pula ada posisi pada bitmap baru yang tidak terisi oleh pixel citra asal (diperbesar). Masalah pada kasus citra yang diperkecil dapat diselesaikan dengan metode *Pixel Selection*, yaitu dengan menyeleksi pixel manakah yang akan menempati posisinya pada bitmap yang baru, sementara pixel yang tidak terseleksi tidak akan dimasukkan dalam bitmap. Masalah pada kasus citra yang diperbesar dapat diselesaikan dengan berbagai metode, diantaranya adalah replikasi dan interpolasi.

Replikasi adalah metode yang mengambil pixel citra asal dan kemudian mereplikasinya (memperbanyak) untuk kemudian dimasukkan ke dalam posisi-posisi pixel yang kosong pada bitmap baru. Interpolasi merupakan metode yang mencari nilai intensitas rata-rata diantara kedua pixel (besar-kecil rata-rata dipengaruhi besar scaling) dan kemudian memasukkan nilai intensitas rata-rata tersebut untuk mengisi posisi pixel yang kosong. Pembesaran citra dengan replikasi akan menyebabkan citra baru tampil kasar (kotak-kotak), sementara pembesaran citra dengan interpolasi akan menyebabkan citra baru tampak buram (blur).

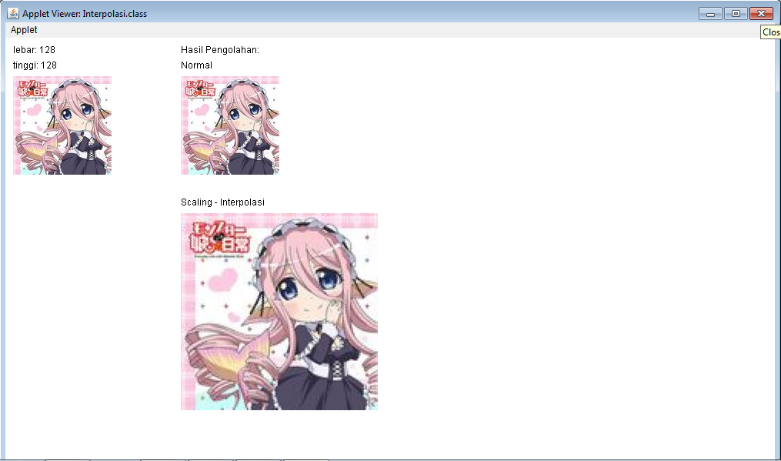
Berikut hasil dari praktik pada pertemuan kali ini:



**Gambar 12.1. Hasil Praktik Pertemuan 12 – Scaling Normal**

****

**Gambar 12.2. Hasil Praktik Pertemuan 12 – Scaling Replikasi**

****

**Gambar 12.3. Hasil Praktik Pertemuan 12 – Scaling Interpolasi**

Sesuai dengan gambar diatas, di pertemuan kali ini kita akan membuat tiga program. Program yang pertama akan melakukan scaling normal (tanpa replikasi ataupun interpolasi, oleh karena itu terdapat posisi pixel yang kosong). Program yang kedua akan melakukan scaling dengan replikasi. Program yang ketiga akan melakukan scaling dengan interpolasi. Ketiga program ini dibuat menggunakan applet java dan ditampilkan melalui appletviewer.

**Program 1: Scaling Normal**

Untuk membuat program yang pertama ini, kita perlu melakukan beberapa persiapan. Persiapan tersebut diantaranya adalah membuat dokumen html pemanggil applet, membuat sejumlah atribut (variable) global di dalam source code utama dan kemudian menginisialisasinya.

Berikut isi dari dokumen html pemanggil applet:

|  |
| --- |
| <html>  <head>  <title>RGB Color Model</title>  </head>  <body>  <applet code=Scaling.class width=1000 height=600>  <!-- applet code=Replikasi.class width=1000 height=600 -->  <!-- applet code=Interpolasi.class width=1000 height=600 -->  <param name = filegambar value = "NEOGDS-153345.jpg">  <param name = sx value = 2>  <param name = sy value = 2>  </applet>  </body>  <html> |

* Sesuai dengan isi dokumen diatas, program kali ini akan menggunakan sejumlah parameter. Parameter “filegambar” digunakan untuk menyimpan nama file citra yang akan di proses. Parameter sx digunakan untuk menampung nilai besar penskalaan horizontal. Parameter sy digunakan untuk menampung nilai besar penskalaan vertikal.

Berikut adalah sejumlah atribut (variable) global yang dibutuhkan di dalam kode program utama:

|  |
| --- |
| Image img;  int lebar, tinggi;  int warna;  String tampil;  double sx, sy; |

* Objek “img” digunakan untuk menampung file citra
* Variable “lebar” dan “tinggi” digunakan untuk menampung lebar dan tinggi citra mula-mula
* Variable “warna” digunakan untuk menampung warna pixel sebelum dipisah
* Variable string “tampil” digunakan untuk menampung string yang akan ditampilkan
* Variable double sx dan sy digunakan untuk menampung besar penskalaan horizontal dan vertikal di dalam program utama.

Setelah atribut global dibuat, selanjutnya adalah melakukan inisialisasi terhadap atribut global tersebut di dalam method init(). Berikut isi dari method tersebut:

|  |
| --- |
| public void init()  {  img=this.getImage(this.getDocumentBase(),getParameter("filegambar"));  sx = Float.parseFloat(getParameter("sx"));  sy = Float.parseFloat(getParameter("sy"));  lebar = img.getWidth(this);  } |

* Pertama-tama, ambil file citra dan masukkan ke dalam objek “img”. Nama file citra diambil dari parameter dalam dokumen html pemanggil melalui fungsi getParameter()
* Selanjutnya, ambil nilai sx dari parameter, konversikan ke float dan masukkan ke variable “sx” (sx = Float.parseFloat(getParameter("sx"));)
* Lakukan hal yang sama untuk sy (sy = Float.parseFloat(getParameter("sy"));)
* Terakhir, ambil lebar citra untuk pengetesan (lebar = img.getWidth(this);)

Setelah semua persiapan selesai dilakukan, selanjutnya adalah membuat kode program utama kita di dalam method paint(). Berikut isi dari method tersebut:

|  |
| --- |
| public void paint(Graphics g)  {  //ambil lebar dan tinggi citra  lebar = img.getWidth(this);  tinggi = img.getHeight(this);  //Tampilkan informasi lebar dan tinggi citra  g.setColor(Color.black);  tampil = String.valueOf(lebar);  g.drawString("lebar: "+tampil,10,20);  tampil = String.valueOf(tinggi);  g.drawString("tinggi: "+tampil,10,40);  g.drawImage(img,10,50,this); //Tampilkan citra dengan drawImage    int[] pixels = new int[lebar\*tinggi]; //tentukan panjang array pixels    PixelGrabber pg = new PixelGrabber(img,0,0,lebar,tinggi,pixels,0,lebar);  try  {  pg.grabPixels(); //grab piksel citra ke object pg.  }  catch (InterruptedException ie)  {  System.out.println("Terjadi kesalahan saat mengambil data pixels");  ie.printStackTrace();  return;  }    int[][] red = new int[tinggi][lebar];  int[][] green = new int[tinggi][lebar];  int[][] blue = new int[tinggi][lebar];  int[][] abuabu = new int[tinggi][lebar];  int[][] hasilA = new int[(int)Math.ceil(tinggi\*sy)][(int)Math.ceil(lebar\*sx)];  int[][] hasilR = new int[(int)Math.ceil(tinggi\*sy)][(int)Math.ceil(lebar\*sx)];  int[][] hasilG = new int[(int)Math.ceil(tinggi\*sy)][(int)Math.ceil(lebar\*sx)];  int[][] hasilB = new int[(int)Math.ceil(tinggi\*sy)][(int)Math.ceil(lebar\*sx)];    g.drawString("Hasil Pengolahan:",lebar+100,20);  g.drawString("Normal",lebar+100,40);  g.drawString("Scaling",lebar+100,tinggi+90);  //isi dengan putih  for(int j=0;j<tinggi\*sy;j++)  {  for(int i=0;i<lebar\*sx;i++)  {  hasilA[j][i] = 255;  hasilR[j][i] = 255;  hasilG[j][i] = 255;  hasilB[j][i] = 255;  }  }    for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  warna = pixels[j\*lebar+i];  red[j][i] = (warna >> 16) & 0xff;  green[j][i] = (warna >> 8) & 0xff;  blue[j][i] = (warna) & 0xff;  abuabu[j][i] = (red[j][i] + green[j][i] + blue[j][i]) / 3;  }  }  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  hasilA[(int)(j\*sy)][(int)(i\*sx)] = abuabu[j][i];  hasilR[(int)(j\*sy)][(int)(i\*sx)] = red[j][i];  hasilG[(int)(j\*sy)][(int)(i\*sx)] = green[j][i];  hasilB[(int)(j\*sy)][(int)(i\*sx)] = blue[j][i];  }  }    //Tampilkan hasil proses  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  g.setColor(new Color(red[j][i],green[j][i],blue[j][i]));  g.drawLine(lebar+100+i,50+j,lebar+100+i,50+j);  }  }  for(int j=0;j<tinggi\*sy;j++)  {  for(int i=0;i<lebar\*sx;i++)  {  g.setColor(new Color(hasilR[j][i],hasilG[j][i],hasilB[j][i]));  g.drawLine(lebar+100+i,100+tinggi+j,lebar+100+i,100+tinggi+j);  }  }  } |

* Pertama-tama, ambil lebar dan tinggi citra (lebar = img.getWidth(this); dan tinggi = img.getHeight(this);)
* Selanjutnya, tampilkan informasi lebar dan tinggi citra (g.setColor(Color.black); hingga g.drawString("tinggi: "+tampil,10,40);)
* Kemudian, tampilkan citra mula-mula dengan method drawImage() (g.drawImage(img,10,50,this);)
* Berikutnya, deklarasikan array untuk menampung pixel citra (int[] pixels = new int[lebar\*tinggi];)
* Lakukan grab pixel (menggunakan PixelGrabber)
* Untuk selanjutnya, deklarasikan bitmap untuk menampung channel warna citra mula-mula (int[][] red = new int[tinggi][lebar]; hingga int[][] abuabu = new int[tinggi][lebar];)
* Deklarasikan pula bitmap untuk menampung channel warna citra setelah scaling (int[][] hasilA = new int[(int)Math.ceil(tinggi\*sy)][(int)Math.ceil(lebar\*sx)]; hingga int[][] hasilB = new int[(int)Math.ceil(tinggi\*sy)][(int)Math.ceil(lebar\*sx)];). Ukuran bitmap citra setelah scaling tergantung dari besar penskalaannya, oleh karena itu ukuran bitmap citra hasil adalah lebar\*sx dan tinggi\*sy. Method Math.ceil() berfungsi untuk selalu membulatkan nilai hasil perkalian ke atas (sehingga bitmap tidak mungkin terlalu kecil). Karena hasil perkalian tersebut bertipe double dan index array hanya mampu menerima integer, maka hasil perkalian perlu untuk dilakukan casting ke tipe integer.
* Setelah deklarasi selesai, buat header (judul) untuk masing-masing citra (g.drawString("Hasil Pengolahan:",lebar+100,20); hingga g.drawString("Scaling",lebar+100,tinggi+90);)
* Selanjutnya, isi bitmap hasil dengan warna putih (for(int j=0;j<tinggi\*sy;j++), for(int i=0;i<lebar\*sx;i++), hasilA[j][i] = 255; hingga hasilB[j][i] = 255;)
* Setelah itu, pisahkan masing-masing channel warna pada citra (for(int j=0;j<tinggi;j++), for(int i=0;i<lebar;i++), warna = pixels[j\*lebar+i]; hingga abuabu[j][i] = (red[j][i] + green[j][i] + blue[j][i]) / 3;)
* Berikutnya, lakukan scaling. Dari kiri-atas hingga kanan-bawah citra (for(int j=0;j<tinggi;j++), for(int i=0;i<lebar;i++)), lakukan:
  + Tempatkan pixel citra abu-abu awal di posisinya yang baru setelah scaling dalam bitmap hasil (hasilA[(int)(j\*sy)][(int)(i\*sx)] = abuabu[j][i];)
  + Tempatkan pixel citra channel merah awal di posisinya yang baru setelah scaling dalam bitmap hasil (hasilR[(int)(j\*sy)][(int)(i\*sx)] = red[j][i];)
  + Tempatkan pixel citra channel hijau awal di posisinya yang baru setelah scaling dalam bitmap hasil (hasilG[(int)(j\*sy)][(int)(i\*sx)] = green[j][i];)
  + Tempatkan pixel citra channel biru awal di posisinya yang baru setelah scaling dalam bitmap hasil (hasilB[(int)(j\*sy)][(int)(i\*sx)] = blue[j][i];)
* Terakhir, tampilkan hasil prosesnya. Tampilkan citra mula-mula (for(int j=0;j<tinggi;j++), for(int i=0;i<lebar;i++), g.setColor(new Color(red[j][i],green[j][i],blue[j][i])); dan g.drawLine(lebar+100+i,50+j,lebar+100+i,50+j);) dan citra hasil prosesnya (for(int j=0;j<tinggi\*sy; j++), for(int i=0;i<lebar\*sx;i++), g.setColor(new Color(hasilR[j][i],hasilG[j][i], hasilB[j][i])); dan g.drawLine(lebar+100+i,100+tinggi+j,lebar+100+i,100+tinggi+j);)

**Program 2: Scaling Replikasi**

Program yang kedua ini menggunakan program yang pertama sebagai basisnya. Program yang kedua ini sebagian besar kode programnya sama dengan program yang pertama, hanya saja setelah proses scaling dilakukanlah replikasi untuk mengisi pixel bitmap hasil yang kosong. Berikut bagian replikasi (terletak setelah proses scaling) tersebut:

|  |
| --- |
| -------------------------  //Replikasi  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  for(int m=0;m<sx;m++)  {  for(int n=0;n<sy;n++)  {  hasilA[(int)(j\*sy+n)][(int)(i\*sx+m)] = abuabu[j][i];  hasilR[(int)(j\*sy+n)][(int)(i\*sx+m)] = red[j][i];  hasilG[(int)(j\*sy+n)][(int)(i\*sx+m)] = green[j][i];  hasilB[(int)(j\*sy+n)][(int)(i\*sx+m)] = blue[j][i];  }  }  }  }  ------------------------- |

* Dari kiri-atas hingga kanan-bawah bitmap (for(int j=0;j<tinggi;j++), for(int i=0;i<lebar;i++)), lakukan:
  + Untuk semua pixel kosong horizontal (for(int m=0;m<sx;m++)) dan pixel kosong vertikal (for(int n=0;n<sy;n++)), lakukan:
    - Isi pixel kosong pada bitmap hasil channel abu-abu dengan warna pixel mula-mula (hasilA[(int)(j\*sy+n)][(int)(i\*sx+m)] = abuabu[j][i];)
    - Isi pixel kosong pada bitmap hasil channel merah dengan warna pixel mula-mula (hasilR[(int)(j\*sy+n)][(int)(i\*sx+m)] = red[j][i];)
    - Isi pixel kosong pada bitmap hasil channel hijau dengan warna pixel mula-mula (hasilG[(int)(j\*sy+n)][(int)(i\*sx+m)] = green[j][i];)
    - Isi pixel kosong pada bitmap hasil channel biru dengan warna pixel mula-mula (hasilB[(int)(j\*sy+n)][(int)(i\*sx+m)] = blue[j][i];)

**Program 3: Scaling Interpolasi**

Program yang ketiga ini juga menggunakan program yang pertama sebagai basisnya. Apabila pada program yang kedua ditambahkan sejumlah instruksi untuk melakukan replikasi, maka pada program yang ketiga ini ditambahkan sejumlah instruksi untuk melakukan interpolasi. Letak instruksinya juga sama, yaitu setelah proses scaling biasa dilakukan. Proses interpolasi ini memerlukan sejumlah tambahan variable global, yakni:

|  |
| --- |
| double jarakA,jarakR,jarakG,jarakB; |

Variable diatas digunakan untuk menampung besar jarak diantara dua buah pixel. Jarak ini juga merupakan rata-rata dari selisih dua pixel mula-mula. Nilai jarak ditambahkan dengan nilai pixel mula-mula untuk mengisi pixel kosong, sehingga tidak terdapat perbedaan yang besar diantara pixel mula-mula dengan pixel kosong yang baru diisi.

Berikut bagian interpolasi dari program yang ketiga (terletak setelah proses scaling):

|  |
| --- |
| -------------------------  //Interpolasi  for(int j=0;j<tinggi;j++)  {  for(int i=0;i<lebar;i++)  {  //Horizontal  if(i+1 < lebar)  {  jarakA = (abuabu[j][i+1] - abuabu[j][i])/sx;  jarakR = (red[j][i+1] - red[j][i])/sx;  jarakG = (green[j][i+1] - green[j][i])/sx;  jarakB = (blue[j][i+1] - blue[j][i])/sx;  }  else  {  jarakA = 0;  jarakR = 0;  jarakG = 0;  jarakB = 0;  }  for(int m=1;m<sx;m++)  {  hasilA[(int)(j\*sy)][(int)(i\*sx+m)] = (int)(abuabu[j][i] + m\*jarakA);  hasilR[(int)(j\*sy)][(int)(i\*sx+m)] = (int)(red[j][i] + m\*jarakR);  hasilG[(int)(j\*sy)][(int)(i\*sx+m)] = (int)(green[j][i] + m\*jarakG);  hasilB[(int)(j\*sy)][(int)(i\*sx+m)] = (int)(blue[j][i] + m\*jarakB);  }    //Vertikal  if(j+1 < tinggi)  {  jarakA = (abuabu[j+1][i] - abuabu[j][i])/sy;  jarakR = (red[j+1][i] - red[j][i])/sy;  jarakG = (green[j+1][i] - green[j][i])/sy;  jarakB = (blue[j+1][i] - blue[j][i])/sy;  }  else  {  jarakA = 0;  jarakR = 0;  jarakG = 0;  jarakB = 0;  }  for(int m=0;m<sx;m++)  {  for(int n=1;n<sy;n++)  {  hasilA[(int)(j\*sy+n)][(int)(i\*sx+m)] = (int)(abuabu[j][i] + n\*jarakA);  hasilR[(int)(j\*sy+n)][(int)(i\*sx+m)] = (int)(red[j][i] + n\*jarakR);  hasilG[(int)(j\*sy+n)][(int)(i\*sx+m)] = (int)(green[j][i] + n\*jarakG);  hasilB[(int)(j\*sy+n)][(int)(i\*sx+m)] = (int)(blue[j][i] + n\*jarakB);  }  }  }  }  ------------------------- |

* Dari kiri-atas hingga kanan-bawah citra (for(int j=0;j<tinggi;j++), for(int i=0;i<lebar;i++)), lakukan:
* Untuk interpolasi horizontal, pertama-tama cek apakah pixel yang diperiksa sekarang bukan pixel paling kanan (if(i+1 < lebar)). Jika ya, maka:
  + Carilah rata-rata selisih antara dua pixel abu-abu (jarakA = (abuabu[j][i+1] - abuabu[j][i])/sx;)
  + Carilah rata-rata selisih antara dua pixel merah (jarakR = (red[j][i+1] - red[j][i])/sx;)
  + Carilah rata-rata selisih antara dua pixel hijau (jarakG = (green[j][i+1] - green[j][i])/sx;)
  + Carilah rata-rata selisih antara dua pixel biru (jarakB = (blue[j][i+1] - blue[j][i])/sx;)
* Jika tidak, maka tidak ada jarak antar dua pixel (jarakA = 0; hingga jarakB = 0;)
* Untuk semua pixel kosong horizontal (for(int m=1;m<sx;m++)), lakukan:
  + Tambahkan nilai pixel abu-abu mula-mula dengan besar jarak \* urutan pixel kosong (m), lalu masukkan di dalam pixel kosong (hasilA[(int)(j\*sy)][(int)(i\*sx+m)] = (int)(abuabu[j][i] + m\*jarakA);)
  + Tambahkan nilai pixel merah mula-mula dengan besar jarak \* urutan pixel kosong (m), lalu masukkan di dalam pixel kosong (hasilR[(int)(j\*sy)][(int)(i\*sx+m)] = (int)(red[j][i] + m\*jarakR);)
  + Tambahkan nilai pixel hijau mula-mula dengan besar jarak \* urutan pixel kosong (m), lalu masukkan di dalam pixel kosong (hasilG[(int)(j\*sy)][(int)(i\*sx+m)] = (int)(green[j][i] + m\*jarakG);)
  + Tambahkan nilai pixel biru mula-mula dengan besar jarak \* urutan pixel kosong (m), lalu masukkan di dalam pixel kosong (hasilB[(int)(j\*sy)][(int)(i\*sx+m)] = (int)(blue[j][i] + m\*jarakB);)
* Untuk interpolasi vertikal, pertama-tama cek apakah pixel yang diperiksa sekarang bukan pixel paling bawah (if(j+1 < tinggi)). Jika ya, maka:
  + Carilah rata-rata selisih antara dua pixel abu-abu vertikal (jarakA = (abuabu[j+1][i] - abuabu[j][i])/sy;)
  + Carilah rata-rata selisih antara dua pixel merah vertikal (jarakR = (red[j+1][i] - red[j][i])/sy;)
  + Carilah rata-rata selisih antara dua pixel hijau vertikal (jarakG = (green[j+1][i] - green[j][i])/sy;)
  + Carilah rata-rata selisih antara dua pixel biru vertikal (jarakB = (blue[j+1][i] - blue[j][i])/sy;)
* Jika tidak, maka tidak ada jarak antar dua pixel (jarakA = 0; hingga jarakB = 0;)
* Untuk semua pixel kosong horizontal (for(int m=0;m<sx;m++)) dan vertikal (for(int n=1;n<sy;n++)), lakukan:
  + Tambahkan nilai pixel abu-abu mula-mula dengan besar jarak \* urutan pixel kosong (n), lalu masukkan di dalam pixel kosong (hasilA[(int)(j\*sy+n)][(int)(i\*sx+m)] = (int)(abuabu[j][i] + n\*jarakA);)
  + Tambahkan nilai pixel merah mula-mula dengan besar jarak \* urutan pixel kosong (n), lalu masukkan di dalam pixel kosong (hasilR[(int)(j\*sy+n)][(int)(i\*sx+m)] = (int)(red[j][i] + n\*jarakR);)
  + Tambahkan nilai pixel hijau mula-mula dengan besar jarak \* urutan pixel kosong (n), lalu masukkan di dalam pixel kosong (hasilG[(int)(j\*sy+n)][(int)(i\*sx+m)] = (int)(green[j][i] + n\*jarakG);)
  + Tambahkan nilai pixel biru mula-mula dengan besar jarak \* urutan pixel kosong (n), lalu masukkan di dalam pixel kosong (hasilB[(int)(j\*sy+n)][(int)(i\*sx+m)] = (int)(blue[j][i] + n\*jarakB);)

LAPORAN PRAKTIKUM PENGOLAHAN CITRA

Pertemuan XIII: Steganografi

Di pertemuan kali ini, kita mempelajari mengenai steganografi, yaitu suatu ilmu yang mempelajari mengenai cara atau metode-metode untuk menyembunyikan informasi (biasanya berupa teks) di dalam sebuah gambar. Steganografi disini bukanlah berarti bahwa suatu gambar diubah tampilannya sepenuhnya untuk memasukkan informasi tertentu. Steganografi lebih fokus kepada teknik-teknik untuk memasukkan informasi di dalam sebuah citra tanpa atau dengan hanya sedikit mengubah tampilan citra tersebut secara kasat mata.

Terdapat banyak teknik-teknik dan algoritma untuk melakukan steganografi. Namun, di pertemuan kali ini, kita akan menerapkan steganografi dengan teknik yang paling sederhana. Teknik tersebut adalah dengan memecah teks informasi menjadi huruf-huruf pembentuknya, mengambil nilai bit dari setiap huruf tersebut, lalu kemudian memasukkan satu per satu rangkaian bit tersebut ke dalam bit terakhir dari pixel warna pada citra. Dengan teknik ini, informasi dapat dimasukkan ke dalam citra dengan hanya sedikit mengubah bit warna citra. Pengubahan bit terakhir hanya mengubah nilai intensitas warna pixel citra sebesar 1 poin, pengubahan tersebut terlalu kecil untuk dapat dikenali oleh mata.

Selain proses untuk menyembunyikan informasi (enkripsi), kita juga akan mempraktikkan mengenai proses untuk mengambil informasi tersembunyi tersebut (dekripsi). Secara sederhana, ekstraksi informasi dari citra yang telah disisipi informasi tersembunyi dengan teknik sebelumnya dapat dilakukan dengan membaca bit terakhir dari masing-masing pixel warna citra. Simpan rangkaian bit dari setiap bit terakhir pixel dan kemudian ubah kembali rangkaian bit tersebut menjadi teks.

Praktik pada pertemuan kali ini tidak akan menggunakan applet, melainkan menggunakan java GUI sama seperti pada pertemuan XI. Maka dari itu, praktik pada pertemuan kali ini akan mengambil sebagian kode program pada pertemuan XI, utamanya untuk membangun GUI sama seperti yang di pertemuan tersebut.

Berikut hasil dari program yang akan kita buat pada pertemuan kali ini:



**Gambar 13.1. Hasil Praktik Pertemuan 13**

Sesuai dengan gambar diatas, pada praktik kali ini, kita akan mengubah sebagian isi program pada pertemuan XI sehingga dapat melakukan proses enkripsi dan dekripsi informasi. Cara kerja program adalah sebagai berikut:

* Pertama-tama panggil program tersebut dengan mengambil sebuah file citra, file citra tersebut dapat berupa citra mentah (untuk enkripsi ke citra tersebut) atau citra yang telah disisipi informasi (untuk dekripsi);
* Selanjutnya, pilih salah satu operasi dalam combo box (enkripsi atau dekripsi):
  + Jika enkripsi, maka masukkan teks yang ingin dienkripsi ke dalam citra di dalam text box yang telah disediakan. Setelah itu, tekan salah satu tombol antara abu-abu atau warna untuk melakukan proses enkripsi dan kemudian mereproduksi citra yang telah disisipi informasi tersebut dalam kotak citra hasil (paling kanan). Setelah enkripsi dilakukan, pilih format citra untuk penyimpanan lalu simpan citra hasil tersebut dengan menekan tombol simpan. Ctt: agar informasi yang dienkripsi tidak hilang, disarankan untuk menyimpan citra hasil dalam format kompresi lossless (.PNG)
  + Jika dekripsi, maka tekan salah satu tombol antara abu-abu atau warna untuk melakukan dekripsi informasi tersembunyi. Informasi tersebut kemudian akan ditampilkan dalam text box yang telah disediakan.

Kode program pada pertemuan kali ini hanya mengubah sebagian dari kode program pada pertemuan XI, tepatnya kode program untuk class “Transformasi”. Bagian yang diubah terdapat di dalam method event button click (actionPerformed()). Bagian tersebut terletak setelah inisialisasi bitmap hasil dengan warna putih.

Berikut bagian yang berbeda tersebut:

|  |
| --- |
| -------------------------  int operasi = comboxOperasi.getSelectedIndex();  if(operasi == 0)//Enkripsi  {  kata = edittextNilai.getText();  int[] b = new int[kata.length()\*8];  //proses pengambilan nilai2 bit tiap karakter  for(int i=0;i<kata.length();i++)  {  //tiap karakter ditampung dahulu dalam sebuah variable temp  int temp=(int) kata.charAt(i);  //tiap bit di ambil dan di simpan pada array b  for(int j=0;j<8;j++)  {  b[i\*8+j] = (temp >>> (8-j-1)) & 0x1;  }  }    for(int y=0;y<tinggi;y++)  {  for(int x=0;x<lebar;x++)  {  if(y\*lebar+x < kata.length()\*8)  {  resultBitmapA[x][y] = bitmapA[x][y] & 11111110;  resultBitmapR[x][y] = bitmapR[x][y] & 11111110;  resultBitmapG[x][y] = bitmapG[x][y] & 11111110;  resultBitmapB[x][y] = bitmapB[x][y] & 11111110;    resultBitmapA[x][y] = resultBitmapA[x][y] | b[y\*lebar+x];  resultBitmapR[x][y] = resultBitmapR[x][y] | b[y\*lebar+x];  resultBitmapG[x][y] = resultBitmapG[x][y] | b[y\*lebar+x];  resultBitmapB[x][y] = resultBitmapB[x][y] | b[y\*lebar+x];  }  else if(y\*lebar+x < (kata.length()+1)\*8)  {  resultBitmapA[x][y] = bitmapA[x][y] & 11111110;  resultBitmapR[x][y] = bitmapR[x][y] & 11111110;  resultBitmapG[x][y] = bitmapG[x][y] & 11111110;  resultBitmapB[x][y] = bitmapB[x][y] & 11111110;  }  else  {  resultBitmapA[x][y] = bitmapA[x][y];  resultBitmapR[x][y] = bitmapR[x][y];  resultBitmapG[x][y] = bitmapG[x][y];  resultBitmapB[x][y] = bitmapB[x][y];  }  }  }  }  else if(operasi == 1)//Dekripsi  {  int temp = 0;  int cacah = 0;  kata = "";  char huruf;    for(int y=0;y<tinggi;y++)  {  for(int x=0;x<lebar;x++)  {  temp = temp << 1;  temp = temp | (bitmapR[x][y] & 0x1);  cacah++;  if(cacah == 8)  {  if(temp == 0x0)  {  break;  }  else  {  huruf = (char)temp;  kata = kata + huruf;  temp = 0;  cacah = 0;  }  }  }  }  edittextNilai.setText(kata);  }  ------------------------- |

* Pertama-tama, ambil operasi yang dipilih oleh user dari combo box (int operasi = comboxOperasi.getSelectedIndex();)
* Jika operasinya adalah enkripsi (if(operasi == 0)), maka:
  + Ambil teks yang akan dienkripsi dari text box dan masukkan ke variable (kata = edittextNilai.getText();)
  + Selanjutnya, deklarasikanlah sebuah array untuk menampung rangkaian bit-bit huruf dari teks tersebut. Karena sebuah huruf dalam komputer disimpan dengan ukuran 8 bit, maka ukuran dari array tersebut adalah panjang kata \* 8 (int[] b = new int[kata.length()\*8];)
  + Kemudian, untuk setiap huruf dalam teks (for(int i=0;i<kata.length();i++)), lakukan:
    - Ambil huruf tersebut, ubah ke dalam kode ASCII-nya (tipe integer) dan simpan ke dalam sebuah variable sementara (int temp=(int) kata.charAt(i);)
    - Kemudian, dari bit pertama hingga bit kedelapan huruf (for(int j=0;j<8;j++)), lakukan:
      * Geser bit huruf sejauh 8-j-1 (sehingga didapat bit huruf ke-1, 2, dst), AND-kan bit tersebut dengan 00000001 (sehingga cuma didapat bit huruf ke-1, 2, dst saja) dan kemudian simpan bit tunggal yang didapat ke dalam array yang telah dibuat sebelumnya (b[i\*8+j] = (temp >>> (8-j-1)) & 0x1;)
  + Selanjutnya, dari kiri-atas hingga kanan-bawah citra (for(int y=0;y<tinggi;y++),for(int x=0;x<lebar;x++)), lakukan:
    - Jika nilai dari posisi pixel sekarang masih lebih kecil daripada jumlah bit yang terdapat dalam teks (if(y\*lebar+x < kata.length()\*8)):
      * AND-kan pixel pada bitmap mula-mula dengan 11111110 (sehingga bit terakhir pixel tersebut menjadi 0) dan kemudian simpan pada bitmap hasil (resultBitmapA[x][y] = bitmapA[x][y] & 11111110; hingga resultBitmapB[x][y] = bitmapB[x][y] & 11111110;)
      * Lalu OR-kan pixel dengan bit terakhir 0 tersebut dengan rangkaian bit yang disimpan dalam array (resultBitmapA[x][y] = resultBitmapA[x][y] | b[y\*lebar+x]; hingga resultBitmapB[x][y] = resultBitmapB[x][y] | b[y\*lebar+x];)
    - Jika seluruh rangkaian bit dari teks telah dimasukkan dalam citra, maka untuk kedelapan bit selanjutnya (else if(y\*lebar+x < (kata.length()+1)\*8)), lakukan:
      * Kosongkan bit terakhir pixel bitmap mula-mula dan kemudian masukkan ke dalam bitmap hasil untuk penanda akhir teks (resultBitmapA[x][y] = bitmapA[x][y] & 11111110; hingga resultBitmapB[x][y] = bitmapB[x][y] & 11111110;)
    - Jika penanda akhir teks sudah ditambahkan, maka sisanya adalah tinggal memasukkan pixel dari bitmap asal ke bitmap hasil (resultBitmapA[x][y] = bitmapA[x][y]; hingga resultBitmapB[x][y] = bitmapB[x][y];)
* Jika operasinya adalah dekripsi (else if(operasi == 1)), maka:
  + Deklarasikan sejumlah variable: “temp” untuk menyimpan rangkaian 8 bit huruf (int temp = 0;), “cacah” untuk pencacah dari 1 hingga 8 (int cacah = 0;), “huruf” untuk menampung huruf yang didapat (char huruf;) dan jangan lupa kosongkan string kata (kata = "";)
  + Dari kiri-atas hingga bawah-kanan citra (for(int y=0;y<tinggi;y++), for(int x=0;x<lebar;x++)), lakukan:
    - Geser temp satu bit ke kiri (temp = temp << 1;)
    - AND-kan nilai pixel pada bitmap (misalnya bitmap channel merah) dengan 00000001 sehingga cuma tersisa bit terakhirnya saja, kemudian OR-kan dengan temp sehingga temp akan berisi rangkaian bit terakhir dari pixel pada bitmap asal (temp = temp | (bitmapR[x][y] & 0x1);)
    - Tambah nilai pencacah (cacah++;)
    - Jika temp sudah berisi rangkaian 8 bit huruf (if(cacah == 8)), maka:
      * Jika temp berisi rangkaian bit penanda akhir teks/00000000 (if(temp == 0x0)), maka keluar dari iterasi (break;)
      * Jika tidak, maka konversikan rangkaian 8 bit temp menjadi huruf dan simpan dalam variable (huruf = (char)temp;), tambahkan huruf tersebut ke dalam string (kata = kata + huruf;), kosongkan temp (temp = 0;) dan reset ulang pencacah (cacah = 0;)
  + Setelah semua proses selesai, tampilkan informasi teks dalam text box yang sudah disediakan (edittextNilai.setText(kata);)