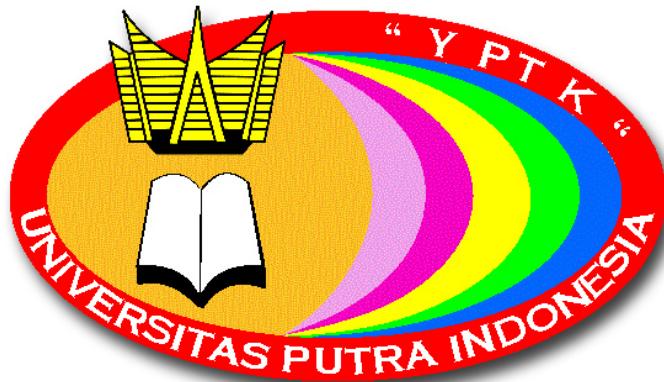


MODUL PRAKTIKUM
PENGOLAHAN CITRA DIGITAL (DIGITAL IMAGE PROCESSING)
MENGGUNAKAN MICROSOFT VISUAL STUDIO 2010 : VISUAL C#

Disusun Oleh

Dr. Ir. Sumijan, M.Sc



PROGRAM STUDI TEKNIK INFORMATIKA [S1]
FAKULTAS ILMU KOMPUTER (FILKOM)
UNIVERSITAS PUTRA INDONESIA "YPTK" PADANG

Revisi : Padang, 20 Januari 2020

Bab 1

Membaca & Menyimpan File Gambar

1.1. Dasar Teori

Pengolahan citra merupakan salah satu bidang ilmu yang mempelajari tentang proses-proses mengolah sebuah citra atau gambar. Dapat dipastikan di dalam pengolahan citra, masukannya adalah gambar, dan luarannya juga berupa gambar.



Gambar 1.1. Prinsip dasar pengolahan citra

Berdasarkan pengertian di atas, dapat dikatakan bahwa pengolahan citra mempunyai beberapa tujuan antara lain:

1. Memperbaiki kualitas gambar (Image Enhancement) seperti memperjelas dan mempertajam sebuah gambar (sharpness), mengubah warna menjadi lebih indah, mengurangi gangguan (noise) pada gambar, mengubah level terang-gelap agar gambar tampak lebih baik dan lain sebagainya.
2. Memperbaiki informasi gambar seperti mendeteksi tepi obyek gambar, mengetahui noise, memperhalus gambar dan lain sebagainya.
3. Mengklasifikasi obyek-obyek gambar seperti membagi gambar atas obyek dan background, memisah-misahkan obyek gambar dan lain sebagainya.

Ada beberapa cara untuk mendapatkan data citra/gambar agar dapat diolah menggunakan proses-proses pengolahan citra, yaitu:

1. Membaca dari file, pada modul ini menggunakan teknik pembacaan citra dari file.
2. Membaca dari kamera.
3. Membaca dari scanner.

Untuk membaca citra dari file menggunakan GDI pada Microsoft Visual C# .Net, diperlukan dua buah obyek yaitu:

1. PictureBox: tempat menampilkan hasil pembacaan dari file
2. OpenFileDialog: jendela dialog dalam memilih folder dan nama file dari file citra yang akan dibaca.

Untuk menyimpan citra ke file menggunakan GDI pada Microsoft Visual C# .Net, diperlukan dua buah obyek yaitu:

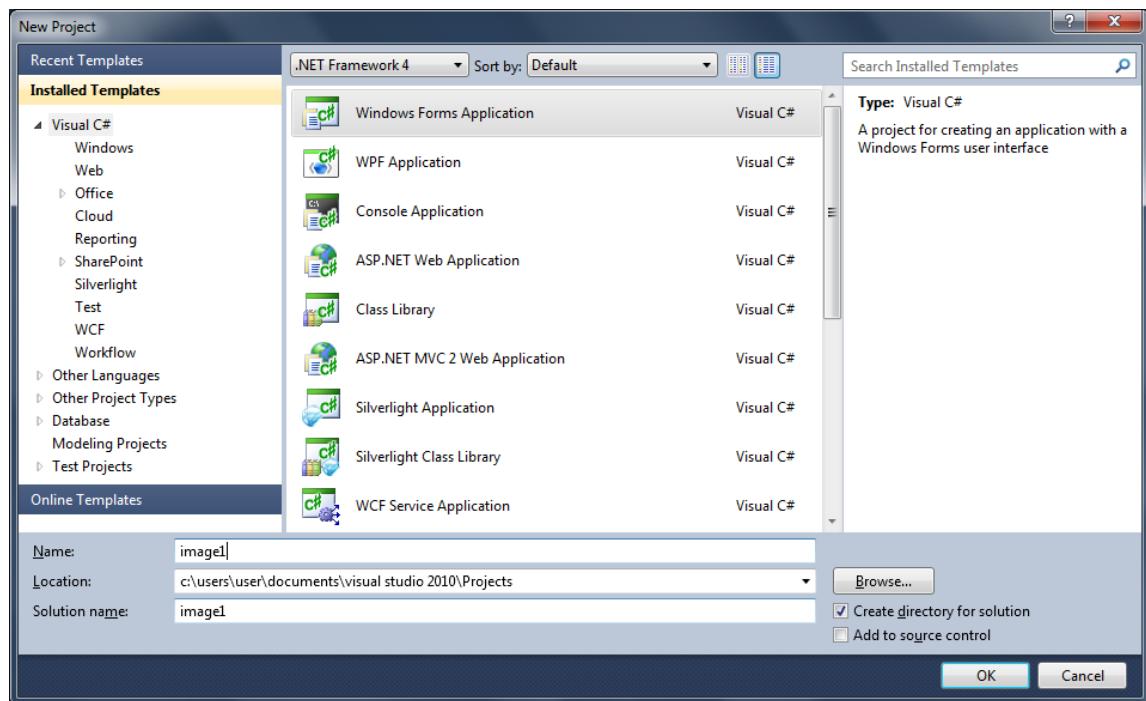
1. Picturebox: tempat tampilan citra yang akan disimpan
2. SaveFileDialog: jendela dialog dalam memilih folder dan nama file dari file citra yang disimpan.

Selain hal di atas, ada obyek yang diperlukan dalam proses membaca dan menyimpan file citra yaitu obyek **Image** yang didefinisikan secara global.

1.2. Petunjuk Praktikum

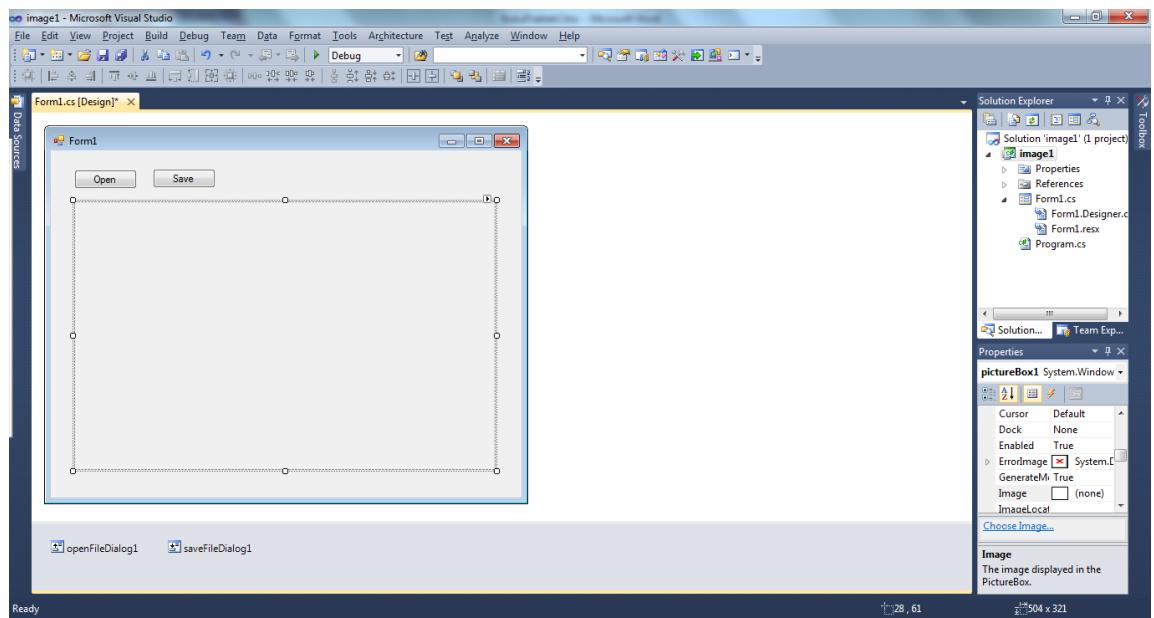
Untuk membuat aplikasi pengolahan citra menggunakan GDI pada Microsoft Visual C# .Net, ikuti langkah-langkah sebagai berikut:

1. Buka Visual Studio .NET 2010
2. Pilih File → New → New Project



Gambar 1.1. New project pada visual studio .Net 2010

3. Pilih Visual C# [Windows Form Application].
4. Beri nama “Image1” dan Solution name mengikuti dari nama, dan tekan [Ok].
5. Setelah keluar form baru dari project yang dibuat, perhatikan jendela [ToolBox].
6. Buat dua obyek button, dengan menarik komponen [Button] dari jendela toolbox ke form.
7. Pada Button1, beri text “Load Image”
8. Pada Button2, beri text “Save Image”
9. Tarik komponen PictureBox dari jendela [Toolbox] ke form. Perhatikan di sudut kanan atas dari picture pada form terdapat sebuah simbol segitiga kecil. Click segitiga tersebut, pilih Size mode dengan StretchImage. Mode ini akan mengakibatkan setiap gambar yang ditampilkan akan memenuhi ruang jendela picture.
10. Tarik komponen OpenFileDialog dari jendela [Toolbox] ke form.
11. Tarik komponen SaveFileDialog dari jendela [Toolbox] ke form.
12. Form sudah siap seperti terlihat pada gambar 1.2 berikut.



Gambar 1.2. Tampilan form

Setelah form selesai dibuat, hal berikutnya adalah mengisinya dengan kode program. Jangan khawatir, pengkodean dengan Visual Studio .Net tidak terlalu sulit untuk dilakukan.

13. Double click pada ruang kosong di dalam form, sehingga muncul layar kode berikut:

```
namespace image1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

Program diletakkan di dalam fungsi **public partial class Form1 : Form**.

14. Pertama tambahkan sebuah obyek Image dengan nama File tepat di bawah

```
class Form1.

namespace image1
{
    public partial class Form1 : Form
    {
        Image File;

        public Form1()
```

```
    {
        InitializeComponent();
    }

}
```

15. Double click button1 untuk membuat program membaca file gambar dan menampilkannya di Picturebox, pada tempat yang ditunjuk tambahkan program berikut:

```
DialogResult d = openFileDialog1.ShowDialog();
if (d == DialogResult.OK)
{
    File = Image.FromFile(openFileDialog1.FileName);
    pictureBox1.Image = File;
}
```

16. Untuk membuat program menyimpan file gambar dari apa yang ada di PictureBox ke dalam file berformat JPG, pertama tambahkan library baru di bagian atas program:

```
using System.Drawing.Imaging;
```

17. Double click pada button2, pada tempat yang ditunjuk tambahkan program berikut:

```
DialogResult d = saveFileDialog1.ShowDialog();
if (d == DialogResult.OK)
{
    File.Save(saveFileDialog1.FileName, ImageFormat.Jpeg);
}
```

Program sudah selesai. Untuk menjalankan program tekan tombol segitiga pada menu atau tekan tombol [F5] atau start debugging. Jalankan program tersebut dan coba tampilkan gambar dari file-file yang sudah tersedia di harddisk.

1.3. Laporan Praktikum

Buatlah program untuk membaca file gambar dan menampilkannya ke dalam Picturebox seperti langkah-langkah di petunjuk praktikum.

- 1) Tuliskan koding lengkap dari program di atas, dan lengkapi dengan keterangan pada setiap baris yang anda anggap penting untuk pengertian dari pembacaan dan penyimpanan file gambar.
- 2) Pada komponen PictureBox, terdapat segitiga kecil di bagian kanan atas yang digunakan untuk mengubah size-mode. Jelaskan apa perbedaan masing-masing size-mode:
 - a. Normal
 - b. StretchImage
 - c. AutoSize
 - d. CenterImage
 - e. Zoom
- 3) Jelaskan apa fungsi dari perintah `pictureBox1.Image = File;`

Bab 2

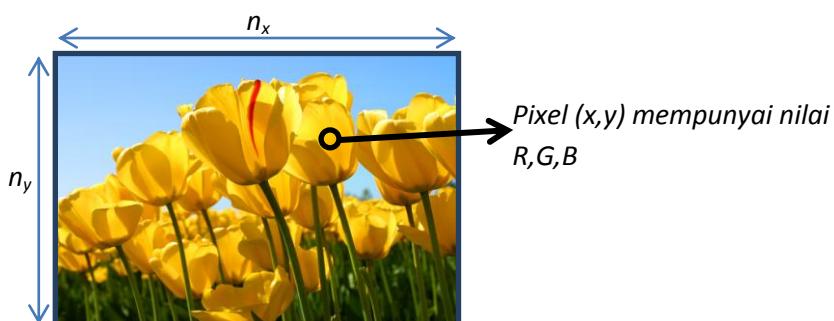
Membaca Data Gambar

2.1. Dasar Teori

Citra/gambar secara konsep data dinyatakan dalam bentuk matrik 2D atau array 2D, dimana setiap pixelnya pada posisi (x,y) dinyatakan dalam $P(x,y)$. Setiap pixel pada data gambar mempunyai nilai sesuai dengan format gambar yang digunakan:

- Pada gambar berwarna, setiap titik mempunyai nilai 24 bit RGB, dimana masing-masing komponen warna R, G dan B mempunyai nilai 8 bit, atau dengan kata lain setiap komponen warna mempunyai nilai 0 s/d 255.
- Pada gambar grayscale atau derajat keabuan, setiap titik mempunyai nilai 8 bit, atau 0 s/d 255.
- Pada gambar hitam putih (BW), setiap titik mempunyai nilai 0 atau 1.

Apapun format gambar yang digunakan, program di dalam Visual Studio .Net hanya mengenal format gambar berwarna RGB, sehingga untuk menampilkan gambar grayscale atau BW digunakan nilai yang sama untuk setiap komponen R, G dan B.



Gambar 2.1. Komponen data citra

7

Untuk bisa mengolah data citra (misalkan dengan nama objBitmap) maka sebelumnya harus didefinisikan sebuah obyek citra dalam type Bitmap.

```
Bitmap objBitmap;
```

Dalam sebuah obyek citra **objBitmap** terdapat beberapa parameter dan fungsi yang dapat digunakan yaitu:

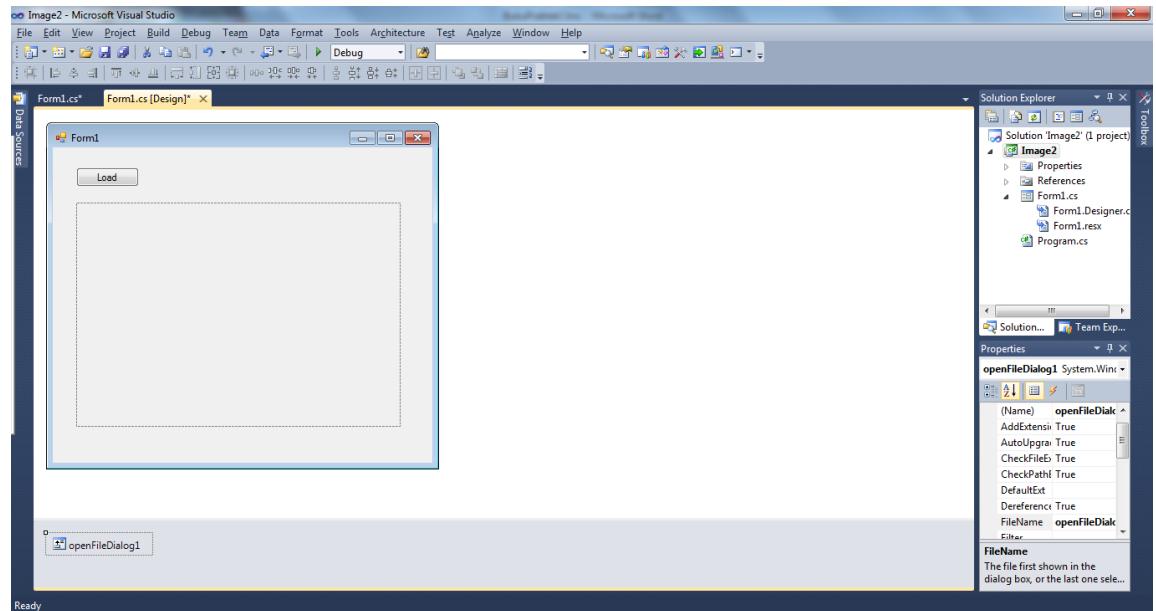
- `objBitmap.width` menyatakan panjang dari sebuah gambar dalam jumlah pixel horisontal.
- `objBitmap.height` menyatakan tinggi dari sebuah gambar dalam jumlah pixel vertikal.
- `objBitmap.getPixel(x,y)` adalah sebuah fungsi untuk membaca data RGB dari pixel (x,y) . Hasil pembacaan berupa nilai warna w yang bertipe `Color` dan menghasilkan 3 data yaitu $w.R$ untuk warna merah, $w.G$ untuk warna hijau dan $w.B$ untuk warna biru.
- `objBitmap.setPixel(x,y,w)` adalah sebuah fungsi untuk memberikan nilai warna RGB pada pixel (x,y) dalam `objBitmap`.

2.2. Petunjuk Praktikum

2.2.1. Membaca Data Gambar Dari File

Langkah-langkah berikut adalah membuat aplikasi yang dapat membaca data gambar dari file untuk bisa dibaca nilai setiap pixelnya:

- (1) Buka Visual Studio .Net 2010
- (2) Buat project baru dengan `File→Project→New Project`
- (3) Pilih `Visual C# [Windows Form Application]`.
- (4) Beri nama “Image2” dan `Solution name` mengikuti dari nama, dan tekan `[Ok]`.
- (5) Setelah keluar form baru dari project yang dibuat, pilih komponen 1 buah button, 1 buah Picturebox dan 1 buah openFileDialog.
- (6) Pada `Button1`, ubah text menjadi “Load”. Pada `Picturebox1`, atur size modenya dengan `StretchImage`.
- (7) Atur tampilannya seperti gambar 2.2 berikut:



Gambar 2.2. Layout dari project Image2

- (8) Tambahkan dua obyek baru pada class Form1 dan letakkan sebelum Public Form1() seperti berikut:

```
Bitmap objBitmap;
```

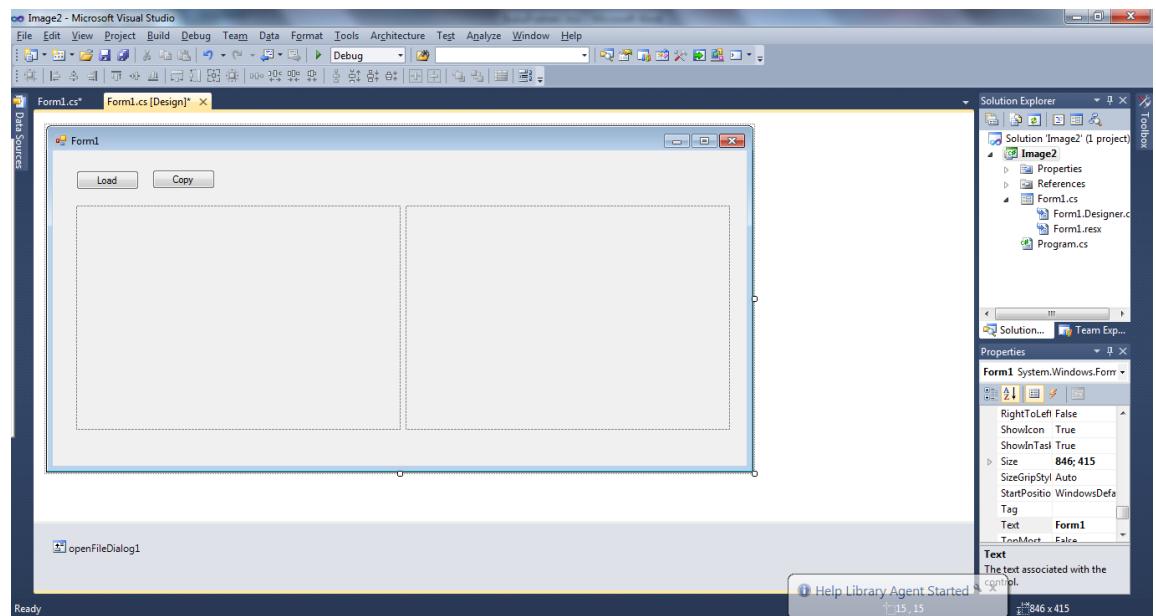
- (9) Double click pada Button1 (Load), dan tambahkan program berikut:

```
DialogResult d = openFileDialog1.ShowDialog();
if (d == DialogResult.OK)
{
    objBitmap = new Bitmap(openFileDialog1.FileName);
    pictureBox1.Image = objBitmap;
}
```

2.2.2. Membaca dan Mengcopy Data Gambar

Langkah-langkah berikut adalah membuat aplikasi yang dapat mengambil data warna setiap pixel pada gambar dengan melanjutkan project Image2 di atas.

- (1) Pada form di Project Image2 di atas, tambahkan komponen-komponen 1 buah button dan 1 buah Picturebox.
- (2) Pada Button2, beri text “Copy”.
- (3) Pada PictureBox2, atur size-modonya dengan StretchImage.
- (4) Atur tampilannya seperti gambar 2.3 berikut:



Gambar 2.3. Hasil layout form baru untuk mengambil dari warna setiap pixel

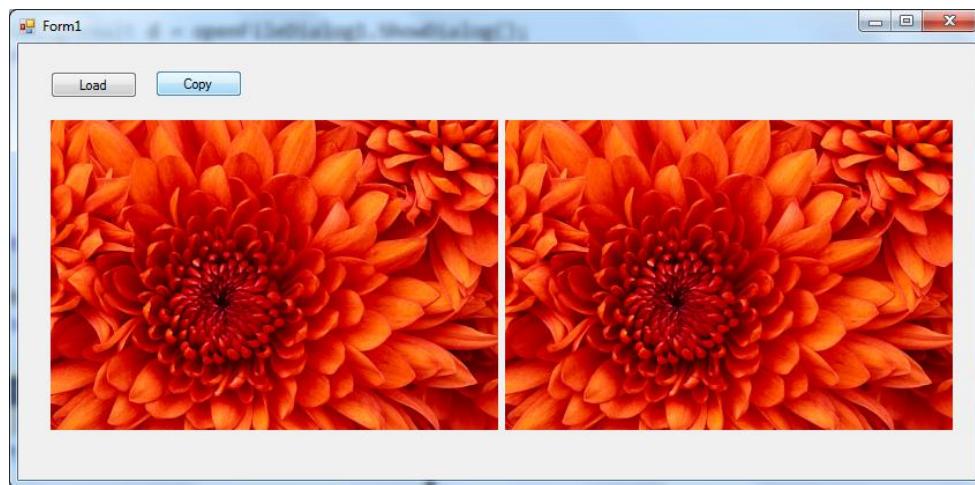
- (5) Tambahkan obyek baru objBitmap1 di Class Form1 untuk penampung data gambar, letakkan setelah **Bitmap objBitmap**;

```
Bitmap objBitmap1;
```

- (6) Double click pada Button2 (Copy) dan tambahkan program berikut:

```
objBitmap1 = new Bitmap(objBitmap);
for(int x=0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        objBitmap1.SetPixel(x, y, w);
    }
pictureBox2.Image = objBitmap1;
```

Perhatikan hasil dari program di atas. Saat kita menekan tombol Load, kita akan memilih file gambar dan disimpan dalam obyek objBitmap, dan ditampilkan di PictureBox1. Saat kita menekan tombol Copy, program akan membaca data warna pada setiap pixel gambar dari objBitmap, mencopy data tersebut pada obyek ObjBitmap1, dan menampilkannya di PictureBox2.

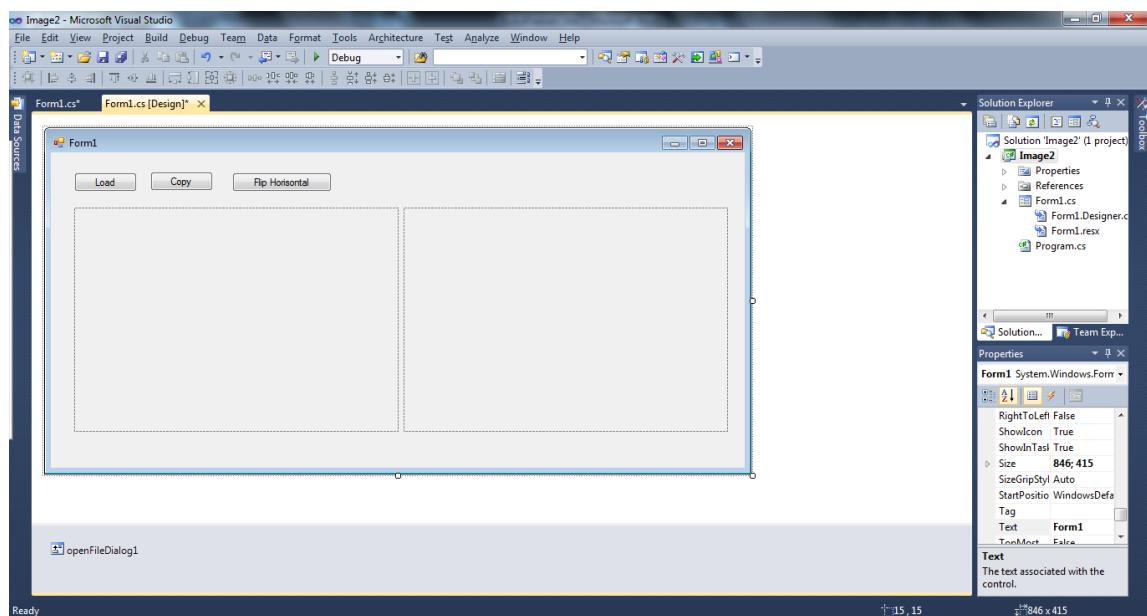


Gambar 2.4. Contoh hasil program dari mencopy data warna setiap pixel

2.2.3. Flip Horisontal

Langkah-langkah berikut adalah membuat aplikasi yang dapat mengambil data warna setiap pixel dan meletakkan secara flip horisontal dengan melanjutkan project Image2 di atas.

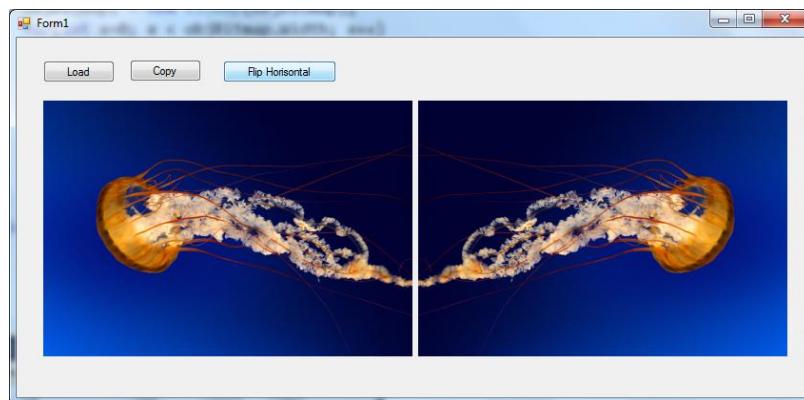
- (1) Pada form di Project Image2 di atas, tambahkan komponen button dan beri text "Flip Horisontal". Atur tampilannya seperti gambar 2.5 berikut:



Gambar 2.5. Hasil layout form baru untuk mengambil dari warna setiap pixel

(2) Double click pada Button2 (Flip Horisontal) dan tambahkan program berikut:

```
objBitmap1 = new Bitmap(objBitmap);
for(int x=0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        objBitmap1.SetPixel(objBitmap.Width-1-x, y, w);
    }
pictureBox2.Image = objBitmap1;
```



Gambar 2.6. Contoh hasil flip horisontal

Pada dasarnya proses flip horisontal adalah meletakkan titik horisontal secara berkebalikan dimana setiap pixel *x* pada gambar asal (*objBitmap*) akan diletakkan di pixel *width-(x+1)* pada gambar hasil (*objBitmap1*).

Berikutnya proses flip vertikal adalah meletakkan titik horisontal secara berkebalikan dimana setiap pixel *y* pada gambar asal (*objBitmap*) akan diletakkan di pixel *height-(y+1)* pada gambar hasil (*objBitmap1*).

2.3. Laporan Praktikum

Buat dan jalankan program di atas sesuai petunjuk.

- (1) Tuliskan kode program secara lengkap dan berikan keterangan pada perintah-perintah yang anda anggap penting.
- (2) Tambahkan program untuk **Flip Vertikal** dan tuliskan kode programnya.

Bab 3

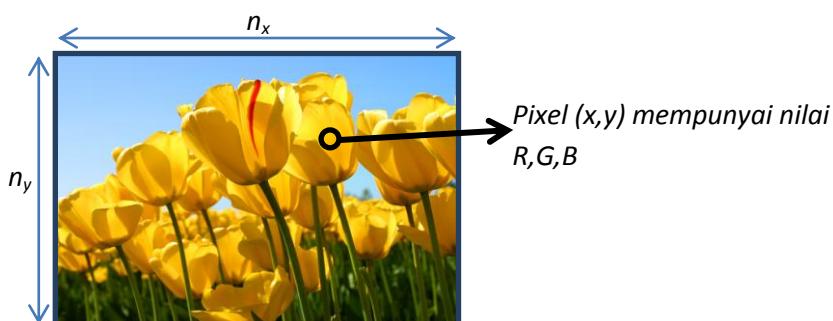
Manipulasi RGB

3.1. Dasar Teori

Citra/gambar secara konsep data dinyatakan dalam bentuk matrik 2D atau array 2D, dimana setiap pixelnya pada posisi (x,y) dinyatakan dalam $P(x,y)$. Setiap pixel pada data gambar mempunyai nilai sesuai dengan format gambar yang digunakan:

- Pada gambar berwarna, setiap titik mempunyai nilai 24 bit RGB, dimana masing-masing komponen warna R, G dan B mempunyai nilai 8 bit, atau dengan kata lain setiap komponen warna mempunyai nilai 0 s/d 255.
- Pada gambar grayscale atau derajat keabuan, setiap titik mempunyai nilai 8 bit, atau 0 s/d 255.
- Pada gambar hitam putih (BW), setiap titik mempunyai nilai 0 atau 1.

Apapun format gambar yang digunakan, program di dalam Visual Studio .Net hanya mengenal format gambar berwarna RGB, sehingga untuk menampilkan gambar grayscale atau BW digunakan nilai yang sama untuk setiap komponen R, G dan B.



Gambar 2.1. Komponen data citra

13

Untuk bisa mengolah data citra (misalkan dengan nama objBitmap) maka sebelumnya harus didefinisikan sebuah obyek citra dalam type Bitmap.

```
Bitmap objBitmap;
```

Dalam sebuah obyek citra **objBitmap** terdapat beberapa parameter dan fungsi yang dapat digunakan yaitu:

- `objBitmap.width` menyatakan panjang dari sebuah gambar dalam jumlah pixel horisontal.
- `objBitmap.height` menyatakan tinggi dari sebuah gambar dalam jumlah pixel vertikal.
- `objBitmap.getPixel(x,y)` adalah sebuah fungsi untuk membaca data RGB dari pixel (x,y) . Hasil pembacaan berupa nilai warna w yang bertipe `Color` dan menghasilkan 3 data yaitu $w.R$ untuk warna merah, $w.G$ untuk warna hijau dan $w.B$ untuk warna biru.
- `objBitmap.setPixel(x,y,w)` adalah sebuah fungsi untuk memberikan nilai warna RGB w pada pixel (x,y) dalam `objBitmap`.
- Untuk memberi nilai RGB pada warna baru wb , bisa menggunakan fungsi `Color wb = Color.FromArgb(r, g, b);`

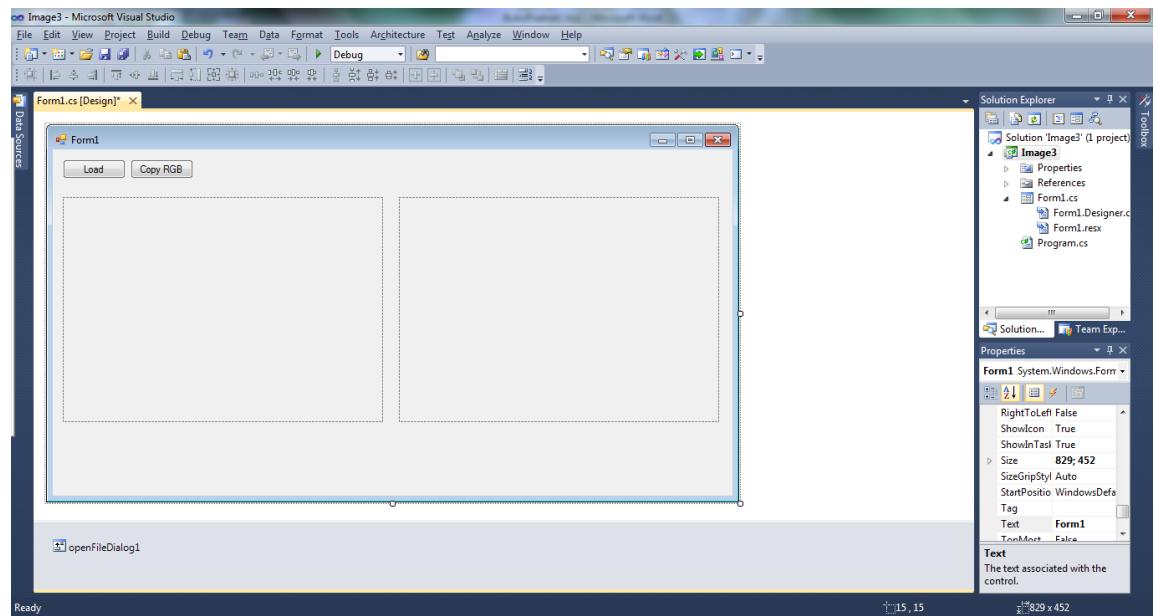
3.2. Petunjuk Praktikum

3.2.1. Mencopy warna RGB

Langkah-langkah berikut untuk memanggil file gambar dan mengambil warna RGB dari setiap pixel serta mengcoponya ke gambar lain.

- (1) Buka Visual Studio .Net 2010
- (2) Buat project baru dengan `File → Project → New Project`
- (3) Pilih `Visual C# [Windows Form Application]`.
- (4) Beri nama “Image3” dan `Solution name` mengikuti dari nama, dan tekan `[Ok]`.
- (5) Setelah keluar form baru dari project yang dibuat, pilih komponen 2 buah button, 2 buah Picturebox dan 1 buah openFileDialog. Pada `Button1`, ubah text menjadi “Load”. Pada `Button2`, ubah text menjadi “Copy RGB”. Pada

Picturebox1 dan Picturebox2, atur size modenya dengan StretchImage. Atur tampilannya seperti gambar 3.1 berikut:



Gambar 3.1. Form Image3

- (6) Tambahkan dua obyek Bitmap yaitu objBitmap dan objBitmap1 yang diletakkan sebagai obyek global dalam class From1.

```
Bitmap objBitmap;
Bitmap objBitmap1;
```

- (7) Double click pada Button1 (Load), dan tambahkan program berikut:

```
DialogResult d = openFileDialog1.ShowDialog();
if (d == DialogResult.OK)
{
    objBitmap = new Bitmap(openFileDialog1.FileName);
    pictureBox1.Image = objBitmap;
}
```

- (8) Double click pada Button2 (Copy RGB), dan tambahkan program berikut:

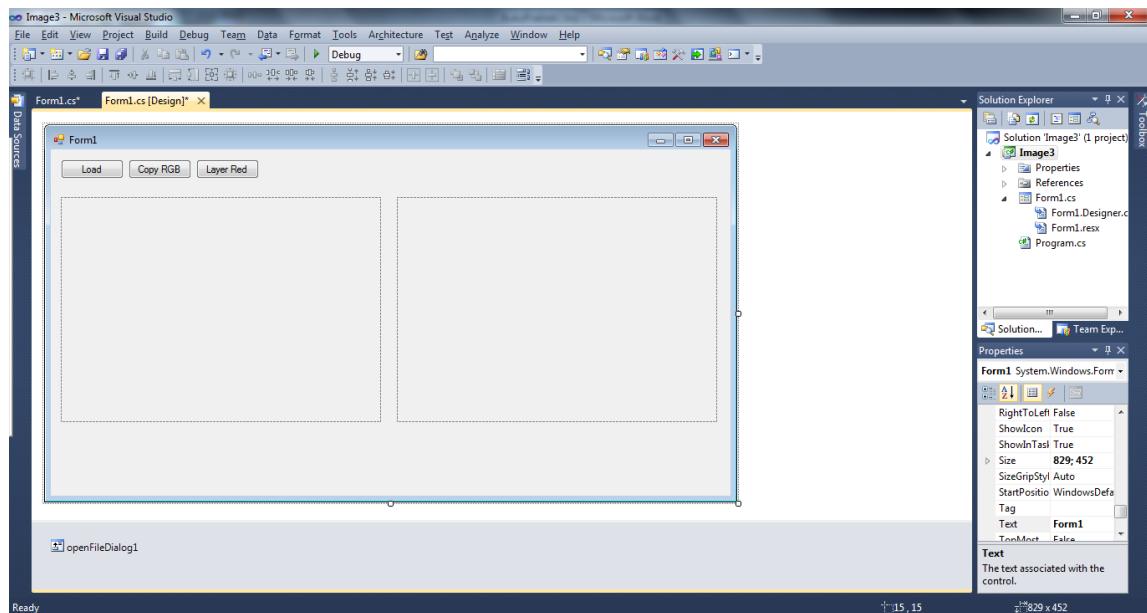
```
objBitmap1 = new Bitmap(objBitmap);
for(int x=0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap1.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int r = w.R;
        int g = w.G;
        int b = w.B;
        Color wb = Color.FromArgb(r, g, b);
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;
```

Program di atas mengambil nilai RGB dari setiap pixel (x,y), kemudian nilai RGB tersebut dijadikan warna baru wb yang kemudian dimasukkan ke obyek citra yang baru objBitmap1.

3.2.2. Menampilkan Gambar Layer Red

Program ini digunakan untuk mengambil nilai dari layer R, dan kemudian ditampilkan dengan meng-nol-kan layer G dan layer B. Langkah-langkahnya dengan meneruskan program pada Image3 adalah sebagai berikut:

- (1) Pada form Image3, tambahkan 1 buah button dan beri text “Layer Red” seperti pada gambar 3.2 berikut.



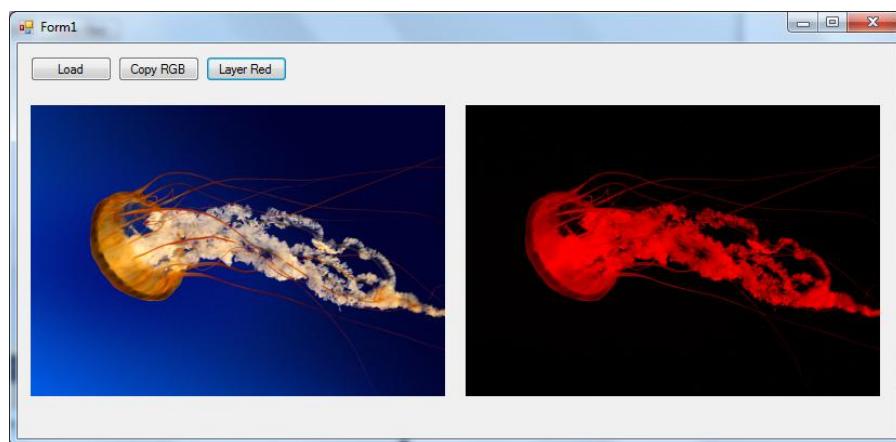
Gambar 3.2. Penambahan button untuk Layer Red

- (2) Double click pada button3 (Layer Red), dan tambahkan program berikut. Perhatikan pada fungsi mengambil warna baru wb, yang dipakai hanya nilai R, sedangkan nilai G dan B ditulis nol.

```
objBitmap1 = new Bitmap(objBitmap);
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap1.Height; y++)
```

```
{  
    Color w = objBitmap.GetPixel(x, y);  
    int r = w.R;  
    Color wb = Color.FromArgb(r, 0, 0);  
    objBitmap1.SetPixel(x, y, wb);  
}  
pictureBox2.Image = objBitmap1;
```

Hasil dari program Layer Red ini adalah seperti terlihat pada gambar 3.3 di bawah ini. Terlihat bahwa gambar hasil adalah berwarna merah. Perhatikan pada titik-titik yang berwarna biru (tidak mempunyai komponen warna merah) akan menjadi hitam.



Gambar 3.3. Hasil dari Layer Red

3.2.3. Menampilkan Gambar Layer Hijau

Program ini digunakan untuk mengambil nilai dari layer G, dan kemudian ditampilkan dengan meng-nol-kan layer R dan layer B. Langkah-langkahnya dengan meneruskan program pada Image3 adalah sebagai berikut:

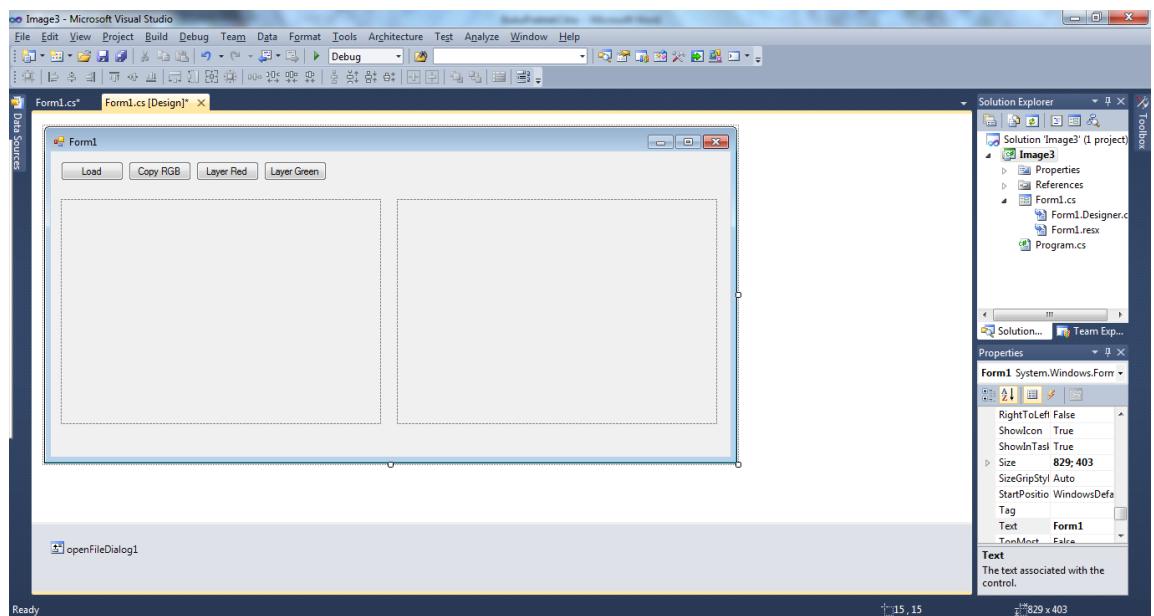
- (1) Pada form Image3, tambahkan 1 buah button dan beri text "Layer Green" seperti pada gambar 3.4 berikut.
- (2) Double click pada button4 (Layer Green), dan tambahkan program berikut. Perhatikan pada fungsi mengambil warna baru wb, yang dipakai hanya nilai G, sedangkan nilai R dan B ditulis nol.

```
objBitmap1 = new Bitmap(objBitmap);  
for (int x = 0; x < objBitmap.Width; x++)
```

```

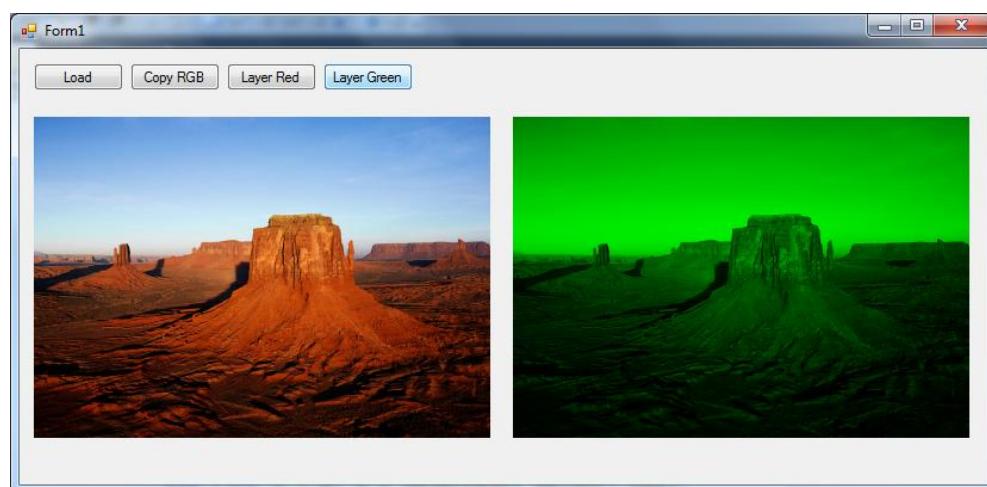
for (int y = 0; y < objBitmap1.Height; y++)
{
    Color w = objBitmap.GetPixel(x, y);
    int g = w.G;
    Color wb = Color.FromArgb(0, g, 0);
    objBitmap1.SetPixel(x, y, wb);
}
pictureBox2.Image = objBitmap1;

```



Gambar 3.4. Penambahan button untuk Layer Green

Hasil dari program Layer Green adalah mengambil dan menampilkan gambar hanya pada layer G atau komponen warna hijau. Perhatikan bahwa titik-titik berwarna merah atau biru (tidak mengandung komponen warna hijau) akan menjadi hitam.



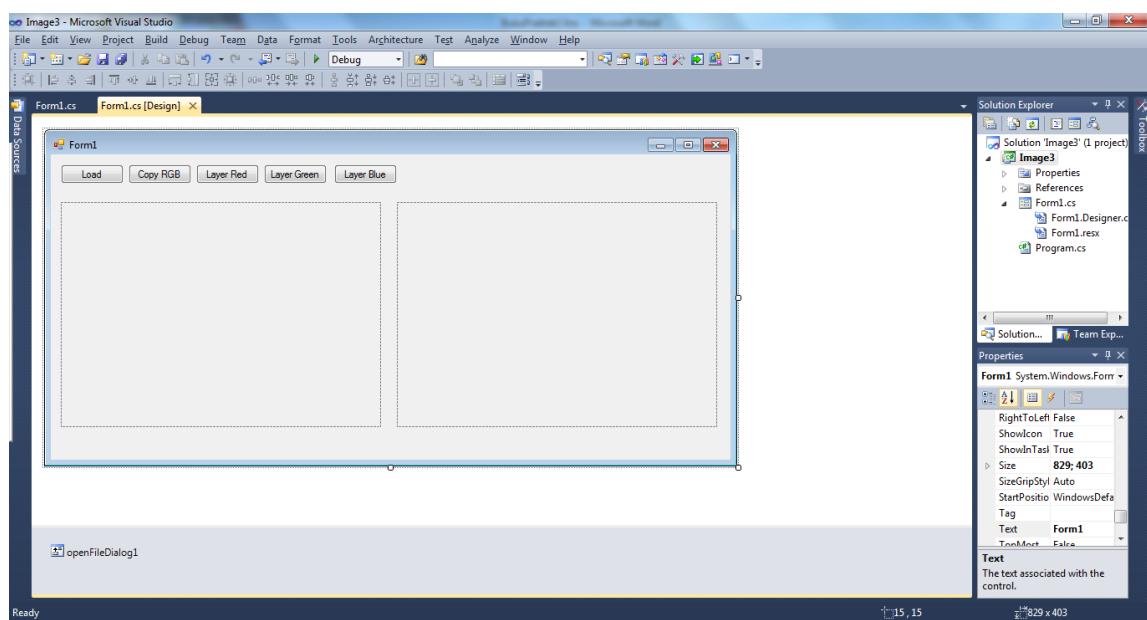
Gambar 3.5. Hasil program Layer Green

3.2.4. Menampilkan Gambar Layer Biru

Program ini digunakan untuk mengambil nilai dari layer B, dan kemudian ditampilkan dengan meng-nol-kan layer R dan layer G. Langkah-langkahnya dengan meneruskan program pada Image3 adalah sebagai berikut:

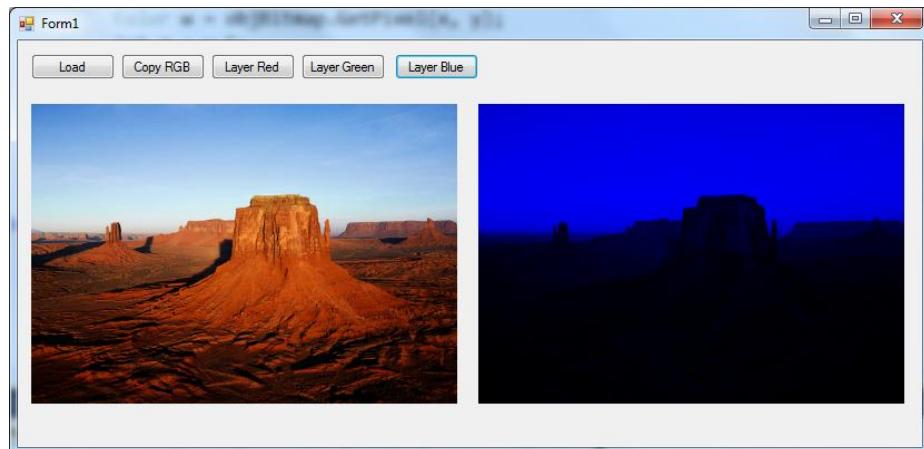
- (1) Pada form Image3, tambahkan 1 buah button dan beri text “Layer Blue” seperti pada gambar 3.6 berikut.
- (2) Double click pada button5 (Layer Blue), dan tambahkan program berikut. Perhatikan pada fungsi mengambil warna baru wb, yang dipakai hanya nilai B, sedangkan nilai R dan G ditulis nol.

```
objBitmap1 = new Bitmap(objBitmap);
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap1.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int b = w.B;
        Color wb = Color.FromArgb(0, 0, b);
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;
```



Gambar 3.6. Penambahan button untuk Layer Blue

Hasil dari program Layer Blue adalah mengambil dan menampilkan gambar hanya pada layer B atau komponen warna biru. Perhatikan bahwa titik-titik berwarna merah atau hijau (tidak mengandung komponen warna biru) akan menjadi hitam.



Gambar 3.7. Hasil program Layer Blue

3.2.5. Menampilkan Gambar Grayscale Dari Satu Layer

Pada dasarnya gambar grayscale atau derajat keabuan adalah gambar yang menggunakan nilai R, G dan B yang sama, misalkan (100,100,100) atau (140,140,140) atau (xg,xg,xg). Dengan demikian bila diambil salah satu warna r, g atau b, dan dijadikan warna baru dengan (r,r,r) atau (g,g,g) atau (b,b,b), akan juga menghasilkan gambar grayscale.

Langkah-langkah untuk menghasilkan gambar grayscale dari warna merah, hijau dan biru dengan meneruskan program pada Image3 adalah sebagai berikut:

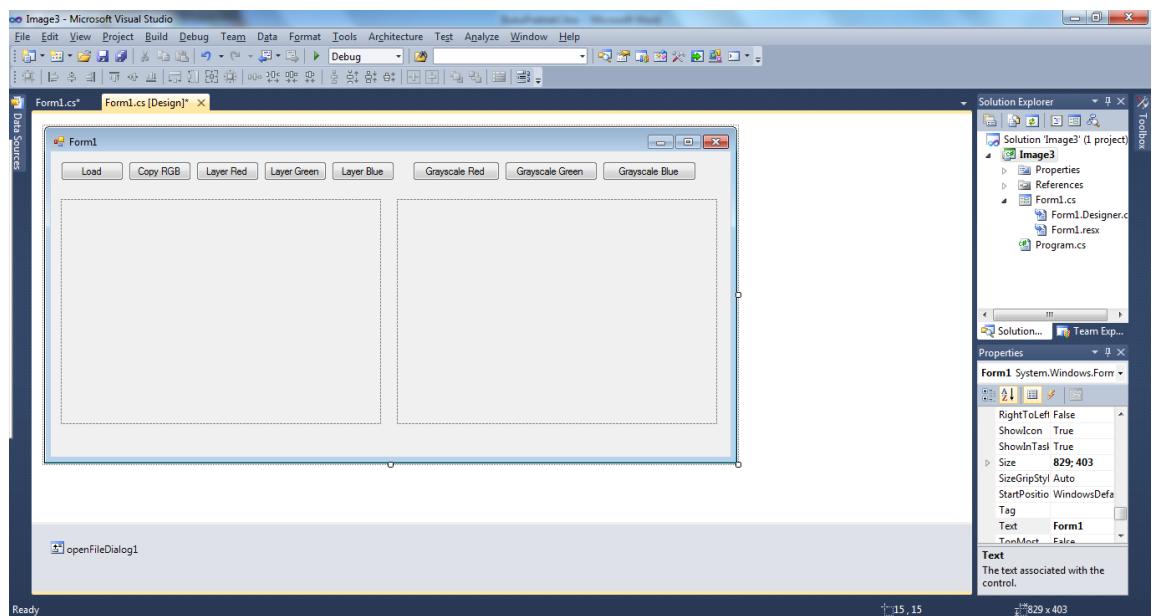
- (1) Pada form Image3, tambahkan 3 buah button dan beri text masing-masing “Grayscale Red”, “Grayscale Green” dan “Grayscale Blue” seperti pada gambar 3.8 berikut.
- (2) Double click pada button6 (Grayscale Red), dan tambahkan program berikut. Perhatikan pada fungsi mengambil warna baru wb, yang dipakai hanya nilai R, dengan menyebutkan (r,r,r).

```
objBitmap1 = new Bitmap(objBitmap);
for (int x = 0; x < objBitmap.Width; x++)
```

```

for (int y = 0; y < objBitmap1.Height; y++)
{
    Color w = objBitmap.GetPixel(x, y);
    int r = w.R;
    Color wb = Color.FromArgb(r, r, r);
    objBitmap1.SetPixel(x, y, wb);
}
pictureBox2.Image = objBitmap1;

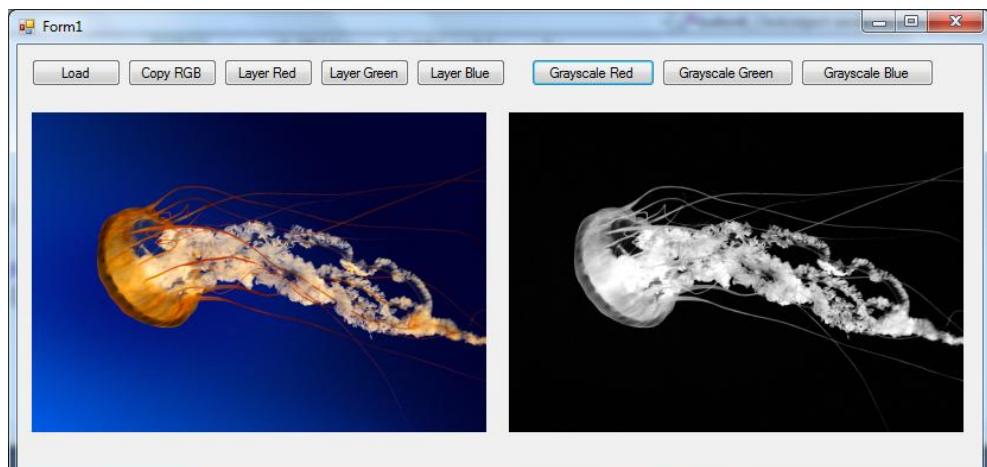
```



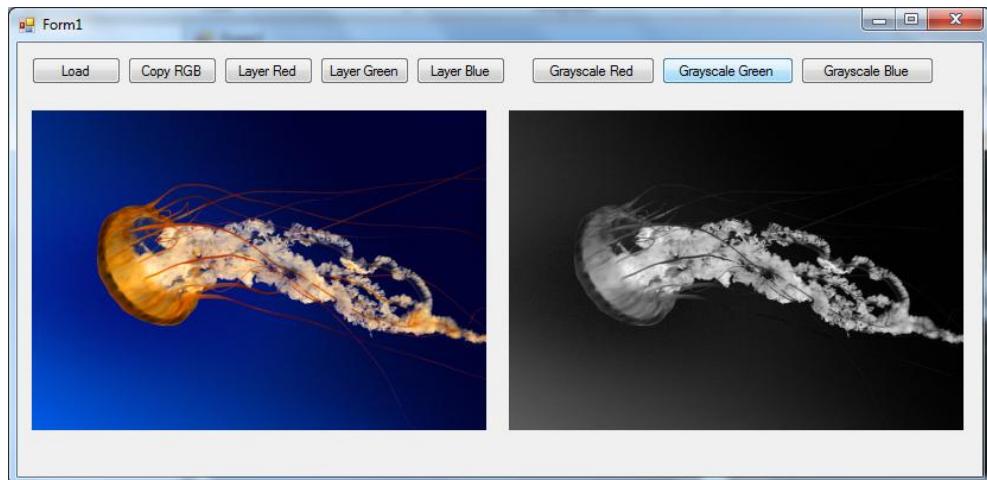
Gambar 3.8. Layout untuk tampilan grayscale red, green dan blue

- (3) Lakukan proses yang sama dengan langkah 2 untuk Grayscale Green dan Grayscale Blue. Untuk grayscale green, pada fungsi mengambil warna baru wb , yang dipakai hanya nilai G, dengan menyebutkan (g,g,g) . Untuk grayscale blue, pada fungsi mengambil warna baru wb , yang dipakai hanya nilai B, dengan menyebutkan (b,b,b) .

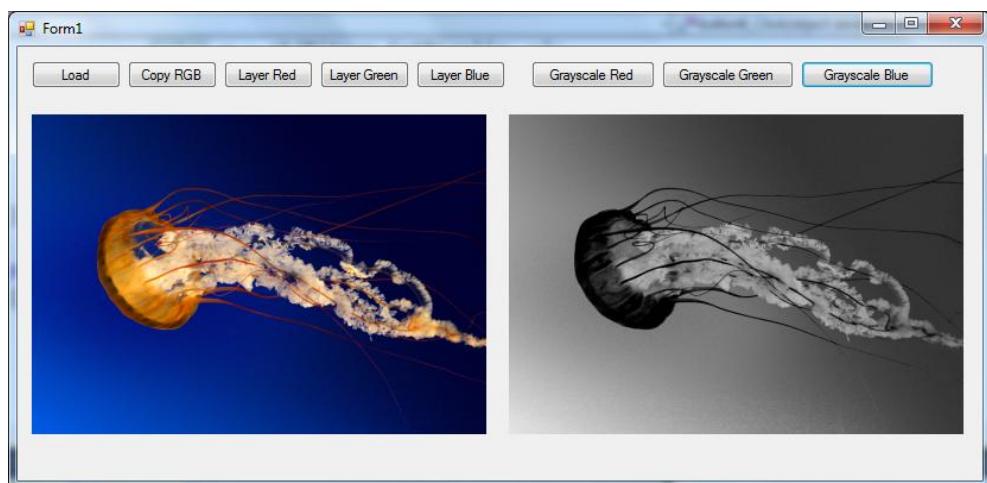
Hasil dari program Grayscale Red, Grayscale Green dan Grayscale Blue seperti terlihat pada gambar 3.9, 3.10 dan 3.11. Perhatikan perubahan derajat keabuan yang dihasilkan. Pada gambar tersebut hasil dari masing-masing derajat keabuan sangat berbeda karena komposisi warna gambar yang terdiri warna biru pada latar belakang dan warna obyek yang berbeda dari warna latar belakangnya.



Gambar 3.9. Hasil program Grayscale Red



Gambar 3.10. Hasil program Grayscale Green



Gambar 3.11. Hasil program Grayscale Blue

3.2. Laporan Praktikum

Laporan praktikum berupa:

- (1) Lengkapi program untuk grayscale green dan grayscale blue.
- (2) Tambahkan sebuah program untuk proses “Sephia” dengan mengambil nilai r dari layer R, kemudian pada warna yang baru ganti nilai R dengan $2*r$, nilai G dengan $1.8*r$ dan nilai B dengan r .
- (3) Tuliskan semua kode program dari praktikum tentang manipulasi RGB
- (4) Lakukan percobaan dengan gambar-gambar berikut, dan laporkan hasilnya serta berikan penjelasan mengapa hasilnya seperti yang tercetak di laporan.



Bab 4

Kuantisasi Data Citra

4.1. Dasar Teori

4.1.1. Citra Derajat Keabuan

Citra/gambar derajat keabuan (grayscale) adalah sebuah gambar yang hanya berisi terang dan gelap tanpa menggunakan warna. Semakin besar nilai derajat keabuan sebuah titik, titik akan terlihat semakin terang. Sebaliknya semakin kecil nilai derajat keabuan sebuah titik, titik akan terlihat semakin gelap.

Ciri dasar dari gambar derajat keabuan adalah nilai R, G dan B sama. Sehingga apapun yang dituliskan dengan format warna (x,x,x) akan menunjukkan gambar derajat keabuan:

$(r,r,r) \rightarrow$ menunjukkan gambar derajat keabuan dengan mengambil nilai dari layer R, sehingga titik-titik yang tidak mempunyai komponen warna merah akan berubah menjadi hitam.

$(g,g,g) \rightarrow$ menunjukkan gambar derajat keabuan dengan mengambil nilai dari layer G, sehingga titik-titik yang tidak mempunyai komponen warna hijau akan berubah menjadi hitam.

$(b,b,b) \rightarrow$ menunjukkan gambar derajat keabuan dengan mengambil nilai dari layer B, sehingga titik-titik yang tidak mempunyai komponen warna biru akan berubah menjadi hitam.

Untuk menghasilkan gambar derajat keabuan yang menggunakan semua komponen warna, bisa menggunakan nilai rata-rata dari semua nilai r dari layer R, g dari layer G dan b dari layer B, dan dituliskan dengan rumus:

$$x = \frac{r + g + b}{3}$$

Nilai x ada nilai derajat keabuan yang menggunakan nilai dari semua layer R, G dan B secara seimbang. Nilai x harus berada pada 0 s/d 255, dan beberapa bahasa pemrograman tidak mengijinkan nilai x keluar dari range tersebut.

4.1.2. Citra Hitam Putih

Citra/gambar hitam putih (BW) adalah sebuah gambar yang hanya berisi warna hitam atau putih. Hitam bernilai 0 dan putih bernilai 255. Untuk menjadikan gambar derajat keabuan menjadi gambar hitam putih dapat menggunakan rumus berikut:

$$xbw = \begin{cases} 0 & \text{jika } x < 128 \\ 255 & \text{jika } x \geq 128 \end{cases}$$

Rumus di atas disebut dengan rumus thresholding, dimana nilai thresholdnya adalah 128. Titik-titik yang nilainya kurang dari 128 akan diubah menjadi 0 (hitam) dan titik-titik yang nilainya di atas atau sama dengan 128 akan diubah menjadi 255 (putih).

4.1.3. Kuantisasi Citra

Nilai setiap titik pada citra/gambar derajat keabuan adalah 0 s/d 255. Ini berarti adalah 256 nilai. Proses kuantisasi citra adalah proses mengubah jumlah nilai keabuan, misalnya menjadi 8 nilai derajat keabuan.

256 step nilai: 0 1 2 3 4 5 6 7 8 9 ... 255

128 step nilai: 0 2 4 6 8 10 12 14 ... 254

64 step nilai: 0 4 8 12 16 20 24 ... 252

16 step nilai: 0 16 32 48 64 80 ... 240

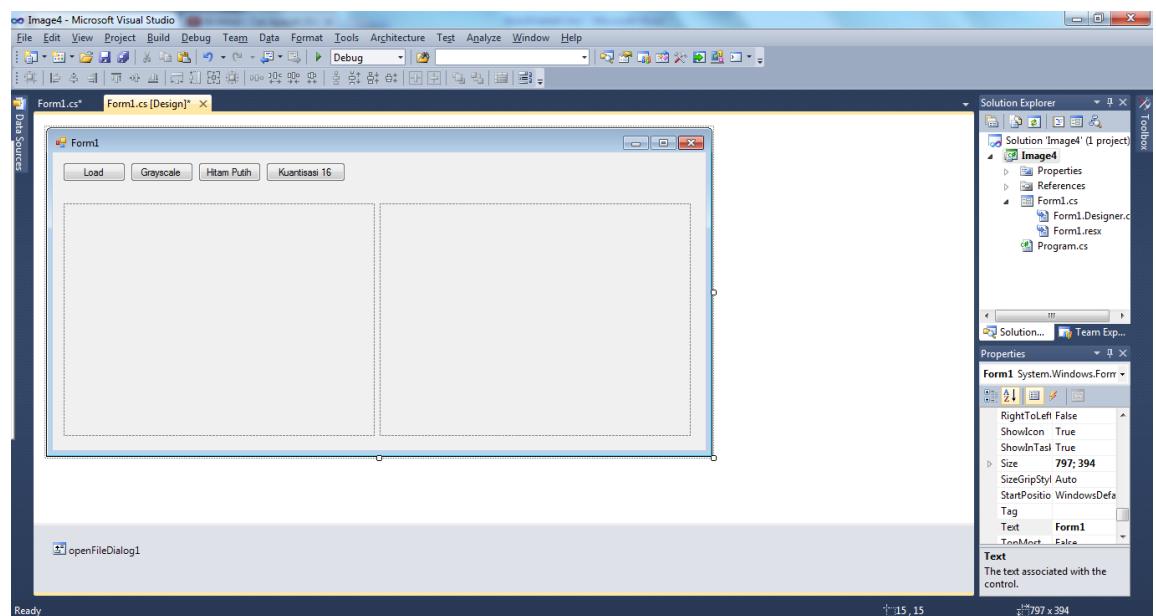
Untuk melakukan kuantisasi n step nilai dapat menggunakan rumus kuantisasi berikut ini:

$$xk = n * \text{int}\left(\frac{x}{n}\right)$$

4.2. Petunjuk Praktikum

Langkah-langkah berikut adalah langkah-langkah membuat program untuk mengubah gambar berwarna RGB menjadi gambar grayscale, gambar biner dan kuantisasi.

- (1) Buka Visual Studio .Net 2010. Buka project baru dengan File→Project→New Project
- (2) Pilih Visual C# [Windows Form Application]. Beri nama “Image4” dan Solution name mengikuti dari nama, dan tekan [Ok].



Gambar 4.1. Layout project Image4

- (3) Setelah keluar form baru dari project yang dibuat, pilih komponen 4 buah button, 2 buah Picturebox dan 1 buah openFileDialog. Pada Button1, ubah text menjadi “Load”. Pada Button2, ubah text menjadi “Grayscale”. Pada Button3, ubah text menjadi “Hitam Putih”. Pada Button 4, ubah text menjadi “Kuantisasi 16”. Pada Picturebox1 dan Picturebox2, atur size modenya dengan StretchImage. Atur tampilannya seperti gambar 4.1 di atas.
- (4) Tambahkan dua obyek Bitmap di Class Form1, yaitu objBitmap dan objBitmap1.

```
Bitmap objBitmap;
Bitmap objBitmap1;
```

(5) Double click pada button1 (Load), tambahkan program berikut:

```
DialogResult d = openFileDialog1.ShowDialog();
if (d == DialogResult.OK)
{
    objBitmap = new Bitmap(openFileDialog1.FileName);
    pictureBox1.Image = objBitmap;
}
```

(6) Double click pada button2 (Grayscale), tambahkan program berikut:

```
objBitmap1 = new Bitmap(objBitmap);
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap1.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int r = w.R;
        int g = w.G;
        int b = w.B;
        int xg = (int)((r + g + b) / 3);
        Color wb = Color.FromArgb(xg, xg, xg);
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;
```

(7) Double click pada button3 (Hitam Putih), tambahkan program berikut:

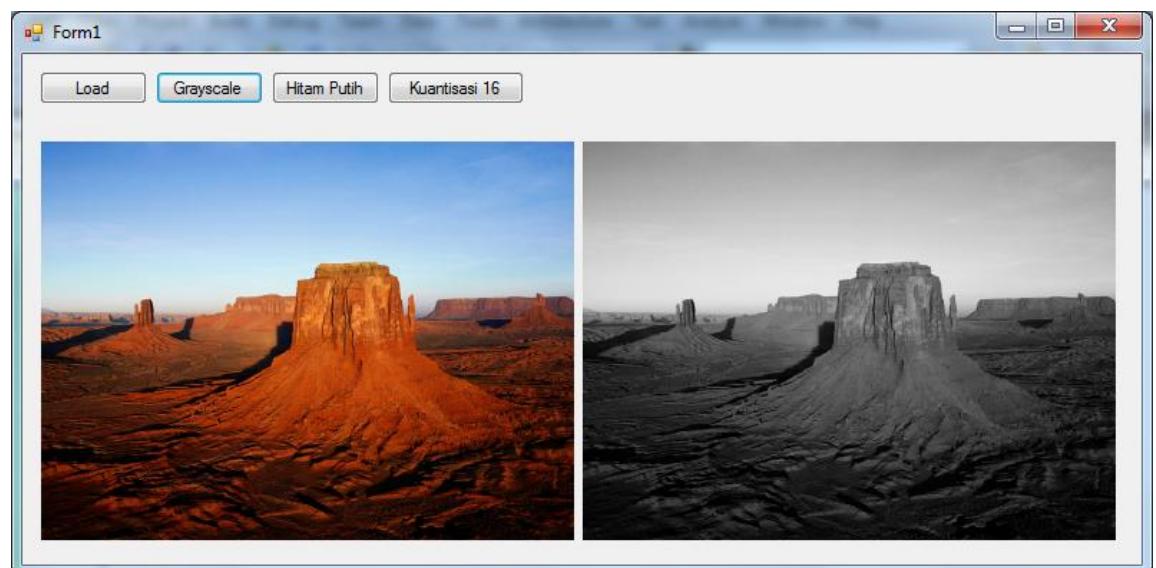
```
objBitmap1 = new Bitmap(objBitmap);
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap1.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int r = w.R;
        int g = w.G;
        int b = w.B;
        int xg = (int)((r + g + b) / 3);
        int xbw = 0;
        if (xg >= 128) xbw = 255;
        Color wb = Color.FromArgb(xbw, xbw, xbw);
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;
```

(8) Double click pada button4 (Kuantisasi 16), tambahkan program berikut:

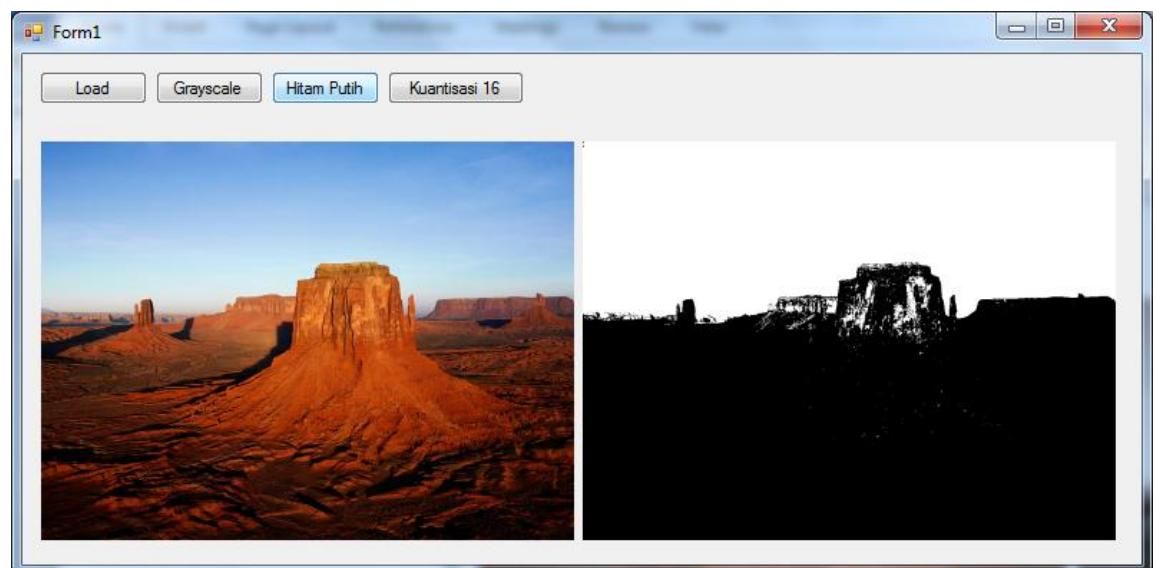
```
objBitmap1 = new Bitmap(objBitmap);
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap1.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int r = w.R;
```

```
int g = w.G;
int b = w.B;
int xg = (int)((r + g + b) / 3);
int xk = 16*int(xg/16);
Color wb = Color.FromArgb(xk, xk, xk);
objBitmap1.SetPixel(x, y, wb);
}
pictureBox2.Image = objBitmap1;
```

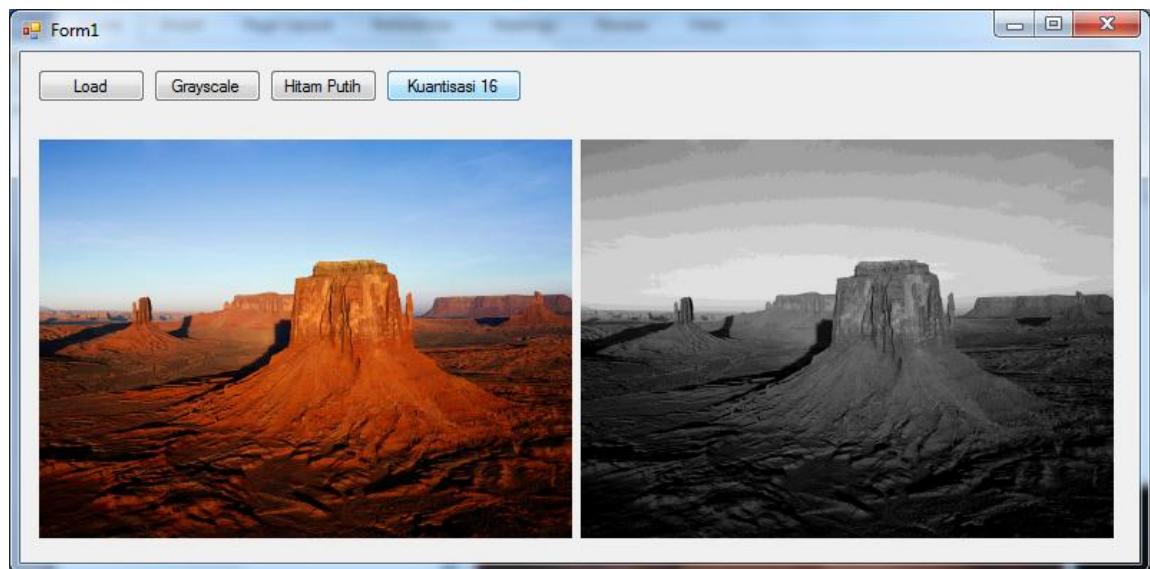
Hasil programnya adalah sebagai berikut:



Gambar 4.2. Hasil grayscale



Gambar 4.3. Hasil Hitam Putih



Gambar 4.4. Hasil Kuantisasi 16

4.1. Laporan Praktikum

Untuk pengolahan gambar derajat keabuan (grayscale), gambar hitam putih dan gambar kuantisasi, Tuliskan beberapa hal di bawah ini:

- (1) Tuliskan semua kode program pada praktikum ini.
- (2) Jelaskan perbedaan grayscale dari nilai rata-rata RGB dan kalau hanya menggunakan satu layer R, G atau B saja.
- (3) Tambahkan program untuk mengubah nilai threshold dari hitam putih dengan nilai 100, 200 dan nilai rata-rata dari derajat keabuan semua titik.
- (4) Tambahkan program untuk mengubah gambar kuantisasi 8, 32 dan 64. Jelaskan apa perbedaan dari hasil masing-masing kuantisasi.

Bab 5

Pengolahan Citra Derajat Keabuan

5.1. Dasar Teori

Beberapa pengolahan dasar dari citra derajat keabuan antara lain adalah: brightness, contrast dan invers.

5.1.1. Brightness

Brightness adalah proses menambah terang/gelapnya sebuah gambar derajat keabuan. Bila pada titik (x,y) mempunyai nilai derajar keabuan xg , maka proses brightness akan menambahkan nilai konstanta brightness kb .

$$xb = xg + kb$$

Bila kb bernilai positif maka hasilnya akan semakin terang, dan bila kb bernilai negatif maka hasilnya akan semakin gelap.

5.1.2. Contrast

Contrast adalah proses menaik-turunkan perbedaan nilai terang dan gelap sebuah gambar. Menaikkan contrast berarti menambah perbedaan antara nilai minimum dan nilai maksimum dari gambar. Demikin juga sebaliknya menurunkan contrast akan mengurangi perbedaan nilai minimum dan nilai maksimum dari gambar. Contrast dirumuskan dengan:

$$xb = c * xg$$

30

Bila c bernilai di atas 1 maka hasilnya akan menambah contrast, dan bila c antara 0 dan 1 maka hasilnya akan mengurangi contrast.

5.1.3. Invers

Invers adalah proses untuk membalik nilai derajat keabuan, dimana titik terang akan menjadi gelap dan titik gelap akan menjadi terang. Atau bisa juga disebut dengan negatif. Karena nilai batas maksimum derajat keabuan adalah 255, maka proses invers dapat dirumuskan dengan:

$$xb = 255 - xg$$

5.1.4. Auto-Level

Auto-Level adalah proses untuk membuat semuanilai derajat keabuan mulai 0 sampai dengan 255 terisi. Pada gambar yang terlalu gelap, biasanya nilai maksimumnya tidak mencapai 255. Pada gambar yang terlalu terang, biasanya nilai minimumnya tidak di nol. Dengan proses auto level ini, maka nilai maksimumnya dibuat 255 dan nilai minimumnya dibuat nol sehingga membuat gambar bisa semakin jelas.

Proses auto level menggunakan rumus sebagai berikut:

$$xb = \frac{255}{xg_{max} - xg_{min}}(xg - xg_{min})$$

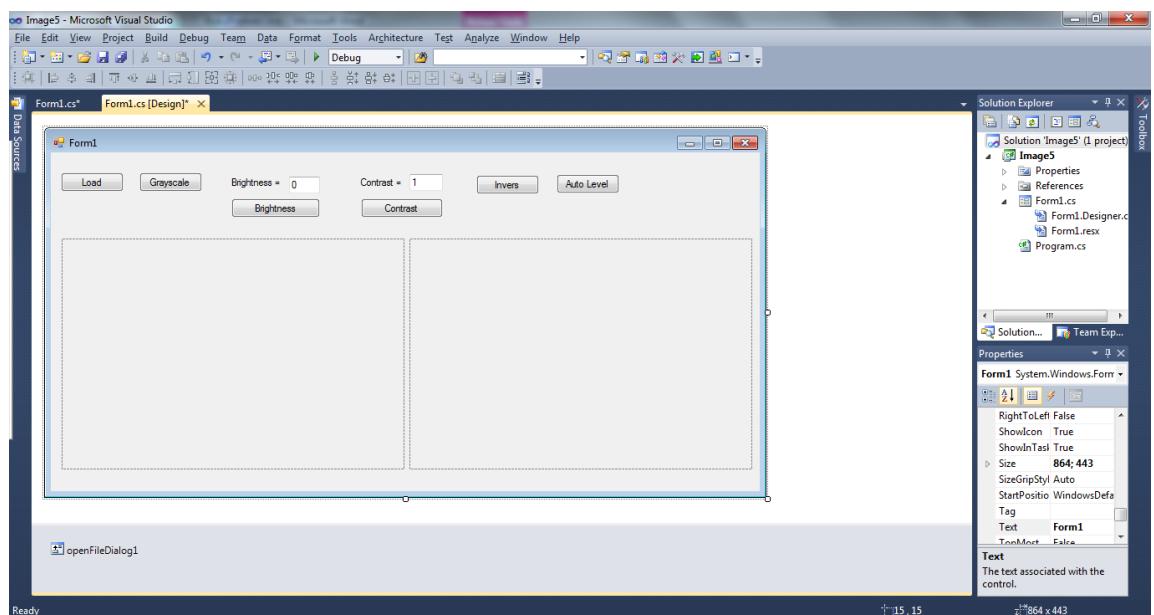
5.2. Petunjuk Praktikum

Langkah-langkah berikut adalah langkah-langkah untuk membuat program pengolahan citra derajat keabuan seperti brightness, contrast dan invers. Mengingat bahwa semua proses dalam gambar derajat keabuan, maka sebelumnya perlu dilakukan proses mengubah gambar RGB menjadi gambar derajat keabuan sebelum proses-proses yang lain.

31

- (1) Buka Visual Studio .Net 2010. Buat project baru dengan File→Project→New Project

- (2) Pilih Visual C# [Windows Form Application]. Beri nama “Image5” dan Solution name mengikuti dari nama, dan tekan [Ok].
- (3) Setelah keluar form baru dari project yang dibuat, pilih komponen 6 buah button, 2 buah Picturebox, 2 buah Label, 2 buah textbox dan 1 buah openFileDialog.
- (4) Pada Button1, ubah text menjadi “Load”. Pada Button2, ubah text menjadi “Grayscale”. Pada Button3, ubah text menjadi “Brightness”. Pada Button4, ubah text menjadi “Contrast”. Pada Button5, ubah text menjadi “Invers”. Pada Button6, ubah text menjadi “Auto Level”. Pada Picturebox1 dan Picturebox2, atur size modenya dengan StretchImage.
- (5) Pada Label1, ubah text menjadi “Brightness = “. Pada label2, ubah text menjadi “Contrast = “. Pada textbox1 dan textbox2, atur nilainya menjadi masing-masing 0 dan 1. Atur tampilannya seperti gambar 5.1 berikut.



Gambar 5.1. Layout project Image5

- (6) Tambahkan dua obyek Bitmap di Class Form1, yaitu objBitmap dan objBitmap1;

```
Bitmap objBitmap;
Bitmap objBitmap1;
```

- (7) Double click pada button1 (Load), tambahkan program berikut:

```
DialogResult d = openFileDialog1.ShowDialog();
if (d == DialogResult.OK)
{
    objBitmap = new Bitmap(openFileDialog1.FileName);
```

```

    pictureBox1.Image = objBitmap;
}

```

(8) Double click pada button2 (Grayscale), tambahkan program berikut:

```

for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int r = w.R;
        int g = w.G;
        int b = w.B;
        int xg = (int)((r + g + b) / 3);
        Color wb = Color.FromArgb(xg, xg, xg);
        objBitmap.SetPixel(x, y, wb);
    }
pictureBox1.Image = objBitmap;

```

(9) Double click pada button3 (Brightness), tambahkan program berikut:

```

objBitmap1 = new Bitmap(objBitmap);
int a=Convert.ToInt16(textBox1.Text);

for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int xg = w.R;
        int xb = xg+a;
        if (xb < 0) xb = 0;
        if (xb > 255) xb = 255;
        Color wb = Color.FromArgb(xb, xb, xb);
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;

```

(10) Double click pada button4 (Contrast), tambahkan program berikut:

```

objBitmap1 = new Bitmap(objBitmap);
float c = Convert.ToSingle(textBox2.Text);

for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int xg = w.R;
        int xb = (int)(c*xg);
        Color wb = Color.FromArgb(xb, xb, xb);
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;

```

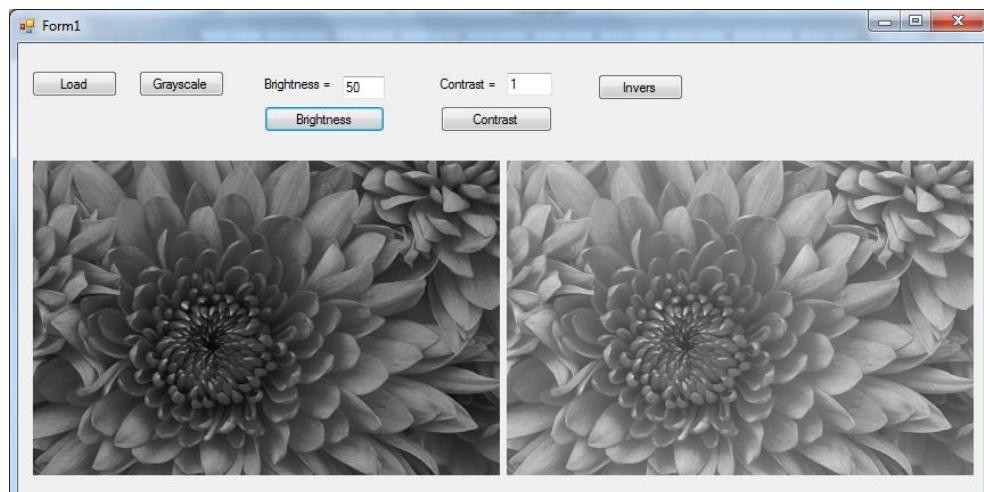
(11) Double click pada button5 (Invers), tambahkan program berikut:

```
objBitmap1 = new Bitmap(objBitmap);
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int xg = w.R;
        int xb = (int)(255-xg);
        Color wb = Color.FromArgb(xb, xb, xb);
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;
```

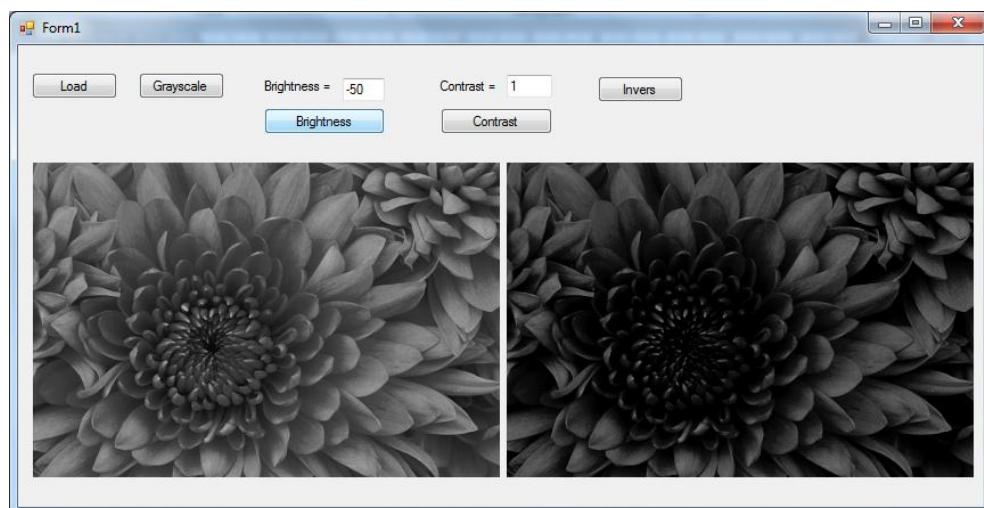
(12) Double click pada button6 (Invers), tambahkan program berikut:

```
objBitmap1 = new Bitmap(objBitmap);
//Mencari nilai maksimum dan minimum
int xgmax=0;
int xgmin=255;
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int xg = w.R;
        if (xg < xgmin) xgmin = xg;
        if (xg > xgmax) xgmax = xg;
    }
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap1.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int xg = w.R;
        int xb = (int)(255*(xg-xgmin)/(xgmax-xgmin));
        Color wb = Color.FromArgb(xb, xb, xb);
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;
```

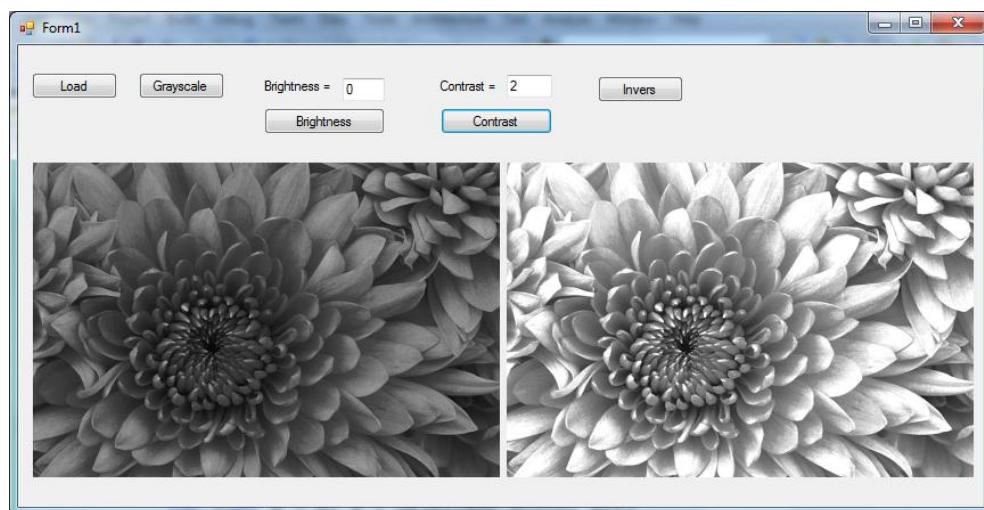
Hasil program di atas adalah sebagai berikut:



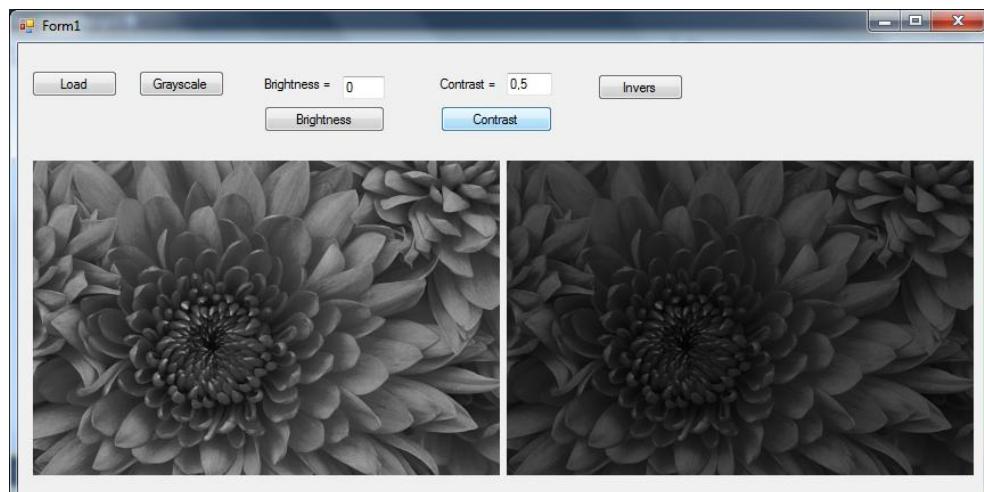
Gambar 5.2. Menaikkan brightness



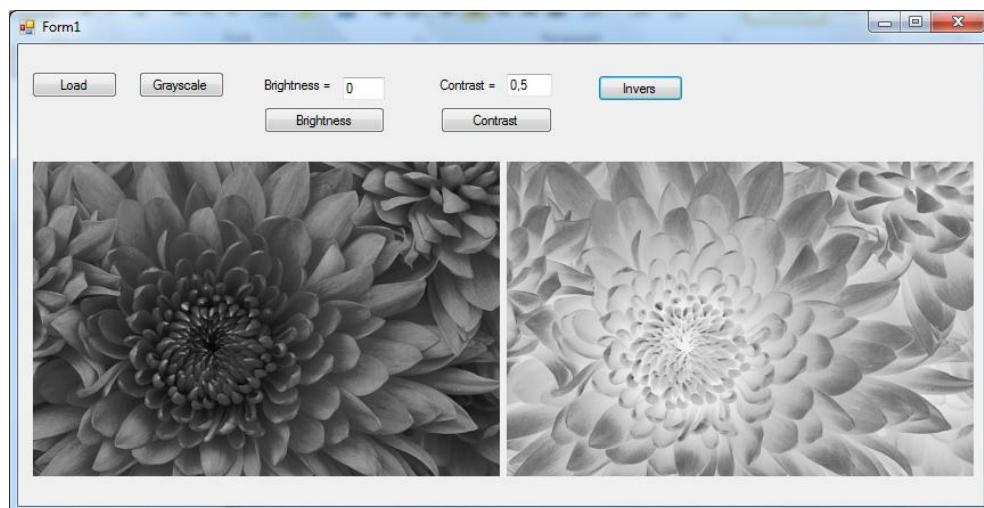
Gambar 5.3. Menurunkan brightness



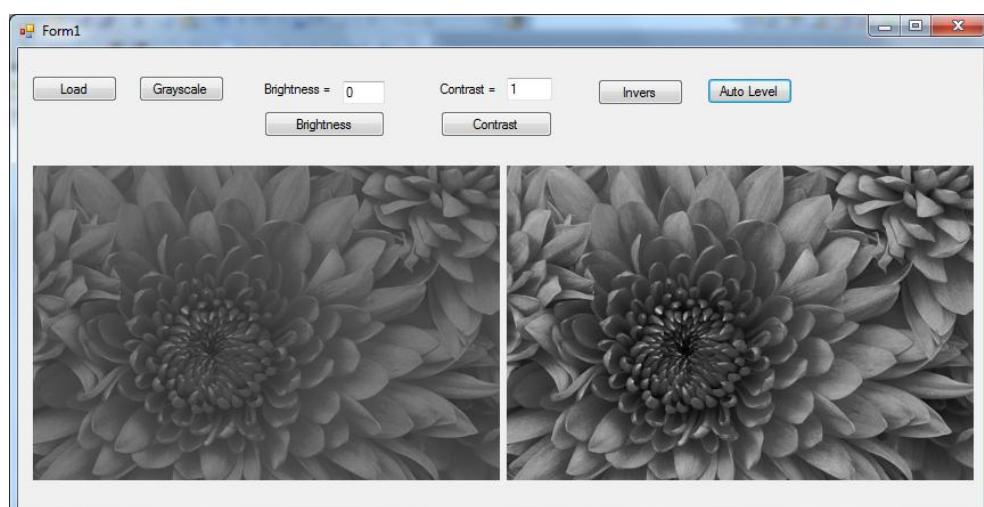
Gambar 5.4. Menaikkan contrast



Gambar 5.5. Menurunkan contrast



Gambar 5.6. Invers



Gambar 5.7. Auto-level

5.3. Laporan Praktikum

Untuk laporan praktikum, tuliskan hal-hal berikut:

- (1) Tuliskan semua kode program pada project Image5.
- (2) Apa perbedaan antara brightness dan contrast.
- (3) Apa yang terjadi bila rumus brightness diganti dengan $xb=xg+255$?
- (4) Bila rumus untuk invers diganti dengan $xb=128-xg$. Apa yang akan terjadi?
- (5) Apa manfaat dari auto-level?

Bab 6

Histogram Citra Derajat Keabuan

6.1. Dasar Teori

6.1.1. Histogram

Histogram citra/gambar derajat keabuan menunjukkan distribusi nilai-nilai derajat keabuan dari gambar tersebut. Histogram $H(xg)$ ini menyatakan jumlah titik yang mempunyai nilai derajat keabuan xg . Untuk menghasilkan histogram derajat keabuan ini, baca nilai derajat keabuan xg dari setiap titik (x,y) . Tambahkan $H(xg)$ secara counter $H(xg)=H(xg)+1$.

Bila histogram $H(xg)$ ini dibagi dengan jumlah titik n_x dan n_y , maka nilainya akan menjadi nilai kemungkinan dari setiap terjadinya derajat keabuan xg , atau disebut juga dengan *probability density function* atau fungsi kepadatan probabilitas atau juga bisa dinamakan dengan fungsi distribusi dari derajat keabuan xg .

Karena histogram menyajikan distribusi setiap nilai derajat keabuan, maka dapat dengan histogram ini kita bisa melihat distribusi terang-gelapnya sebuah gambar. Gambar yang histogramnya cenderung ke kiri adalah gambar yang gelap, sedangkan gambar yang histogramnya cenderung ke kanan adalah gambar yang terang.

6.1.2. Fungsi Distribusi Kumulatif

Fungsi Distribusi Kumulatif atau Cumulative Density Function (CDF) $C(xg)$ dari sebuah gambar derajat keabuan menunjukkan jumlah titik yang nilai derajat keabuannya mulai nol sampai xg . Fungsi ini menyatakan model kurva histogram yang selalu naik. Fungsi distribusi kumulatif $C(xg)$ dapat dirumuskan dengan:

$$C(xg) = \sum_{i=0}^{xg} H(i)$$

Atau secara sederhana bisa dituliskan dengan konsep counter:

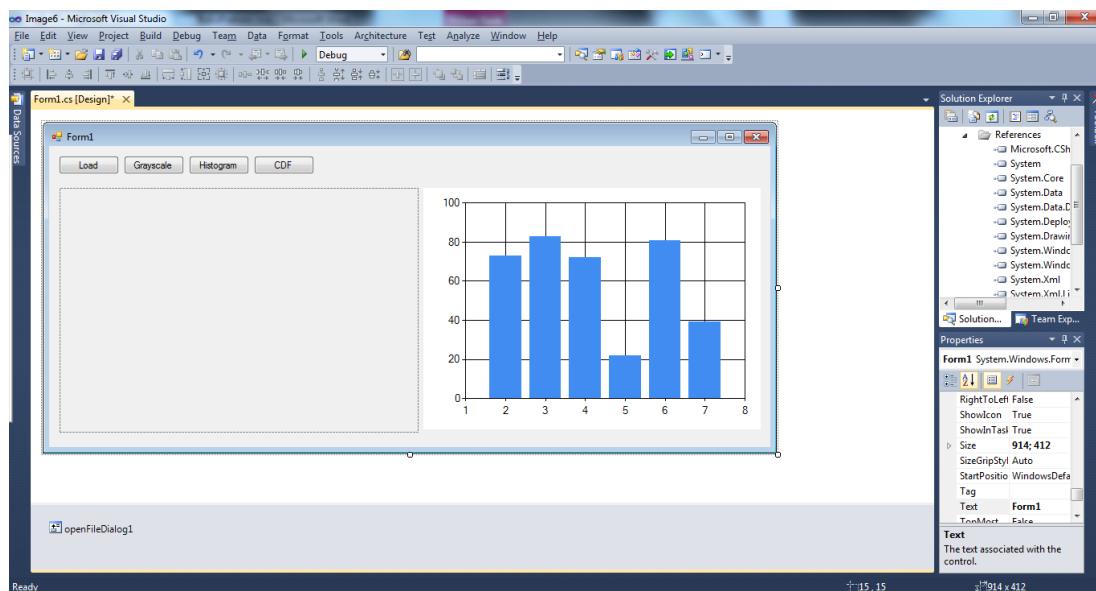
$$C(0) = H(0)$$

$$C(xg) = C(xg - 1) + H(xg)$$

6.2. Petunjuk Praktikum

Langkah-langkah berikut adalah membuat program untuk menampilkan histogram dan fungsi distribusi kumulatif dari gambar derajat keabuan.

- (1) Buka Visual Studio .Net 2010. Buat project baru dengan File→Project→New Project.
- (2) Pilih Visual C# [Windows Form Application]. Beri nama “Image6” dan Solution name mengikuti dari nama, dan tekan [Ok].
- (3) Setelah keluar form baru dari project yang dibuat, pilih komponen 4 buah button, 1 buah Picturebox, 1 buah Chart dan 1 buah openFileDialog.



39

Gambar 6.1. Layout project Image6

(4) Pada Button1, ubah text menjadi “Load”. Pada Button2, ubah text menjadi “Grayscale”. Pada Button3, ubah text menjadi “Histogram”. Pada Button4, ubah text menjadi “CDF”. Pada Picturebox1, atur size modenya dengan StretchImage. Untuk Chart1, hapus legendnya dengan meng-click property Legend, lalu setelah keluar jendela Legend, tekan tombol Remove yang ada di bagian bawah jendela. Atur tampilannya seperti gambar 6.1 di atas.

(5) Tambahkan obyek Bitmap di Class Form1, yaitu objBitmap dan objBitmap1.

```
Bitmap objBitmap;
```

(6) Double click pada button1 (Load), tambahkan program berikut:

```
DialogResult d = openFileDialog1.ShowDialog();
if (d == DialogResult.OK)
{
    objBitmap = new Bitmap(openFileDialog1.FileName);
    pictureBox1.Image = objBitmap;
}
```

(7) Double click pada button2 (Grayscale), tambahkan program berikut:

```
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int r = w.R;
        int g = w.G;
        int b = w.B;
        int xg = (int)((r + g + b) / 3);
        Color wb = Color.FromArgb(xg, xg, xg);
        objBitmap.SetPixel(x, y, wb);
    }
pictureBox1.Image = objBitmap;
```

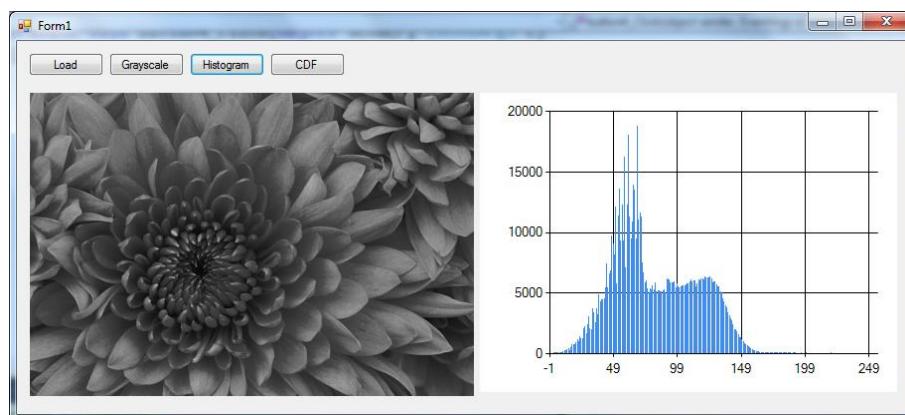
(8) Double click pada button3 (Histogram), tambahkan program berikut:

```
float[] h = new float[256];
int i;
for(i = 0 ; i < 256; i++) h[i]=0;
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int xg = w.R;
        h[xg] = h[xg] + 1;
    }
for (i = 0; i < 256; i++)
{
    chart1.Series["Series1"].Points.AddXY(i, h[i]);
}
```

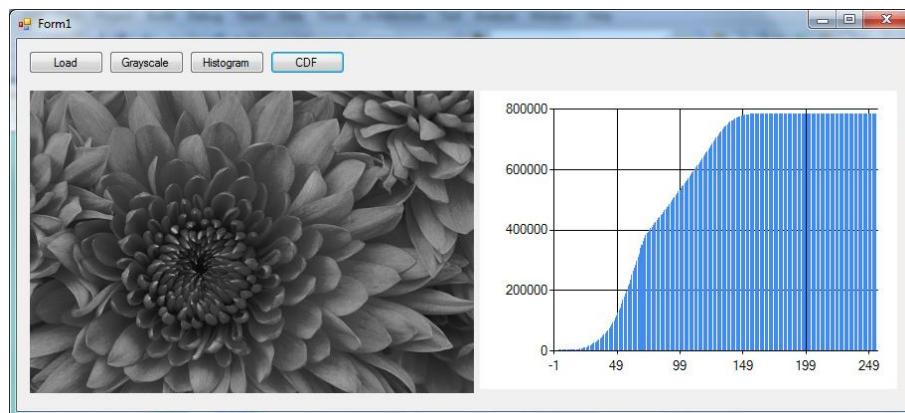
(9) Double click pada button4 (CDF), tambahkan program berikut:

```
float[] h = new float[256];
float[] c = new float[256];
int i;
for (i = 0; i < 256; i++) h[i] = 0;
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
{
    Color w = objBitmap.GetPixel(x, y);
    int xg = w.R;
    h[xg] = h[xg] + 1;
}
c[0] = h[0];
for (i = 1; i < 256; i++) c[i] = c[i - 1] + h[i];
for (i = 0; i < 256; i++)
{
    chart1.Series["Series1"].Points.AddXY(i, c[i]);
}
```

Hasil dari program di atas adalah:



Gambar 6.2. Hasil histogram



Gambar 6.3. Hasil CDF

6.3. Laporan Praktikum

Dari program di atas, tulis dan jawab pertanyaan-pertanyaan berikut dalam bentuk laporan praktikum:

- (1) Tuliskan semua kode program dan jelaskan bagian-bagian yang dianggap penting.
- (2) Tambahkan program untuk mendapatkan pdf (*probability density function*) dari gambar dengan membagi histogram dengan jumlah titik pada gambar tersebut.
- (3) Tambahkan program untuk mendapatkan CDF dari PDF di atas.
- (4) Apa perbedaan menggunakan histogram dan PDF?
- (5) Apa perbedaan histogram dari gambar-gambar derajat keabuan yang berasal dari gambar-gambar berikut:



Bab 7

Histogram Equalization

7.1. Dasar Teori

Histogram Equalization adalah sebuah proses untuk memperbaiki terang/gelap gambar derajat keabuan dengan memanfaatkan distribusi nilai-nilai derajat keabuan dari gambar tersebut. Histogram equalization ini mengubah distribusi kemunculan setiap nilai derajat keabuan ke arah seragam, artinya setiap nilai derajat keabuan mempunyai kemungkinan yang sama baik dari bagian gelap, sedang dan terang.

Hasil dari histogram equalization adalah membuat distribusi derajat keabuan gambar menjadi seragam dan gambar tampak lebih jelas. Gambar yang terlalu gelap akan diterangkan menjadi normal, dan gambar yang terlalu terang akan digelapkan menjadi normal.

Proses dari histogram equalization adalah:

- (1) Buat histogram derajat keabuan $H(xg)$
- (2) Buat distribusi kumulatif $C(xg)$
- (3) Ambil nilai derajat keabuan xg pada setiap titik, lalu ubah menjadi:

$$xb = \frac{255 \cdot C(xg)}{n_x \cdot n_y}$$

Dimana:

xg adalah nilai derajat keabuan pada titik (x,y)

$C(xg)$ adalah distribusi kumulatif dari xg

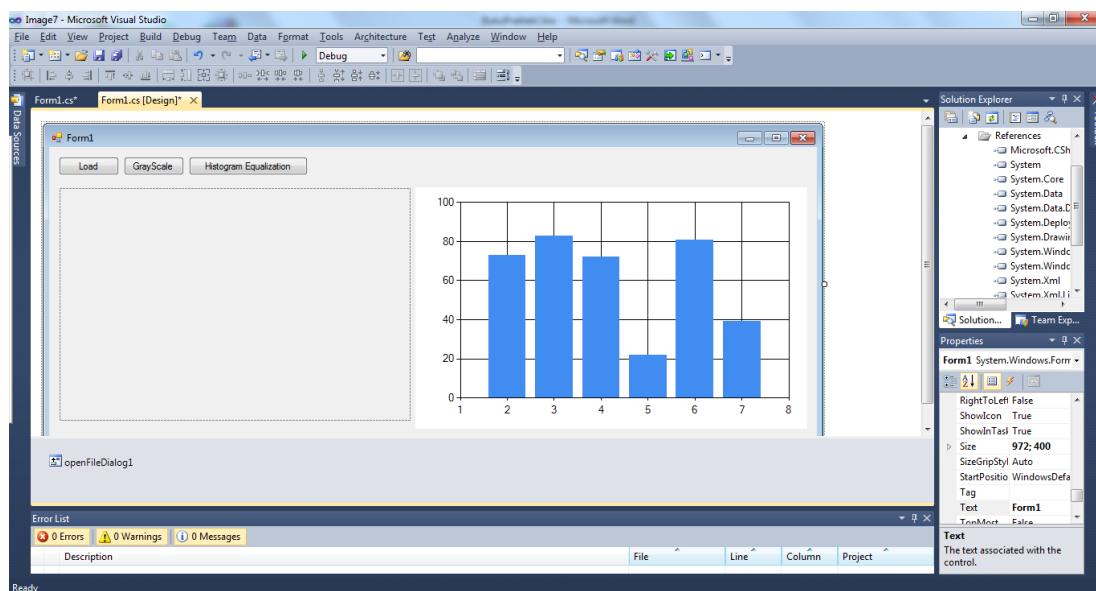
n_x adalah lebar gambar

n_y adalah tinggi gambar.

7.2. Petunjuk Praktikum

Langkah-langkah berikut adalah membuat program untuk memperbaiki gambar derajat keabuan menggunakan Histogram Equalization.

- (1) Buka Visual Studio .Net 2010. Buka project baru dengan File→Project→New Project.
- (2) Pilih Visual C# [Windows Form Application]. Beri nama “Image7” dan Solution name mengikuti dari nama, dan tekan [Ok].
- (3) Setelah keluar form baru dari project yang dibuat, pilih komponen 3 buah button, 1 buah Picturebox, 1 buah Chart dan 1 buah openFileDialog.



Gambar 7.1. Layout project Image7

- (4) Pada Button1, ubah text menjadi “Load”. Pada Button2, ubah text menjadi “Grayscale”. Pada Button3, ubah text menjadi “Histogram Equalization”. Pada Picturebox1, atur size modenya dengan StretchImage. Untuk Chart1, hapus legendnya dengan meng-click property Legend, lalu setelah keluar jendela Legend, tekan tombol Remove yang ada di bagian bawah jendela. Atur tampilannya seperti gambar 7.1 di atas.

(5) Tambahkan obyek Bitmap di Class Form1, yaitu objBitmap dan objBitmap1.

```
Bitmap objBitmap;
```

(6) Double click pada button1 (Load), tambahkan program berikut:

```
DialogResult d = openFileDialog1.ShowDialog();
if (d == DialogResult.OK)
{
    objBitmap = new Bitmap(openFileDialog1.FileName);
    pictureBox1.Image = objBitmap;
}
```

(7) Double click pada button2 (Grayscale), tambahkan program berikut:

```
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int r = w.R;
        int g = w.G;
        int b = w.B;
        int xg = (int)((r + g + b) / 3);
        Color wb = Color.FromArgb(xg, xg, xg);
        objBitmap.SetPixel(x, y, wb);
    }
pictureBox1.Image = objBitmap;
```

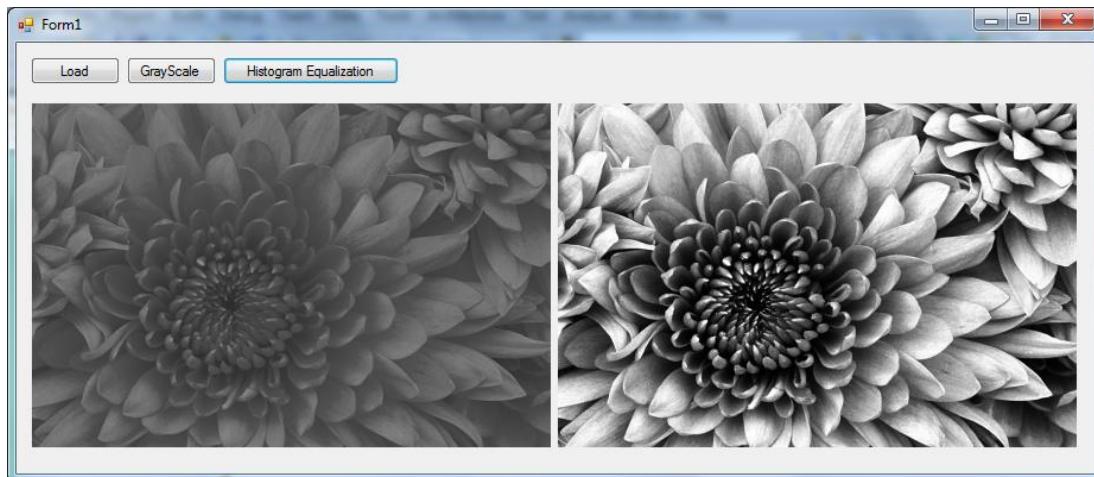
(8) Double click pada button3 (Histogram Equalization), tambahkan program berikut:

```
float[] h = new float[256];
float[] c = new float[256];
int i;
for (i = 0; i < 256; i++) h[i] = 0;
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int xg = w.R;
        h[xg] = h[xg] + 1;
    }
c[0] = h[0];
for (i = 1; i < 256; i++) c[i] = c[i-1] + h[i];
int nx = objBitmap.Width;
int ny = objBitmap.Height;
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int xg = w.R;
        int xb=(int)(255*c[xg]/nx/ny);
        Color wb = Color.FromArgb(xb, xb, xb);
        objBitmap.SetPixel(x, y, wb);
```

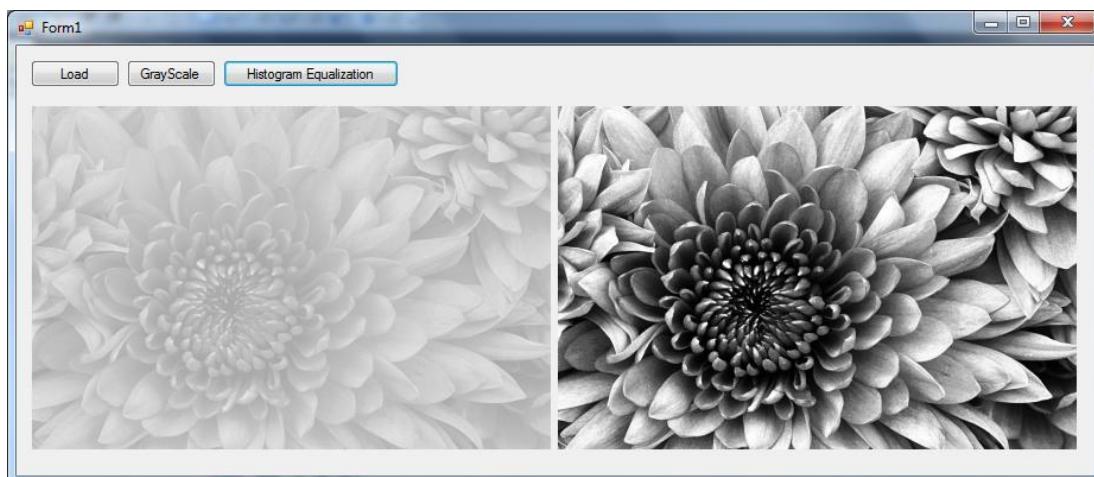


```
        }
        pictureBox2.Image = objBitmap;
```

Hasil dari program di atas adalah sebagai berikut:



Gambar 7.2. Hasil histogram equalization pada gambar yang gelap



Gambar 7.3. Hasil histogram equalization pada gambar yang terang

7.3. Laporan Praktikum

Dari program di atas, tulis dan jawab pertanyaan-pertanyaan berikut dalam bentuk laporan praktikum:

- (1) Tuliskan semua kode program dan jelaskan bagian-bagian yang dianggap penting.

- (2) Apa perbedaan hasil dari histogram equalization dan auto level?
- (3) Apa yang terjadi bila gambar-gambar berikut diproses dengan histogram equalization?



Bab 8

Konvolusi & Image Filtering

8.1. Dasar Teori

Image filtering adalah proses filter pada gambar baik berupa low-pass filter, high-pass filter maupun band-stop filter. Untuk melakukan filter pada image, diperlukan apa yang dinamakan dengan kernel filter H. Hasil filter berupa gambar Q diperoleh dengan cara melakukan konvolusi antara kernel filter H dan gambar P atau dituliskan dengan:

$$Q = H \otimes P$$

Dimana konvolusi ini bisa dituliskan dengan:

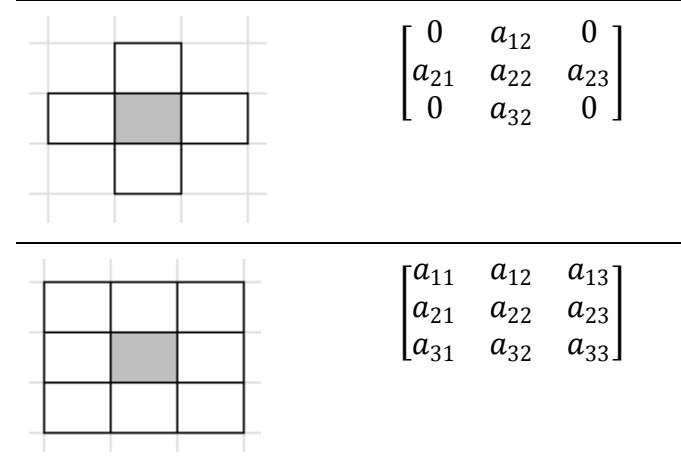
$$Q(x, y) = \sum_{j=-s}^s \sum_{i=-s}^s H(i + s, j + s) \cdot P(x + i, y + j)$$

Sebagai contoh, bila kita mempunyai kernel filter 3x3, maka proses filter dengan konvolusi di atas akan menghasilkan operasi sebagai berikut:

$$\begin{aligned} Q(x, y) = & h(1,1).p(x - 1, y - 1) + h(1,2).p(x - 1, y) + h(1,3).p(x - 1, y + 1) \\ & + h(2,1).p(x, y - 1) + h(2,2).p(x, y) + h(2,3).p(x, y + 1) \\ & + h(3,1).p(x + 1, y - 1) + h(3,2).p(x + 1, y) + h(3,3).p(x + 1, y + 1) \end{aligned}$$

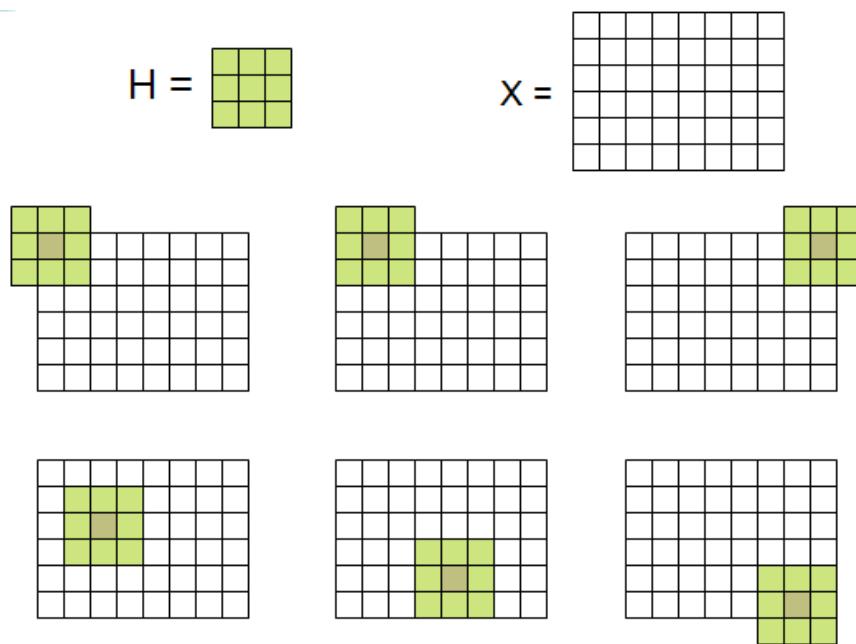
Proses konvolusi pada dasarnya adalah proses mengubah nilai sebuah titik menggunakan nilai dari titik-titik tetangganya. Ada beberapa model titik tetangga yang bisa dimanfaatkan seperti 4 titik tetangga, 8 titik tetangga, dan 24 titik tetangga. Meskipun tidak menutup kemungkinan penggunaan titik tetangga dalam jumlah yang besar, hanya saja ada pertimbangan “*time consume*” saat memilih

jumlah titik tetangga yang dinyatakan dalam bentuk kernel. Gambar 8.1 memberikan contoh hubungan antara matrik kernel dan titik tetangga.



Gambar 8.1. Hubungan titik tetangga dan matrik kernel

Secara visual proses konvolusi dengan matrik kernel H berukuran 3x3 dan image X dapat digambarkan seperti pada gambar 8.2 berikut:

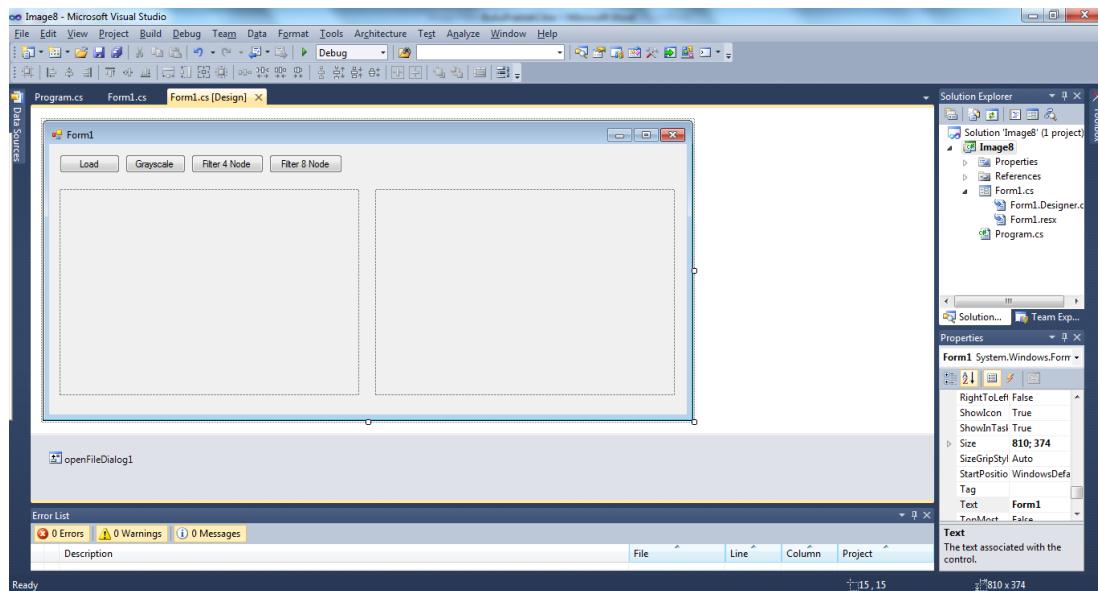


Gambar 8.2. Proses konvolusi secara visual

8.2. Petunjuk Praktikum

Langkah-langkah berikut adalah pembuatan program Image filtering menggunakan konvolusi dengan filter 4 node, filter 8 node dan filter 24 node. Jumlah node menyatakan jumlah tetangga.

- (1) Buka Visual Studio .Net 2010. Buat project baru dengan File→Project→New Project.
- (2) Pilih Visual C# [Windows Form Application]. Beri nama “Image8” dan Solution name mengikuti dari nama, dan tekan [Ok].
- (3) Setelah keluar form baru dari project yang dibuat, pilih komponen 4 buah button, 2 buah Picturebox dan 1 buah openFileDialog.



Gambar 8.3. Layout project Image8

- (4) Pada Button1, ubah text menjadi “Load”. Pada Button2, ubah text menjadi “Grayscale”. Pada Button3, ubah text menjadi “Filter 4 Node”. Pada Button4, ubah text menjadi “Filter 8 Node”. Pada Picturebox1, atur size modenya dengan StretchImage. Atur tampilannya seperti gambar 8.3 di atas.
- (5) Tambahkan obyek Bitmap di Class Form1, yaitu objBitmap dan objBitmap1.

```
Bitmap objBitmap;
Bitmap objBitmap1;
```

- (6) Double click pada button1 (Load), tambahkan program berikut:

```

    DialogResult d = openFileDialog1.ShowDialog();
    if (d == DialogResult.OK)
    {
        objBitmap = new Bitmap(openFileDialog1.FileName);
        pictureBox1.Image = objBitmap;
    }
}

```

(7) Double click pada button2 (Grayscale), tambahkan program berikut:

```

for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int r = w.R;
        int g = w.G;
        int b = w.B;
        int xg = (int)((r + g + b) / 3);
        Color wb = Color.FromArgb(xg, xg, xg);
        objBitmap.SetPixel(x, y, wb);
    }
pictureBox1.Image = objBitmap;

```

(8) Program berikut adalah konvolusi dengan kernel 4 node yang diambil contoh:

$$H = \begin{bmatrix} 0 & 0.2 & 0 \\ 0.2 & 0.2 & 0.2 \\ 0 & 0.2 & 0 \end{bmatrix}$$

Double click pada button2 (Filter 4 Node), tambahkan program berikut:

```

float[] a= new float[5];
a[1]=(float)0.2;
a[2]=(float)0.2;
a[3]=(float)0.2;
a[4]=(float)0.2;
a[0]=(float)0.2;
objBitmap1 = new Bitmap(objBitmap);
for (int x = 1; x < objBitmap.Width-1; x++)
    for (int y = 1; y < objBitmap.Height-1; y++)
    {
        Color w1 = objBitmap.GetPixel(x - 1, y);
        Color w2 = objBitmap.GetPixel(x + 1, y);
        Color w3 = objBitmap.GetPixel(x, y - 1);
        Color w4 = objBitmap.GetPixel(x, y + 1);
        Color w = objBitmap.GetPixel(x, y);
        int x1 = w1.R;
        int x2 = w2.R;
        int x3 = w3.R;
        int x4 = w4.R;
        int xg = w.R;
        int xb = (int)(a[0]*xg);
        xb=(int)(xb+a[1]*x1+a[2]*x2+a[3]*x3+a[4]*x4);
        if (xb < 0) xb = 0;
    }
}

```

```

        if (xb > 255) xb = 255;
        Color wb = Color.FromArgb(xb, xb, xb);
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;

```

- (9) Program berikut adalah konvolusi dengan kernel 8 node yang diambil contoh:

$$H = \begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{bmatrix}$$

Double click pada button2 (Filter 4 Node), tambahkan program berikut:

```

float[] a= new float[10];
a[1]=(float)0.1;
a[2]=(float)0.1;
a[3]=(float)0.1;
a[4]=(float)0.1;
a[5]=(float)0.2;
a[6]=(float)0.1;
a[7]=(float)0.1;
a[8]=(float)0.1;
a[9]=(float)0.1;
objBitmap1 = new Bitmap(objBitmap);
for (int x = 1; x < objBitmap.Width-1; x++)
    for (int y = 1; y < objBitmap.Height-1; y++)
    {
        Color w1 = objBitmap.GetPixel(x-1, y-1);
        Color w2 = objBitmap.GetPixel(x-1, y);
        Color w3 = objBitmap.GetPixel(x-1, y+1);
        Color w4 = objBitmap.GetPixel(x, y-1);
        Color w5 = objBitmap.GetPixel(x, y);
        Color w6 = objBitmap.GetPixel(x, y+1);
        Color w7 = objBitmap.GetPixel(x+1, y-1);
        Color w8 = objBitmap.GetPixel(x+1, y);
        Color w9 = objBitmap.GetPixel(x+1, y+1);
        int x1 = w1.R;
        int x2 = w2.R;
        int x3 = w3.R;
        int x4 = w4.R;
        int x5 = w5.R;
        int x6 = w6.R;
        int x7 = w7.R;
        int x8 = w8.R;
        int x9 = w9.R;
        int xb = (int)(a[1]*x1+a[2]*x2+a[3]*x3);
        xb=(int)(xb+a[4]*x4+a[5]*x5+a[6]*x6);
        xb=(int)(xb+a[7]*x7+a[8]*x8+a[9]*x9);
        if (xb < 0) xb = 0;
        if (xb > 255) xb = 255;
        Color wb = Color.FromArgb(xb, xb, xb);

```

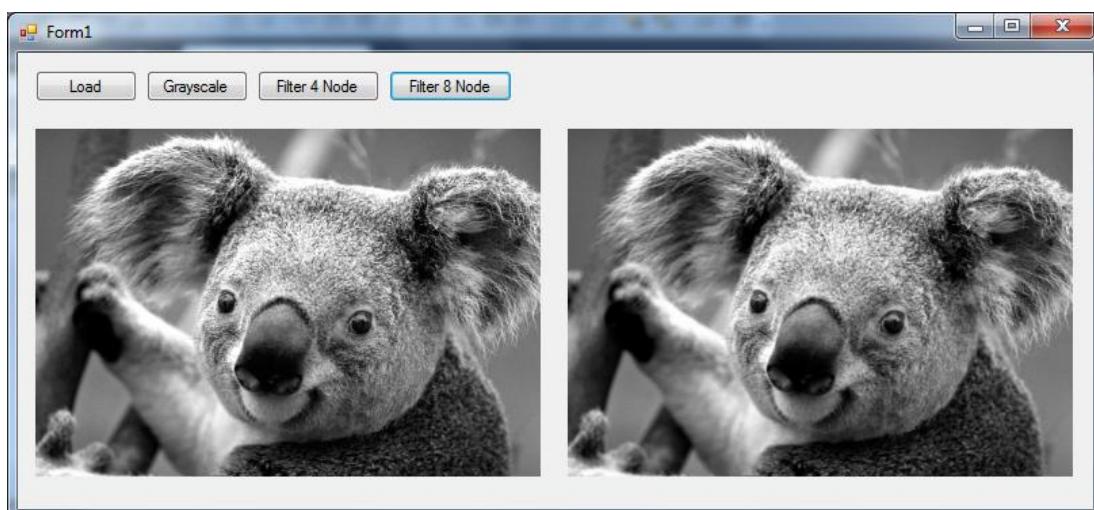


```
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;
```

Hasil dari program di atas adalah sebagai berikut. Tidak tampak perbedaan yang berarti, tetapi kalau diperhatikan lebih seksama, gambar hasil semakin halus dan blur. Filter dengan 8 node akan semakin halus dan blur dibandingkan dengan filter dengan 4 node.



Gambar 8.4. Hasil filter 4 node



53

Gambar 8.5. Hasil filter 8 node

8.3. Laporan Praktikum

Dari program di atas, tulis dan jawab pertanyaan-pertanyaan berikut dalam bentuk laporan praktikum:

(1) Tuliskan semua kode program dan jelaskan bagian-bagian yang dianggap penting.

(2) Ubah matrik kernel 4 node dengan matrik berikut, dan apa yang terjadi?

a. $H = \begin{bmatrix} 0 & -0.5 & 0 \\ -0.5 & 0 & 0.5 \\ 0 & 0.5 & 0 \end{bmatrix}$

b. $H = \begin{bmatrix} 0 & -0.5 & 0 \\ -0.5 & 1 & 0.5 \\ 0 & 0.5 & 0 \end{bmatrix}$

(3) Ubah matrik kernel 8 node dengan matrik berikut, dan apa yang terjadi?

a. $H = \begin{bmatrix} -1 & -0.5 & 0 \\ -0.5 & 0 & 0.5 \\ 0 & 0.5 & 1 \end{bmatrix}$

b. $H = \begin{bmatrix} -1 & -0.5 & 0 \\ -0.5 & 1 & 0.5 \\ 0 & 0.5 & 1 \end{bmatrix}$

Bab 9

Image Noise

9.1. Dasar Teori

Image noise bisa dikatakan sebagai titik-titik yang mengganggu atau membuat sebuah gambar menjadi cacat. Atau ada yang menyebutkan dengan titik-titik semut pada sebuah gambar, karena umumnya noise berupa titik-titik seperti semut yang mengganggu visualisasi sebuah gambar. Noise pada gambar sebenarnya terdapat beberapa macam tergantung pada jenis cacat yang terjadi.

Ada tiga macam jenis noise yang banyak terjadi pada sebuah gambar sebagai akibat cacatnya peralatan dalam menangkap gambar atau kamera, yaitu:

- 1) Noise Gaussian: Noise ini berupa titik-titik berwarna dan menimbulkan efek seperti bintik-bintik. Noise ini muncul akibat pemakaian sensor yang terlalu sensitif, sehingga warna yang seharusnya seragam malah menjadi tidak seragam dan bahkan memunculkan noise ini.



55

Gambar 9.1. Noise Gaussian

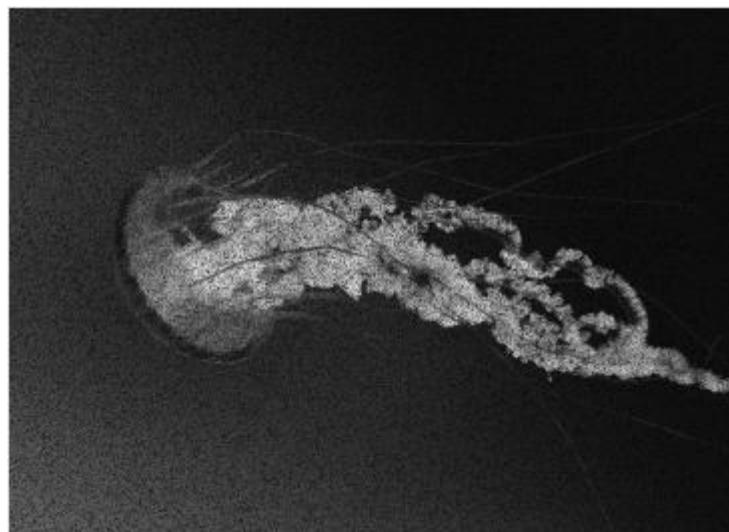


Sebuah titik (x,y) yang terkena noise gaussian, nilainya akan berubah menjadi:

$$xb = xg \pm ns$$

Dimana ns adalah bilangan acak antara -128 s/d 128.

- 2) Noise Speckle: Noise ini berupa titik-titik hitam dan muncul akibat dari adanya sensor-sensor yang mati. Sebuah titik (x,y) yang terkena noise speckle, nilainya akan berubah menjadi nol.



Gambar 9.2. Noise Speckle

- 3) Noise Salt & Pepper: Noise ini berupa titik-titik putih, dan muncul akibat dari adanya debu atau media lain yang menyebabkan lensa kotor. Sebuah titik (x,y) yang terkena noise speckle, nilainya akan berubah menjadi nol.

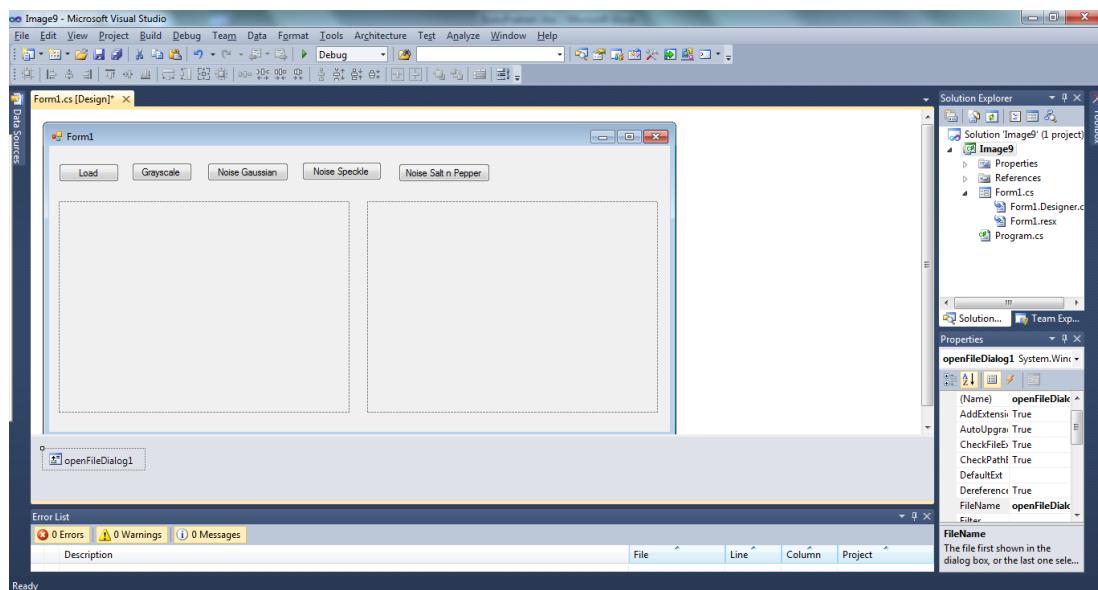


Gambar 9.3. Noise Salt & Pepper

9.2. Petunjuk Praktikum

Langkah-langkah berikut adalah pembuatan program untuk menghasilkan noise tiruan baik berupa noise Gaussian, noise speckle dan noise salt & pepper.

- (1) Buka Visual Studio .Net 2010. Buat project baru dengan File→Project→New Project.
- (2) Pilih Visual C# [Windows Form Application]. Beri nama “Image9” dan Solution name mengikuti dari nama, dan tekan [Ok].
- (3) Setelah keluar form baru dari project yang dibuat, pilih komponen 5 buah button, 2 buah Picturebox dan 1 buah openFileDialog.



Gambar 9.4. Layout project Image9

- (4) Pada Button1, ubah text menjadi “Load”. Pada Button2, ubah text menjadi “Grayscale”. Pada Button3, ubah text menjadi “Noise Gaussian”. Pada Button4, ubah text menjadi “Noise Speckle”. Pada Button5, ubah text menjadi “Noise Salt n Pepper”. Pada PictureBox1, atur size modanya dengan StretchImage. Atur tampilannya seperti gambar 9.4 di atas.
- (5) Tambahkan obyek Bitmap di Class Form1, yaitu objBitmap dan objBitmap1;

```
Bitmap objBitmap;
Bitmap objBitmap1;
```

- (6) Double click pada button1 (Load), tambahkan program berikut:

```

    DialogResult d = openFileDialog1.ShowDialog();
    if (d == DialogResult.OK)
    {
        objBitmap = new Bitmap(openFileDialog1.FileName);
        pictureBox1.Image = objBitmap;
    }

```

(7) Double click pada button2 (Grayscale), tambahkan program berikut:

```

for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int r = w.R;
        int g = w.G;
        int b = w.B;
        int xg = (int)((r + g + b) / 3);
        Color wb = Color.FromArgb(xg, xg, xg);
        objBitmap.SetPixel(x, y, wb);
    }
pictureBox1.Image = objBitmap;

```

(8) Program berikut adalah menambahkan noise gaussian pada gambar dengan 20% noise. Double click pada button3 (noise gaussian) dan tambahkan program berikut:

```

objBitmap1 = new Bitmap(objBitmap);
Random r = new Random();
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int xg = w.R;
        int xb = xg;
        int nr = r.Next(0, 100);
        if (nr < 20)
        {
            int ns = r.Next(0, 256) - 128;
            xb = (int)(xg + ns);
            if (xb < 0) xb = -xb;
            if (xb > 255) xb = 255;
        }
        Color wb = Color.FromArgb(xb, xb, xb);
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;

```

58

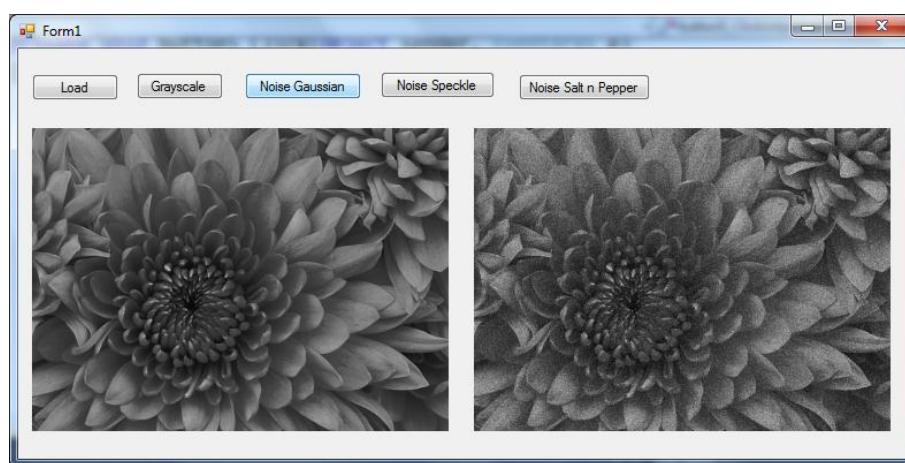
(9) Program berikut adalah menambahkan noise gaussian pada gambar dengan 20% noise. Double click pada button4 (noise speckle) dan tambahkan program berikut:

```
objBitmap1 = new Bitmap(objBitmap);
Random r = new Random();
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int xg = w.R;
        int xb = xg;
        int nr = r.Next(0, 100);
        if (nr < 20) xb = 0;
        Color wb = Color.FromArgb(xb, xb, xb);
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;
```

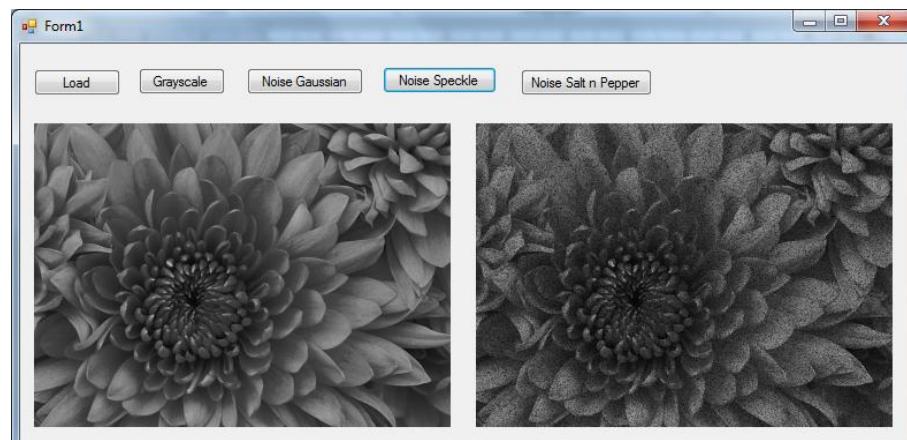
- (10) Program berikut adalah menambahkan noise gaussian pada gambar dengan 20% noise. Double click pada button5 (noise salt n pepper) dan tambahkan program berikut:

```
objBitmap1 = new Bitmap(objBitmap);
Random r = new Random();
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int xg = w.R;
        int xb = xg;
        int nr = r.Next(0, 100);
        if (nr < 20) xb = 255;
        Color wb = Color.FromArgb(xb, xb, xb);
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;
```

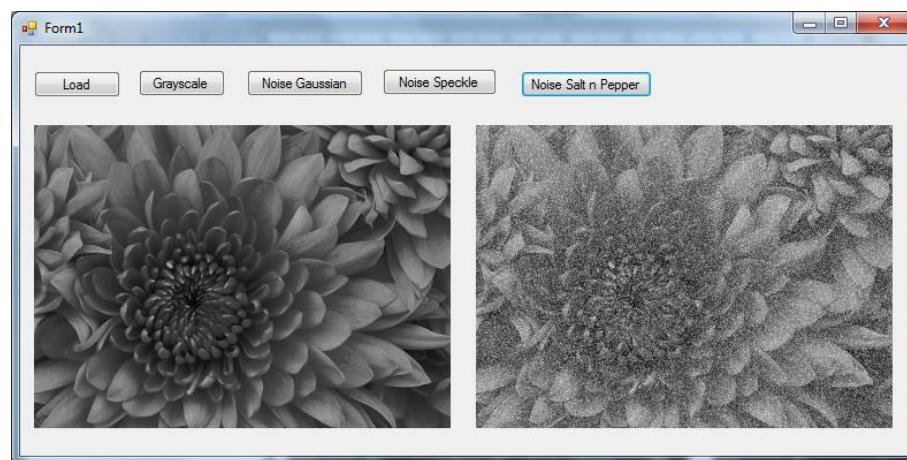
Hasil dari program di atas adalah:



Gambar 9.5. Noise Gaussian



Gambar 9.6. Noise Speckle



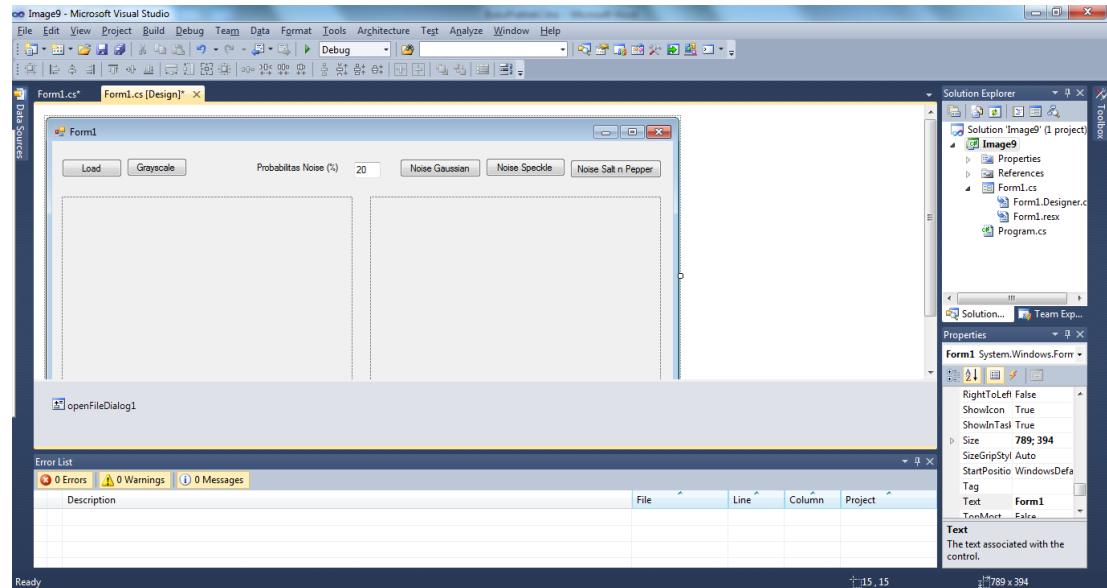
Gambar 9.7. Noise Salt & Pepper

9.3. Laporan Praktikum

Dari program di atas, tulis dan jawab pertanyaan-pertanyaan berikut dalam bentuk laporan praktikum:

- (1) Tuliskan semua kode program dan jelaskan bagian-bagian yang dianggap penting.
- (2) Pada program di atas, probabilitas kemunculan noise sudah ditetapkan 20%. Tambahkan textbox untuk mengubah nilai probabilitas kemunculan noise mulai dari 0% dan 100%, dan buat programnya. Atur layoutnya seperti pada gambar 9.7 di bawah ini.
- (3) Tampilkan hasil noise gaussian dengan 5%, 10%, 20%, 30% dan 40%.

- (4) Tampilkan hasil noise speckle dengan 5%, 10%, 20%, 30% dan 40%.
- (5) Tampilkan hasil noise salt & pepper dengan 5%, 10%, 20%, 30% dan 40%.



Gambar 9.7. Layout dengan probabilitas noise

Bab 10

Noise Reduction

10.1. Dasar Teori

Noise reduction adalah proses untuk mengurangi noise menggunakan Low Pass Filter (LPF). Ada 3 filer yang bisa digunakan untuk mengurangi noise pada gambar, yaitu filter rata-rata, filter gaussian dan filter median.

Hal yang perlu kita ingat pada kernel filter untuk noise reduction adalah: filter yang digunakan adalah LPF dengan sifat-sifat kernel sebagai berikut:

- Semua nilai kernel nol atau positif: $h_{ij} \geq 0$
- Total nilai kernel sama dengan satu: $\sum \sum h_{ij} = 0$

10.1.1. Filter Rata-rata

Filter rata-rata adalah filter yang nilai-nilai kernelnya seragam sesuai dengan jumlah titik kernelnya. Contoh kernel filter rata-rata 3x3 adalah:

$$H = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{ atau } H = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

10.1.2. Filter Gaussian

Filter Gaussian sebenarnya mirip dengan filter rata-rata, adalah filter yang nilai-nilai kernelnya seragam sesuai dengan jumlah titik kernelnya dengan memberikan pembobot lebih pada titik tengahnya. Contoh kernel filter Gaussian 3x3 adalah:

$$H = \frac{1}{13} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{ atau } H = \begin{bmatrix} \frac{1}{13} & \frac{1}{13} & \frac{1}{13} \\ \frac{1}{13} & \frac{4}{13} & \frac{1}{13} \\ \frac{1}{13} & \frac{1}{13} & \frac{1}{13} \end{bmatrix}$$

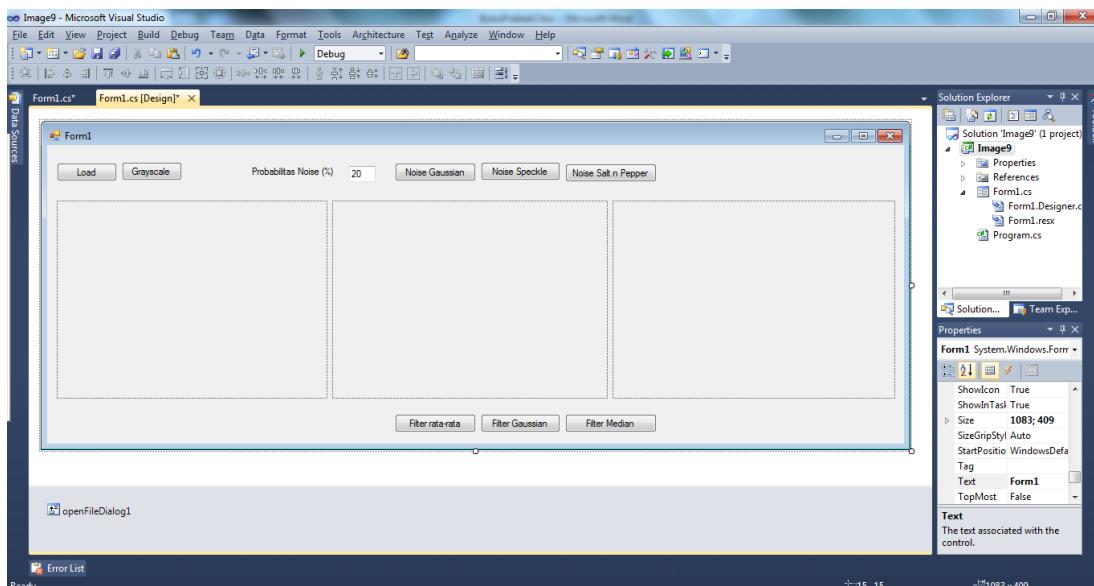
10.1.3. Filter Median

Filter Median adalah filter non linier untuk mengurangi noise. Dalam proses ini, kita mengambil nilai-nilai pada titik (x,y) dan 8 tetangganya. Setelah itu dari nilai-nilai tersebut dicari median (titik-tengahnya). Nilai median ini yang akan mengisi pada titik (x,y) yang baru. Hasilnya memang sangat bagus, namun prosesnya lambat karena perhitungan median menggunakan proses sorting.

10.2. Petunjuk Praktikum

Langkah-langkah berikut adalah pembuatan program untuk mereduksi noise.

- (1) Buka Visual Studio .Net 2010. Buka project “Image9” dengan File→Project→Open Project.
- (2) Setelah keluar form baru dari project yang dibuat, tambahkan 3 komponen button dan 1 buah picturebox seperti gambar 10.1.



Gambar 10.1. Layout project untuk reduksi noise

- (3) Pada Button6, ubah text menjadi “Filter Rata-rata”. Pada Button7, ubah text menjadi “Filter Gaussian”. Pada Button8, ubah text menjadi “Filter Median”. Pada Picturebox2, atur size modenya menjadi StretchImage.
- (4) Tambahkan obyek Bitmap di Class Form1, yaitu objBitmap2.

```
Bitmap objBitmap2;
```

(5) Double click pada button6 (Filter rata-rata), tambahkan program berikut:

```
objBitmap2 = new Bitmap(objBitmap1);
for (int x = 1; x < objBitmap.Width - 1; x++)
    for (int y = 1; y < objBitmap.Height - 1; y++)
    {
        Color w1 = objBitmap1.GetPixel(x - 1, y - 1);
        Color w2 = objBitmap1.GetPixel(x - 1, y);
        Color w3 = objBitmap1.GetPixel(x - 1, y + 1);
        Color w4 = objBitmap1.GetPixel(x, y - 1);
        Color w5 = objBitmap1.GetPixel(x, y);
        Color w6 = objBitmap1.GetPixel(x, y + 1);
        Color w7 = objBitmap1.GetPixel(x + 1, y - 1);
        Color w8 = objBitmap1.GetPixel(x + 1, y);
        Color w9 = objBitmap1.GetPixel(x + 1, y + 1);
        int x1 = w1.R;
        int x2 = w2.R;
        int x3 = w3.R;
        int x4 = w4.R;
        int x5 = w5.R;
        int x6 = w6.R;
        int x7 = w7.R;
        int x8 = w8.R;
        int x9 = w9.R;
        int xb = (int)((x1 + x2 + x3 + x4 + x5 + x6 +
x7 + x8 + x9) / 9);
        if (xb < 0) xb = 0;
        if (xb > 255) xb = 255;
        Color wb = Color.FromArgb(xb, xb, xb);
        objBitmap2.SetPixel(x, y, wb);
    }
pictureBox3.Image = objBitmap2;
```

(6) Double click pada button7 (Filter Gaussian), tambahkan program berikut:

```
objBitmap2 = new Bitmap(objBitmap1);
for (int x = 1; x < objBitmap.Width - 1; x++)
    for (int y = 1; y < objBitmap.Height - 1; y++)
    {
        Color w1 = objBitmap1.GetPixel(x - 1, y - 1);
        Color w2 = objBitmap1.GetPixel(x - 1, y);
        Color w3 = objBitmap1.GetPixel(x - 1, y + 1);
        Color w4 = objBitmap1.GetPixel(x, y - 1);
        Color w5 = objBitmap1.GetPixel(x, y);
        Color w6 = objBitmap1.GetPixel(x, y + 1);
        Color w7 = objBitmap1.GetPixel(x + 1, y - 1);
        Color w8 = objBitmap1.GetPixel(x + 1, y);
        Color w9 = objBitmap1.GetPixel(x + 1, y + 1);
        int x1 = w1.R;
```

```

        int x2 = w2.R;
        int x3 = w3.R;
        int x4 = w4.R;
        int x5 = w5.R;
        int x6 = w6.R;
        int x7 = w7.R;
        int x8 = w8.R;
        int x9 = w9.R;
        int xb = (int)((x1 + x2 + x3 + x4 + 4 * x5 + x6
+ x7 + x8 + x9) / 13);
        if (xb < 0) xb = 0;
        if (xb > 255) xb = 255;
        Color wb = Color.FromArgb(xb, xb, xb);
        objBitmap2.SetPixel(x, y, wb);
    }
pictureBox3.Image = objBitmap2;

```

(7) Double click pada button8 (Filter median), tambahkan program berikut:

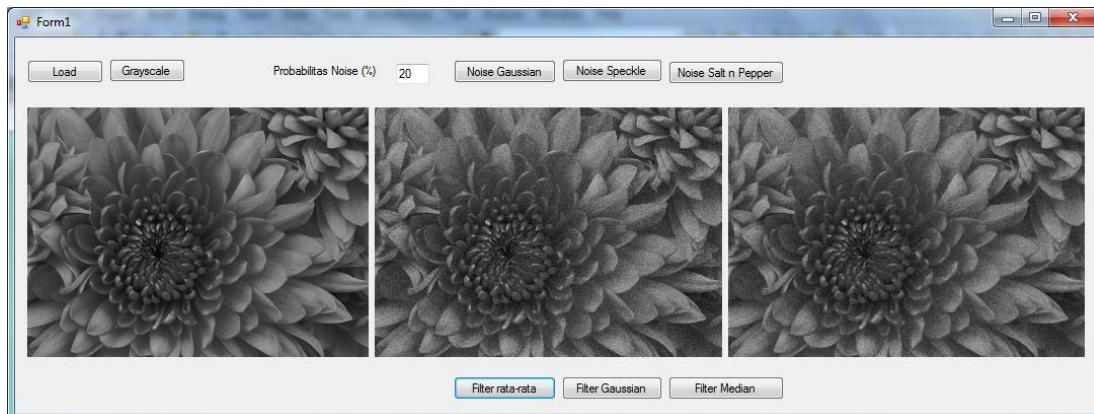
```

int[] xt = new int[10];
objBitmap2 = new Bitmap(objBitmap1);
for (int x = 1; x < objBitmap.Width - 1; x++)
    for (int y = 1; y < objBitmap.Height - 1; y++)
    {
        Color w1 = objBitmap1.GetPixel(x - 1, y - 1);
        Color w2 = objBitmap1.GetPixel(x - 1, y);
        Color w3 = objBitmap1.GetPixel(x - 1, y + 1);
        Color w4 = objBitmap1.GetPixel(x, y - 1);
        Color w5 = objBitmap1.GetPixel(x, y);
        Color w6 = objBitmap1.GetPixel(x, y + 1);
        Color w7 = objBitmap1.GetPixel(x + 1, y - 1);
        Color w8 = objBitmap1.GetPixel(x + 1, y);
        Color w9 = objBitmap1.GetPixel(x + 1, y + 1);
        xt[1] = w1.R;
        xt[2] = w2.R;
        xt[3] = w3.R;
        xt[4] = w4.R;
        xt[5] = w5.R;
        xt[6] = w6.R;
        xt[7] = w7.R;
        xt[8] = w8.R;
        xt[9] = w9.R;
        for(int i=1; i < 9; i++)
            for (int j = 1; j < 9; j++)
            {
                if (xt[j] > xt[j + 1])
                {
                    int a = xt[j];
                    xt[j] = xt[j + 1];
                    xt[j + 1] = a;
                }
            }
    }

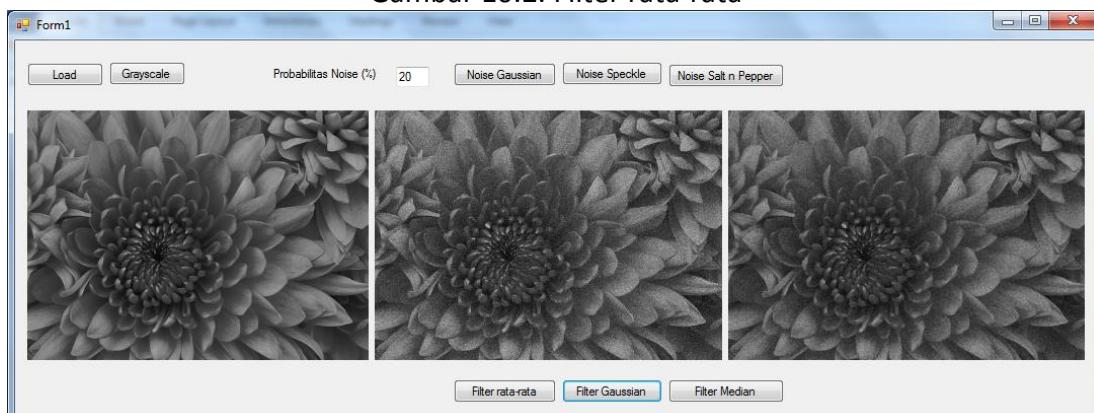
```

```
int xb = xt[5];
Color wb = Color.FromArgb(xb, xb, xb);
objBitmap2.SetPixel(x, y, wb);
}
pictureBox3.Image = objBitmap2;
```

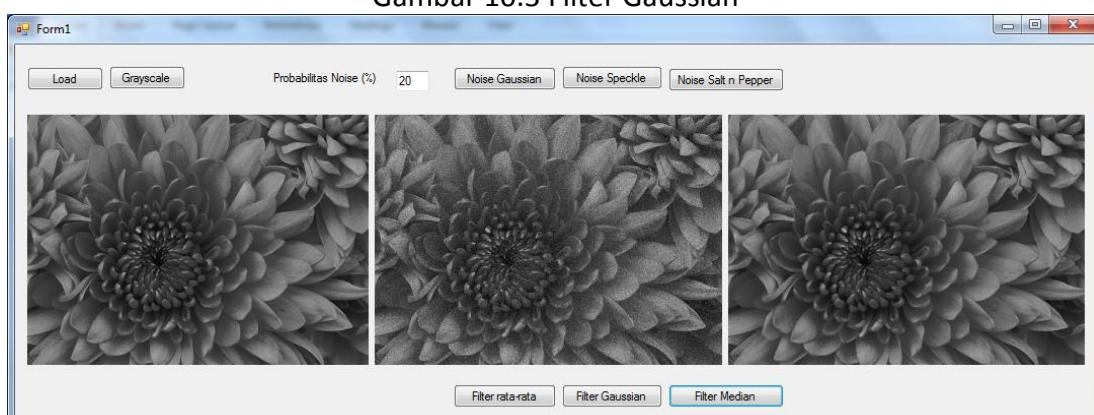
Hasil dari program reduksi noise ini adalah sebagai berikut. Noise yang digunakan adalah noise gaussian dengan prosentase 20%.



Gambar 10.2. Filter rata-rata



Gambar 10.3 Filter Gaussian



Gambar 10.4 Filter Median

10.3. Laporan Praktikum

Dari program di atas, tulis dan jawab pertanyaan-pertanyaan berikut dalam bentuk laporan praktikum:

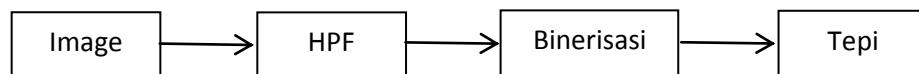
- (1) Tuliskan semua kode program dan jelaskan bagian-bagian yang dianggap penting.
- (2) Tambahkan 5%, 10% atau 20% noise gaussian. Kemudian lakukan reduksi noise dengan filter rata-rata. Apa yang bisa dilihat dari hasil filter tersebut?
- (3) Tambahkan 5%, 10% atau 20% noise speckle. Kemudian lakukan reduksi noise dengan filter rata-rata. Apa yang bisa dilihat dari hasil filter tersebut?
- (4) Tambahkan 5%, 10% atau 20% noise salt & papper. Kemudian lakukan reduksi noise dengan filter rata-rata. Apa yang bisa dilihat dari hasil filter tersebut?
- (5) Tambahkan 5%, 10% atau 20% noise gaussian. Kemudian lakukan reduksi noise dengan filter gaussian. Apa yang bisa dilihat dari hasil filter tersebut?
- (6) Tambahkan 5%, 10% atau 20% noise speckle. Kemudian lakukan reduksi noise dengan filter gaussian. Apa yang bisa dilihat dari hasil filter tersebut?
- (7) Tambahkan 5%, 10% atau 20% noise salt & papper. Kemudian lakukan reduksi noise dengan filter gaussian. Apa yang bisa dilihat dari hasil filter tersebut?
- (8) Tambahkan 5%, 10% atau 20% noise gaussian. Kemudian lakukan reduksi noise dengan filter median. Apa yang bisa dilihat dari hasil filter tersebut?
- (9) Tambahkan 5%, 10% atau 20% noise speckle. Kemudian lakukan reduksi noise dengan filter median. Apa yang bisa dilihat dari hasil filter tersebut?
- (10) Tambahkan 5%, 10% atau 20% noise salt & papper. Kemudian lakukan reduksi noise dengan filter median. Apa yang bisa dilihat dari hasil filter tersebut?

Bab 11

Deteksi Tepi

11.1. Dasar Teori

Deteksi tepi adalah sebuah proses untuk mendapatkan informasi tepi dari sebuah gambar. Hasil dari deteksi tepi ini seperti sebuah sketsa dari gambar. Proses deteksi tepi memanfaatkan high pass filter dan urutannya bisa digambarkan sebagai berikut:



Gambar 11.1. Proses deteksi tepi

Ada beberapa metode yang bisa digunakan untuk proses deteksi tepi, antara lain: metode Robert, metode Prewitt, Metode Sobel dan metode Laplacian.

11.1.1. Metode Robert

Metode Robert merupakan metode dasar dengan memanfaatkan filter baris atau filter kolom:

$$H = [-1 \ 1] \rightarrow \text{untuk filter horisontal}$$

$$H = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \rightarrow \text{untuk filter vertikal}$$

11.1.2. Metode Prewitt

Metode Prewitt merupakan metode deteksi tepi dengan memanfaatkan filter kernel 3x3 sebagai berikut:

$$H = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \rightarrow \text{untuk filter horisontal}$$

$$H = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \rightarrow \text{untuk filter vertikal}$$

11.1.3. Metode Sobel

Metode Sobel merupakan metode deteksi tepi dengan memanfaatkan filter kernel 3x3 sebagai berikut:

$$H = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \rightarrow \text{untuk filter horisontal}$$

$$H = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \rightarrow \text{untuk filter vertikal}$$

11.1.4. Metode Laplacian

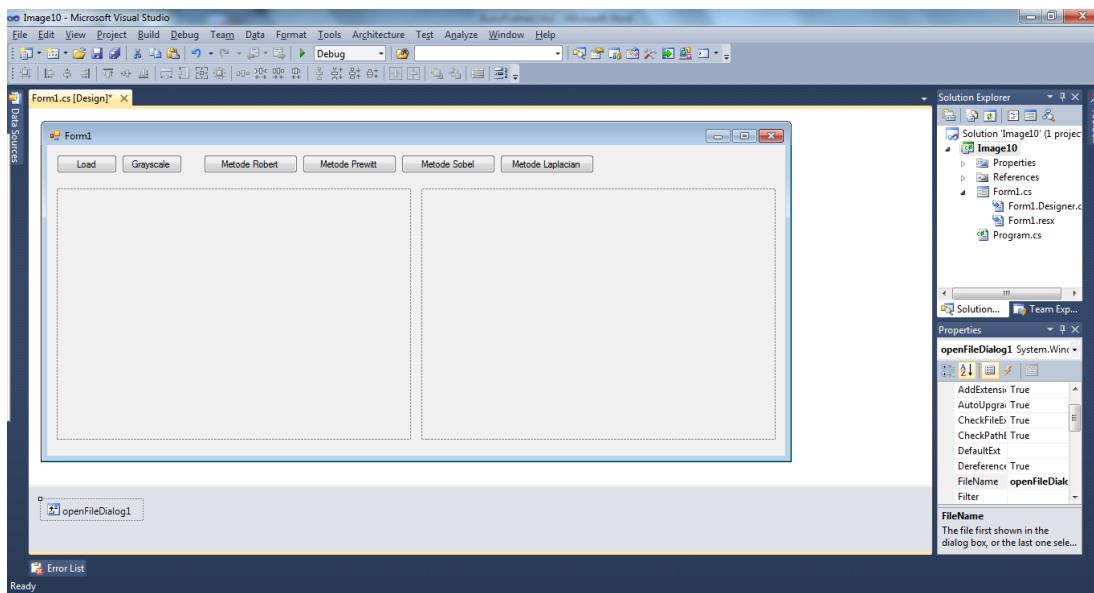
Metode Sobel merupakan metode deteksi tepi dengan memanfaatkan filter kernel 3x3 sebagai berikut. Kernel filter Laplacian tidak memerlukan filter horisontal dan filter vertikal. Filter ini adalah turunan dari filter Gaussian.

$$H = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

11.2. Petunjuk Praktikum

Langkah-langkah berikut adalah pembuatan program untuk menghasilkan deteksi tepi.

- (1) Buka Visual Studio .Net 2010. Buat project baru dengan File→Project→New Project.
- (2) Pilih Visual C# [Windows Form Application]. Beri nama “Image10” dan Solution name mengikuti dari nama, dan tekan [Ok].
- (3) Setelah keluar form baru dari project yang dibuat, pilih komponen 6 buah button, 2 buah Picturebox dan 1 buah openFileDialog.



Gambar 11.1. Layout project Image10

- (4) Pada Button1, ubah text menjadi “Load”. Pada Button2, ubah text menjadi “Grayscale”. Pada Button3, ubah text menjadi “Metode Robert”. Pada Button4, ubah text menjadi “Metode Prewitt”. Pada Button5, ubah text menjadi “Metode Sobel”. Pada Button6, ubah text menjadi “Metode Laplacian”. Pada Picturebox1, atur size modenya dengan StretchImage. Atur tampilannya seperti gambar 10.1 di atas.

- (5) Tambahkan obyek Bitmap di Class Form1, yaitu objBitmap dan objBitmap1;

```
Bitmap objBitmap;
Bitmap objBitmap1;
```

- (6) Double click pada button1 (Load), tambahkan program berikut:

```
DialogResult d = openFileDialog1.ShowDialog();
if (d == DialogResult.OK)
{
    objBitmap = new Bitmap(openFileDialog1.FileName);
    pictureBox1.Image = objBitmap;
}
```

- (7) Double click pada button2 (Grayscale), tambahkan program berikut:

```
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int r = w.R;
        int g = w.G;
```

```

        int b = w.B;
        int xg = (int)((r + g + b) / 3);
        Color wb = Color.FromArgb(xg, xg, xg);
        objBitmap.SetPixel(x, y, wb);
    }
pictureBox1.Image = objBitmap;

```

(8) Double click pada button3 (Metode Robert), tambahkan program berikut:

```

objBitmap1 = new Bitmap(objBitmap);
for (int x = 1; x < objBitmap.Width; x++)
    for (int y = 1; y < objBitmap.Height; y++)
    {
        Color w1 = objBitmap.GetPixel(x - 1, y);
        Color w2 = objBitmap.GetPixel(x, y);
        Color w3 = objBitmap.GetPixel(x, y - 1);
        Color w4 = objBitmap.GetPixel(x, y);
        int x1 = w1.R;
        int x2 = w2.R;
        int x3 = w3.R;
        int x4 = w4.R;
        int xb = (int)((x2 - x1) + (x4 - x3));
        if (xb < 0) xb = -xb;
        if (xb > 255) xb = 255;
        Color wb = Color.FromArgb(xb, xb, xb);
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;

```

(9) Double click pada button4 (Metode Prewitt), tambahkan program berikut:

```

objBitmap1 = new Bitmap(objBitmap);
for (int x = 1; x < objBitmap.Width - 1; x++)
    for (int y = 1; y < objBitmap.Height - 1; y++)
    {
        Color w1 = objBitmap.GetPixel(x - 1, y - 1);
        Color w2 = objBitmap.GetPixel(x - 1, y);
        Color w3 = objBitmap.GetPixel(x - 1, y + 1);
        Color w4 = objBitmap.GetPixel(x, y - 1);
        Color w5 = objBitmap.GetPixel(x, y);
        Color w6 = objBitmap.GetPixel(x, y + 1);
        Color w7 = objBitmap.GetPixel(x + 1, y - 1);
        Color w8 = objBitmap.GetPixel(x + 1, y);
        Color w9 = objBitmap.GetPixel(x + 1, y + 1);
        int x1 = w1.R;
        int x2 = w2.R;
        int x3 = w3.R;
        int x4 = w4.R;
        int x5 = w5.R;
        int x6 = w6.R;
        int x7 = w7.R;
        int x8 = w8.R;
        int x9 = w9.R;

```

```

        int xh = (int)(-x1 - x4 - x7 + x3 + x6 + x9);
        int xv = (int)(-x1 - x2 - x3 + x7 + x8 + x9);
        int xb=(int)(xh + xv);
        if (xb < 0) xb = -xb;
        if (xb > 255) xb = 255;
        Color wb = Color.FromArgb(xb, xb, xb);
        objBitmap1.SetPixel(x, y, wb);
    }
    pictureBox2.Image = objBitmap1;

```

- (10) Double click pada button5 (Metode Sobel), tambahkan program berikut:

```

objBitmap1 = new Bitmap(objBitmap);
for (int x = 1; x < objBitmap.Width - 1; x++)
    for (int y = 1; y < objBitmap.Height - 1; y++)
    {
        Color w1 = objBitmap.GetPixel(x - 1, y - 1);
        Color w2 = objBitmap.GetPixel(x - 1, y);
        Color w3 = objBitmap.GetPixel(x - 1, y + 1);
        Color w4 = objBitmap.GetPixel(x, y - 1);
        Color w5 = objBitmap.GetPixel(x, y);
        Color w6 = objBitmap.GetPixel(x, y + 1);
        Color w7 = objBitmap.GetPixel(x + 1, y - 1);
        Color w8 = objBitmap.GetPixel(x + 1, y);
        Color w9 = objBitmap.GetPixel(x + 1, y + 1);
        int x1 = w1.R;
        int x2 = w2.R;
        int x3 = w3.R;
        int x4 = w4.R;
        int x5 = w5.R;
        int x6 = w6.R;
        int x7 = w7.R;
        int x8 = w8.R;
        int x9 = w9.R;
        int xh = (int)(-x1-2*x4 - x7 + x3 + 2*x6 + x9);
        int xv = (int)(-x1-2*x2 - x3 + x7 + 2*x8 + x9);
        int xb = (int)(xh + xv);
        if (xb < 0) xb = -xb;
        if (xb > 255) xb = 255;
        Color wb = Color.FromArgb(xb, xb, xb);
        objBitmap1.SetPixel(x, y, wb);
    }
    pictureBox2.Image = objBitmap1;

```

- (11) Double click pada button3 (Metode Laplacian), tambahkan program berikut:

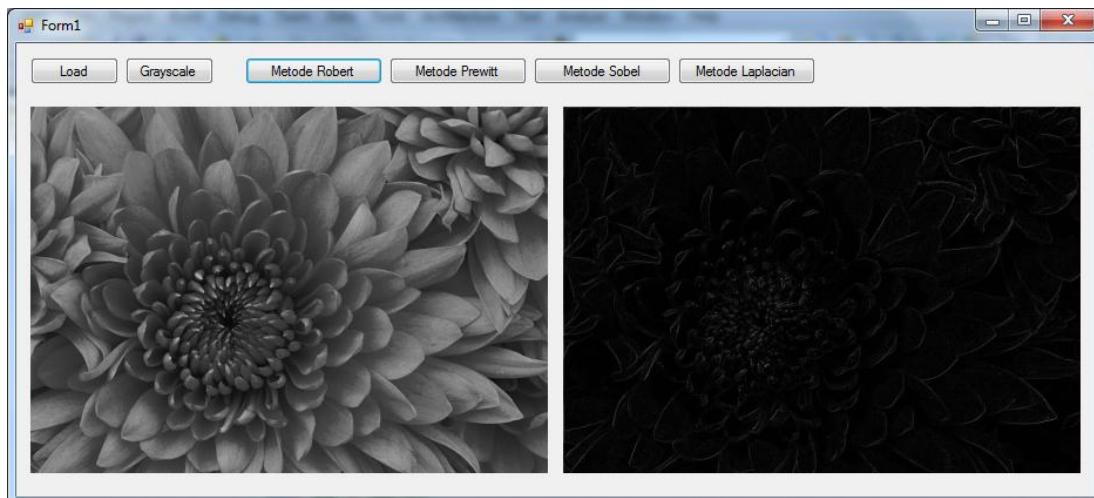
```

objBitmap1 = new Bitmap(objBitmap);
for (int x = 1; x < objBitmap.Width - 1; x++)
    for (int y = 1; y < objBitmap.Height - 1; y++)
    {
        Color w1 = objBitmap.GetPixel(x - 1, y - 1);
        Color w2 = objBitmap.GetPixel(x - 1, y);
        Color w3 = objBitmap.GetPixel(x - 1, y + 1);

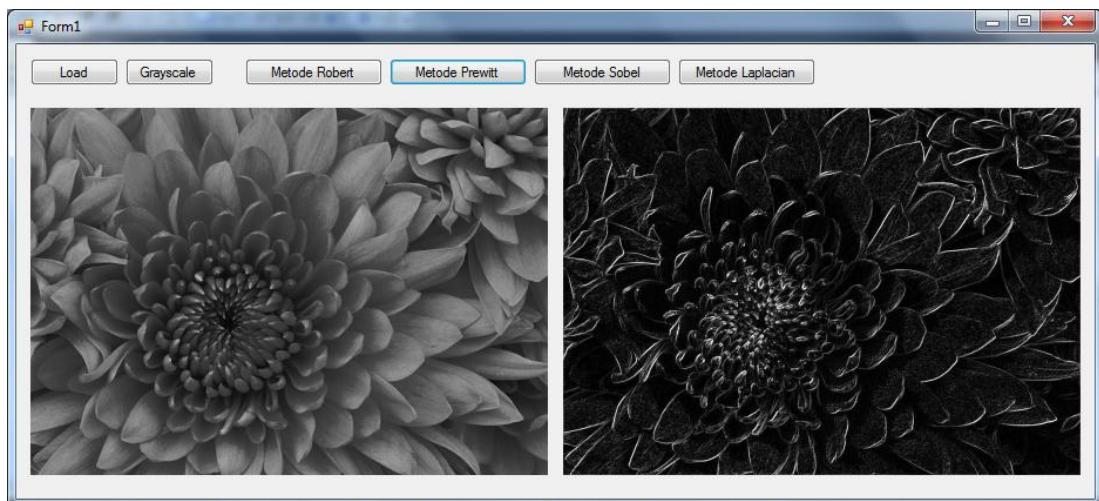
```

```
Color w4 = objBitmap.GetPixel(x, y - 1);
Color w5 = objBitmap.GetPixel(x, y);
Color w6 = objBitmap.GetPixel(x, y + 1);
Color w7 = objBitmap.GetPixel(x + 1, y - 1);
Color w8 = objBitmap.GetPixel(x + 1, y);
Color w9 = objBitmap.GetPixel(x + 1, y + 1);
int x1 = w1.R;
int x2 = w2.R;
int x3 = w3.R;
int x4 = w4.R;
int x5 = w5.R;
int x6 = w6.R;
int x7 = w7.R;
int x8 = w8.R;
int x9 = w9.R;
int xb = (int)(x1-2*x2+x3-2*x4+4*x5-2*x6+x7-
2*x8+x9);
if (xb < 0) xb = -xb;
if (xb > 255) xb = 255;
Color wb = Color.FromArgb(xb, xb, xb);
objBitmap1.SetPixel(x, y, wb);
}
pictureBox2.Image = objBitmap1;
```

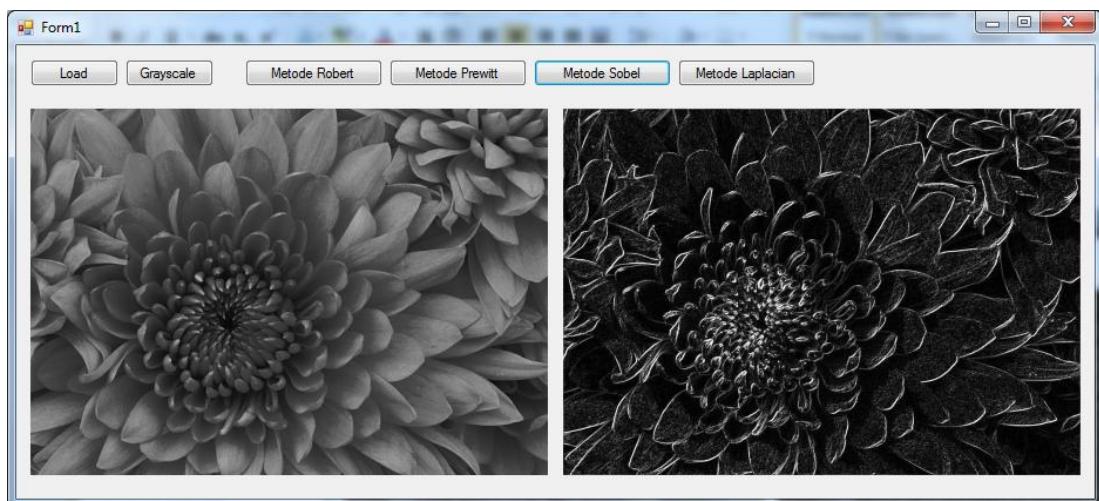
Hasil program di atas adalah sebagai berikut:



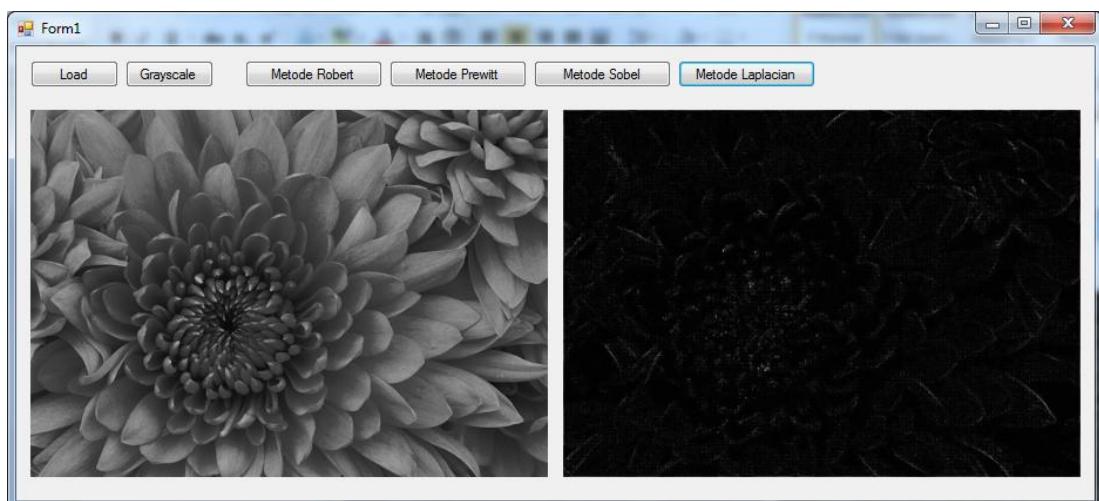
Gambar 10.2. Metode Robert



Gambar 10.3. Metode Prewitt



Gambar 10.4. Metode Sobel



Gambar 10.5. Metode Laplacian

11.3. Laporan Praktikum

Dari program di atas, tulis dan jawab pertanyaan-pertanyaan berikut dalam bentuk laporan praktikum:

- (1) Tuliskan semua kode program dan jelaskan bagian-bagian yang dianggap penting.
- (2) Tambahkan proses binerisasi untuk setiap metode deteksi tepi.
- (3) Tambahkan proses invers untuk setiap metode deteksi tepi untuk mendapatkan sketsa sebuah gambar.
- (4) Jelaskan apa perbedaan antara metode Prewitt dan metode Sobel?

Bab 12

Sharpness

12.1. Dasar Teori

Sharpness adalah proses untuk mendapatkan gambar yang lebih tajam. Proses sharpness ini memanfaatkan BSF (Band-Stop Filter) yang merupakan gabungan antara LPF (Low Pass Filter) dan HPF (High Pass Filter).

Ciri-ciri kernel filter yang bisa digunakan untuk proses sharpness adalah sebagai berikut:

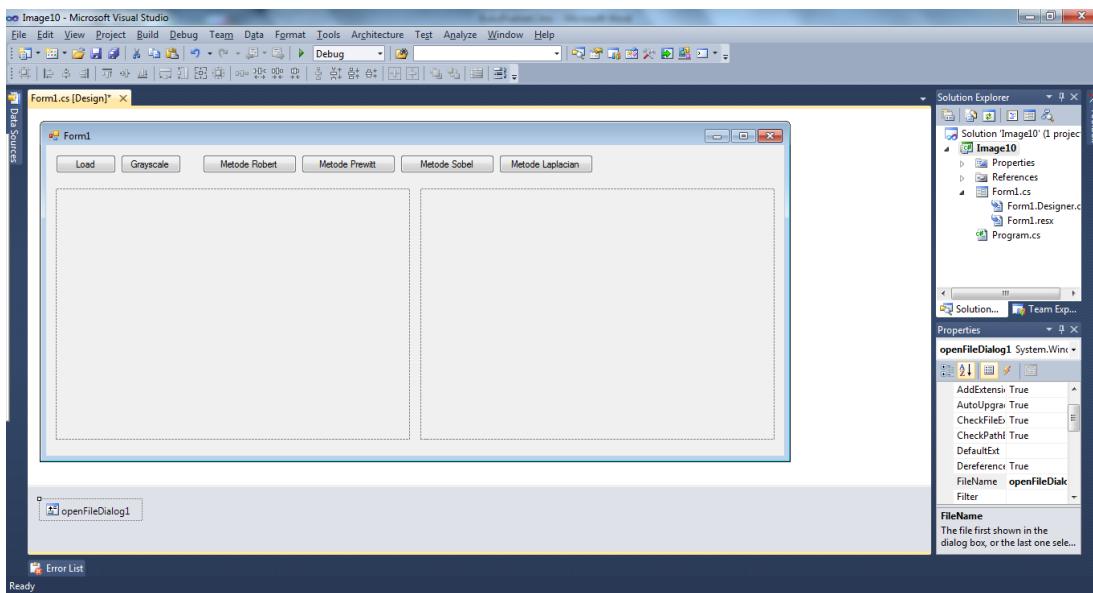
- (1) Nilai di dalam filter kernel harus ada nilai positif dan nilai negatif.
- (2) Jumlah nilai dalam filter kernel harus sama dengan satu.

Contoh filter kernel yang digunakan untuk proses Sharpness dengan memanfaatkan filter rata-rata dan filter sobel. Perbandingan dalam pemakaian filter rata-rata dan filter sobel menjadi kunci untuk menghasilkan tingkat ketajaman gambar.

12.2. Petunjuk Praktikum

Langkah-langkah berikut adalah pembuatan program untuk proses sharpness.

- (1) Buka Visual Studio .Net 2010. Buat project baru dengan File→Project→New Project.
- (2) Pilih Visual C# [Windows Form Application]. Beri nama “Image11” dan Solution name mengikuti dari nama, dan tekan [Ok].
- (3) Setelah keluar form baru dari project yang dibuat, pilih komponen 3 buah button, 2 buah Picturebox dan 1 buah openFileDialog.



Gambar 12.1. Layout project Image11

- (4) Pada Button1, ubah text menjadi “Load”. Pada Button2, ubah text menjadi “Grayscale”. Pada Button3, ubah text menjadi “Sharpness”. Pada Picturebox1, atur size modenya dengan StretchImage. Atur tampilannya seperti gambar 10.1 di atas.
- (5) Tambahkan obyek Bitmap di Class Form1, yaitu objBitmap dan objBitmap1.

```
Bitmap objBitmap;
Bitmap objBitmap1;
```

- (6) Double click pada button1 (Load), tambahkan program berikut:

```
DialogResult d = openFileDialog1.ShowDialog();
if (d == DialogResult.OK)
{
    objBitmap = new Bitmap(openFileDialog1.FileName);
    pictureBox1.Image = objBitmap;
}
```

- (7) Double click pada button2 (Grayscale), tambahkan program berikut:

```
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int r = w.R;
        int g = w.G;
        int b = w.B;
        int xg = (int)((r + g + b) / 3);
        Color wb = Color.FromArgb(xg, xg, xg);
        objBitmap.SetPixel(x, y, wb);
    }
```

```
pictureBox1.Image = objBitmap;
```

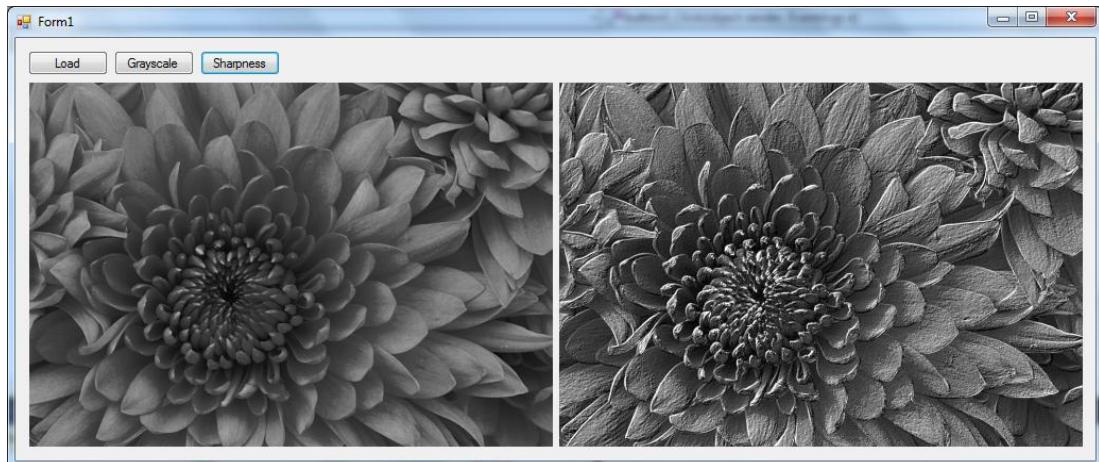
(8) Double click pada button2 (Sharpness), tambahkan program berikut:

```
objBitmap1 = new Bitmap(objBitmap);
for (int x = 1; x < objBitmap.Width-1; x++)
    for (int y = 1; y < objBitmap.Height-1; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int xg = w.R;
        Color w1 = objBitmap.GetPixel(x - 1, y - 1);
        Color w2 = objBitmap.GetPixel(x - 1, y);
        Color w3 = objBitmap.GetPixel(x - 1, y + 1);
        Color w4 = objBitmap.GetPixel(x, y - 1);
        Color w5 = objBitmap.GetPixel(x, y);
        Color w6 = objBitmap.GetPixel(x, y + 1);
        Color w7 = objBitmap.GetPixel(x + 1, y - 1);
        Color w8 = objBitmap.GetPixel(x + 1, y);
        Color w9 = objBitmap.GetPixel(x + 1, y + 1);
        int x1 = w1.R;
        int x2 = w2.R;
        int x3 = w3.R;
        int x4 = w4.R;
        int x5 = w5.R;
        int x6 = w6.R;
        int x7 = w7.R;
        int x8 = w8.R;
        int x9 = w9.R;
        int xt1 = (int)((x1 + x2 + x3 + x4 + x5 + x6 +
x7 + x8 + x9)/9);
        int xt2 = (int)(-x1 - 2*x2 - x3 + x7 + 2*x8 +
x9);
        int xt3 = (int)(-x1 - 2 * x4 - x7 + x3+ 2 * x6
+ x9);
        int xb = (int)(xt1 + xt2+xt3);
        if (xb < 0) xb = -xb;
        if (xb > 255) xb = 255;
        Color wb = Color.FromArgb(xb, xb, xb);
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;
```

Dari program di atas terlihat bahwa proses sharpness adalah penggabungan proses Low Pass Filter (LPF) dan High Pass Filter (HPF). Dengan proses sharpness ini, gambar akan menjadi semakin tajam.

78

Hasil program di atas adalah sebagai berikut:



Gambar 12.2. Sharpness

12.3. Laporan Praktikum

Dari program di atas, tulis dan jawab pertanyaan-pertanyaan berikut dalam bentuk laporan praktikum:

- (1) Tuliskan semua kode program dan jelaskan bagian-bagian yang dianggap penting.
- (2) Gunakan perbandingan 2:1 untuk LPF dan HPF dengan rumus berikut dan jelaskan hasilnya.

$$xb = \frac{2 * xt1 + xt2 + xt3}{2}$$

- (3) Gunakan perbandingan 1:2 untuk LPF dan HPF dengan rumus berikut dan jelaskan hasilnya.

$$xb = \frac{xt1 + 2 * xt2 + 2 * xt3}{3}$$

Bab 13

Pengolahan Dasar Gambar Berwarna

13.1. Dasar Teori

Pengolahan dasar gambar berwarna seperti brightness, contrast, auto level, dan invers, sebenarnya hampir sama dengan pengolahan gambar derajat keabuan, hanya saja operasionalnya diberlakukan pada setiap nilai R, G dan B.

13.1.1. Brightness

Brightness pada gambar berwarna dengan nilai perubahan a , menggunakan operasional sebagai berikut:

$$r_{baru} = r + a$$

$$g_{baru} = g + a$$

$$b_{baru} = b + a$$

Dimana (r, g, b) adalah warna asal, dan $(r_{baru}, g_{baru}, b_{baru})$ adalah warna hasil brightness.

13.1.2. Contrast

Contrast pada gambar berwarna dengan nilai perubahan contrast c , menggunakan operasional sebagai berikut:

$$r_{baru} = c \cdot r$$

$$g_{baru} = c \cdot g$$

$$b_{baru} = c \cdot b$$

80

Dimana (r, g, b) adalah warna asal, dan $(r_{baru}, g_{baru}, b_{baru})$ adalah warna hasil contrast.

13.1.3. Invers

Invers pada gambar berwarna, menggunakan operasional sebagai berikut:

$$r_{baru} = 255 - r$$

$$g_{baru} = 255 - g$$

$$b_{baru} = 255 - b$$

Dimana (r, g, b) adalah warna asal, dan $(r_{baru}, g_{baru}, b_{baru})$ adalah warna hasil invers.

13.1.4. Auto Level

Auto-level pada gambar berwarna, menggunakan operasional sebagai berikut:

$$r_{baru} = \frac{255}{r_{max} - r_{min}} (r - r_{min})$$

$$g_{baru} = \frac{255}{g_{max} - g_{min}} (g - g_{min})$$

$$b_{baru} = \frac{255}{b_{max} - b_{min}} (b - b_{min})$$

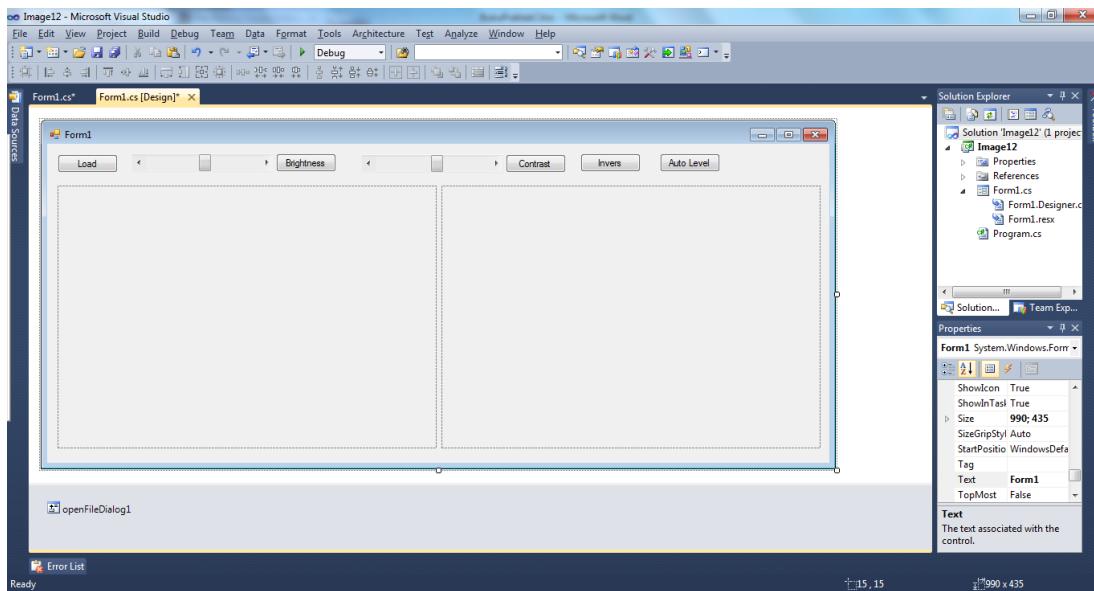
Dimana (r, g, b) adalah warna asal, $(r_{baru}, g_{baru}, b_{baru})$ adalah warna hasil auto-level, r_{max} adalah nilai r maksimum, r_{min} adalah nilai r minimum, g_{max} adalah nilai g maksimum, g_{min} adalah nilai g minimum, b_{max} adalah nilai b maksimum dan b_{min} adalah nilai b minimum.

13.2. Petunjuk Praktikum

Langkah-langkah berikut adalah membuat aplikasi pengolahan gambar berwarna yang terdiri dari Brightness, Contrast, Invers dan Auto-level:

- (1) Buka Visual Studio .Net 2010. Buat project baru dengan File→Project→New Project.
- (2) Pilih Visual C# [Windows Form Application]. Beri nama “Image12” dan Solution name mengikuti dari nama, dan tekan [Ok].

- (3) Setelah keluar form baru dari project yang dibuat, pilih komponen 5 buah button, 2 buah Picturebox, 2 buah horisontal scroll dan 1 buah openFileDialog.



Gambar 13.1. Layout project Image12

- (4) Pada Button1, ubah text menjadi "Load". Pada Button2, ubah text menjadi "Brightness". Pada Button3, ubah text menjadi "Contrast". Pada Button4, ubah text menjadi "Invers". Pada Button5, ubah text menjadi "Auto Level". Pada hScroll1, nilai minimum=-100, nilai maksimum=100, value=0. Pada hScroll2, nilai minimum=10, nilai maksimum=200, value=100. Pada PictureBox1, atur size modonya dengan StretchImage. Atur tampilannya seperti gambar 10.1 di atas.

- (5) Tambahkan obyek Bitmap di Class Form1, yaitu objBitmap dan objBitmap1.

```
Bitmap objBitmap;
Bitmap objBitmap1;
```

- (6) Double click pada button1 (Load), tambahkan program berikut:

```
DialogResult d = openFileDialog1.ShowDialog();
if (d == DialogResult.OK)
{
    objBitmap = new Bitmap(openFileDialog1.FileName);
    pictureBox1.Image = objBitmap;
}
```

- (7) Double click pada button2 (Brightness), tambahkan program berikut:

```
int a = hScrollBar1.Value;
objBitmap1 = new Bitmap(objBitmap);
for (int x = 0; x < objBitmap.Width; x++)
```

```

        for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int r = (int)(w.R + a);
        int g = (int)(w.G + a);
        int b = (int)(w.B + a);
        if (r < 0) r = 0;
        if (r > 255) r = 255;
        if (g < 0) g = 0;
        if (g > 255) g = 255;
        if (b < 0) b = 0;
        if (b > 255) b = 255;
        Color wb = Color.FromArgb(r, g, b);
        objBitmap1.SetPixel(x, y, wb);
    }
    pictureBox2.Image = objBitmap1;
    hScrollBar1.Value = 0;

```

(8) Double click pada button3 (Sharpness), tambahkan program berikut:

```

float a = (float)(hScrollBar2.Value/100.0);
button3.Text = a.ToString();
objBitmap1 = new Bitmap(objBitmap);
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int r = (int)(a * (float)w.R);
        int g = (int)(a * (float)w.G);
        int b = (int)(a * (float)w.B);
        if (r < 0) r = 0;
        if (r > 255) r = 255;
        if (g < 0) g = 0;
        if (g > 255) g = 255;
        if (b < 0) b = 0;
        if (b > 255) b = 255;
        Color wb = Color.FromArgb(r, g, b);
        objBitmap1.SetPixel(x, y, wb);
    }
    pictureBox2.Image = objBitmap1;
    hScrollBar2.Value = 100;

```

(9) Double click pada button4 (Invers), tambahkan program berikut:

```

objBitmap1 = new Bitmap(objBitmap);
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int r = (int)(255-w.R);
        int g = (int)(255-w.G);
        int b = (int)(255-w.B);
        Color wb = Color.FromArgb(r, g, b);

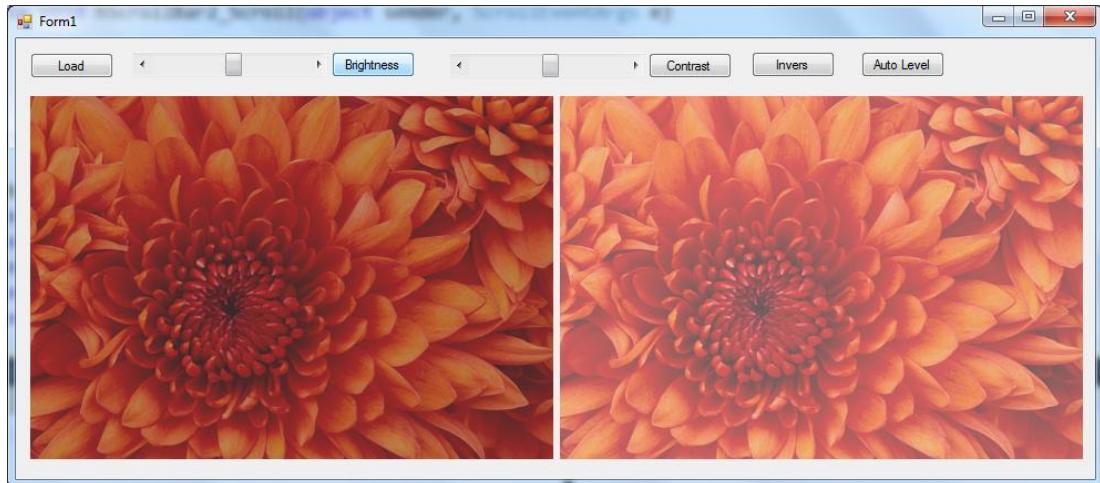
```

```
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;
```

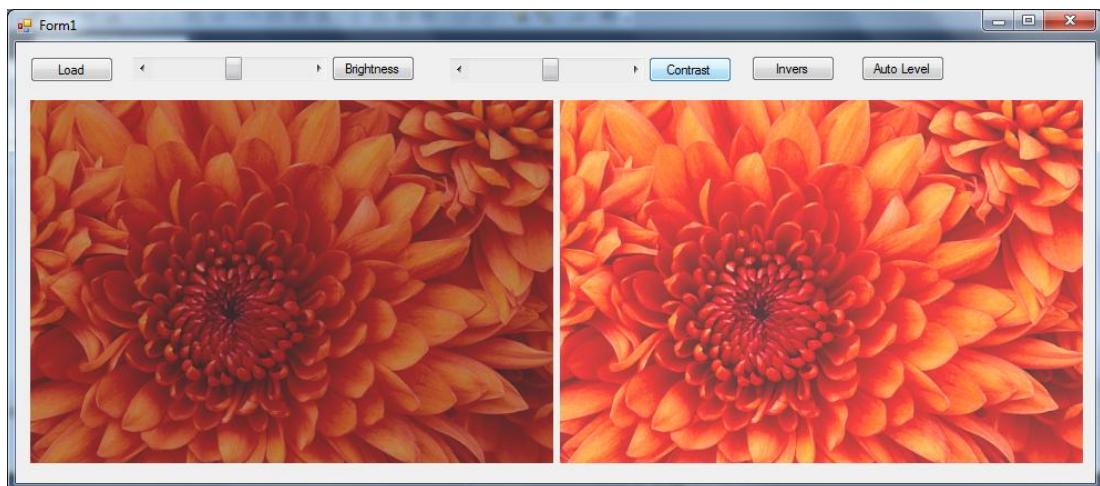
- (10) Double click pada button3 (Auto-Level), tambahkan program berikut:

```
objBitmap1 = new Bitmap(objBitmap);
int rmin = 255;
int gmin = 255;
int bmin = 255;
int rmax = 0;
int gmax = 0;
int bmax = 0;
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int r = w.R;
        int g = w.G;
        int b = w.B;
        if (r < rmin) rmin = r;
        if (r > rmax) rmax = r;
        if (g < gmin) gmin = g;
        if (g > gmax) gmax = g;
        if (b < bmin) bmin = b;
        if (b > bmax) bmax = b;
    }
    for (int x = 0; x < objBitmap.Width; x++)
        for (int y = 0; y < objBitmap.Height; y++)
    {
        Color w = objBitmap.GetPixel(x, y);
        int r = w.R;
        int g = w.G;
        int b = w.B;
        int rbaru = (int)(255 * (r - rmin) / (rmax -
rmin));
        int gbaru = (int)(255 * (g - gmin) / (gmax -
gmin));
        int bbaru = (int)(255 * (b - bmin) / (bmax -
bmin));
        Color wb = Color.FromArgb(rbaru, gbaru, bbaru);
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;
```

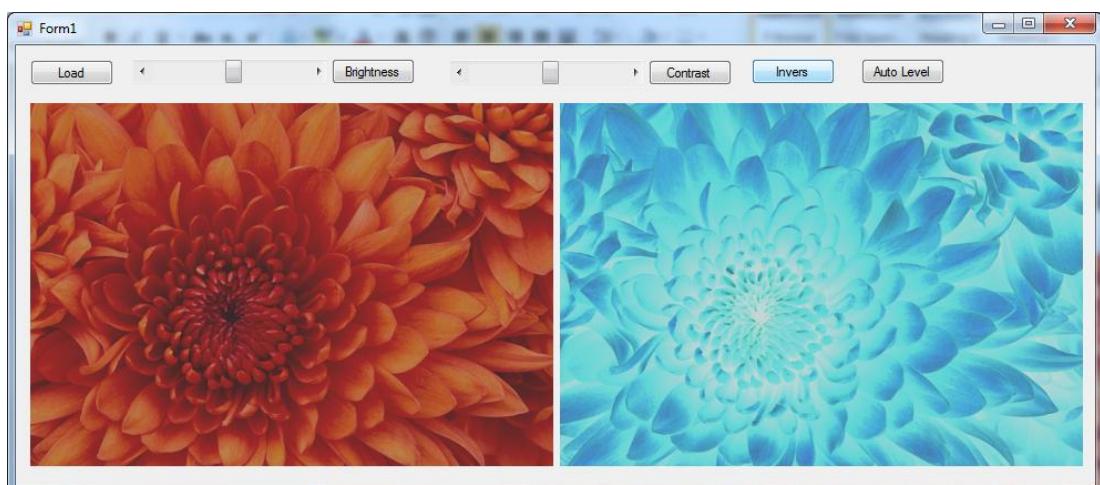
Hasil dari program di atas adalah sebagai berikut:



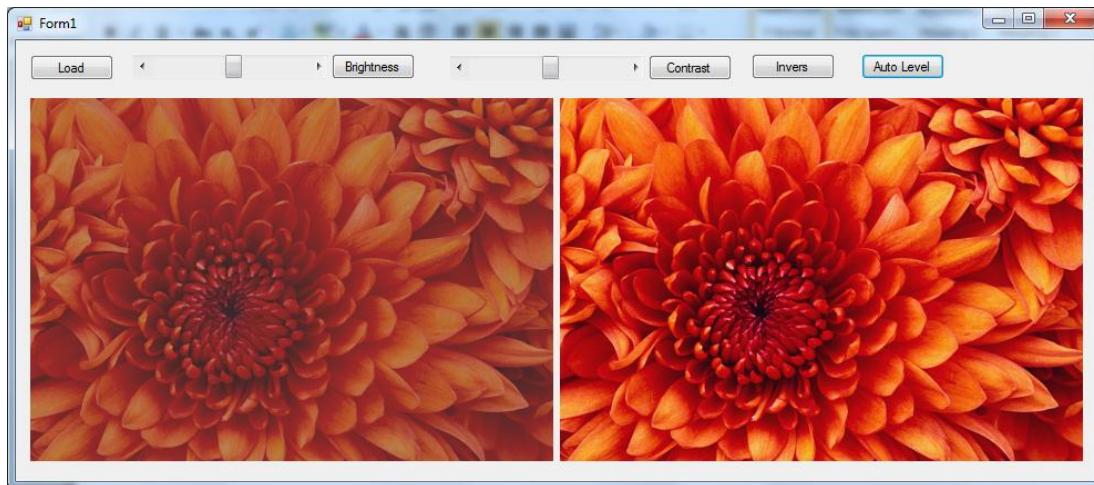
Gambar 13.2. Brightness



Gambar 13.3. Contrast



Gambar 13.4. Invers



Gambar 13.5. Auto level

13.3. Laporan Praktikum

Dari program di atas, tulis dan jawab pertanyaan-pertanyaan berikut dalam bentuk laporan praktikum:

- (1) Tuliskan semua kode program dan jelaskan bagian-bagian yang dianggap penting.
- (2) Lakukan percobaan untuk beberapa nilai brightness yang berbeda, tampilkan nilai brightness dan hasilnya.
- (3) Lakukan percobaan untuk beberapa nilai contrast yang berbeda, tampilkan nilai contrast dan hasilnya.
- (4) Buatlah program untuk kuantisasi gambar berwarna dengan masing-masing warna menggunakan kuantisasi 4 bit.
- (5) Buatlah program untuk histogram equalization pada gambar berwarna.

Bab 14

Filter Gambar Berwarna

14.1. Dasar Teori

Filter gambar berwarna seperti filter rata-rata untuk reduksi noise, metode prewitt untuk deteksi tepi dan filter sharpness, sebenarnya hampir sama dengan pengolahan gambar derajat keabuan, hanya saja operasionalnya diberlakukan pada setiap nilai R, G dan B.

14.1.1. Menambahkan Noise Gaussian

Menambahkan noise gaussian adalah menambah nilai warna r , g dan b pada titik (x,y) yang terkena noise dengan:

$$r_{noise} = r + e$$

$$g_{noise} = g + e$$

$$b_{noise} = b + e$$

Dimana (r,g,b) adalah warna asal sebelum terkena noise, $(r_{noise}, g_{noise}, b_{noise})$ adalah warna setelah terkena noise, dan e adalah nilai noise $-50 < e < 50$.

14.1.2. Filter rata-rata

Seperti filter rata-rata pada gambar derajat keabuan, filter rata-rata pada gambar berwarna menggunakan nilai rata-rata sebuah titik (x,y) dengan titik-titik tetangganya yang berlaku untuk setiap nilai r , g dan b .

$$r_{filter} = \frac{1}{9} \sum_{i=-1}^1 \sum_{j=-1}^1 r(x+i, y+j)$$

$$g_{filter} = \frac{1}{9} \sum_{i=-1}^1 \sum_{j=-1}^1 g(x+i, y+j)$$

$$b_{filter} = \frac{1}{9} \sum_{i=-1}^1 \sum_{j=-1}^1 b(x+i, y+j)$$

Dimana (r, g, b) adalah warna asal, $(r_{filter}, g_{filter}, b_{filter})$ adalah warna hasil filter.

14.1.3. Metode Prewitt

Seperti metode Prewitt pada gambar derajat keabuan, metode Prewitt pada gambar berwarna menggunakan filter yang sama dengan filter horisontal dan filter vertikal.

Filter horisontal:

$$H1 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Filter vertikal:

$$H1 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

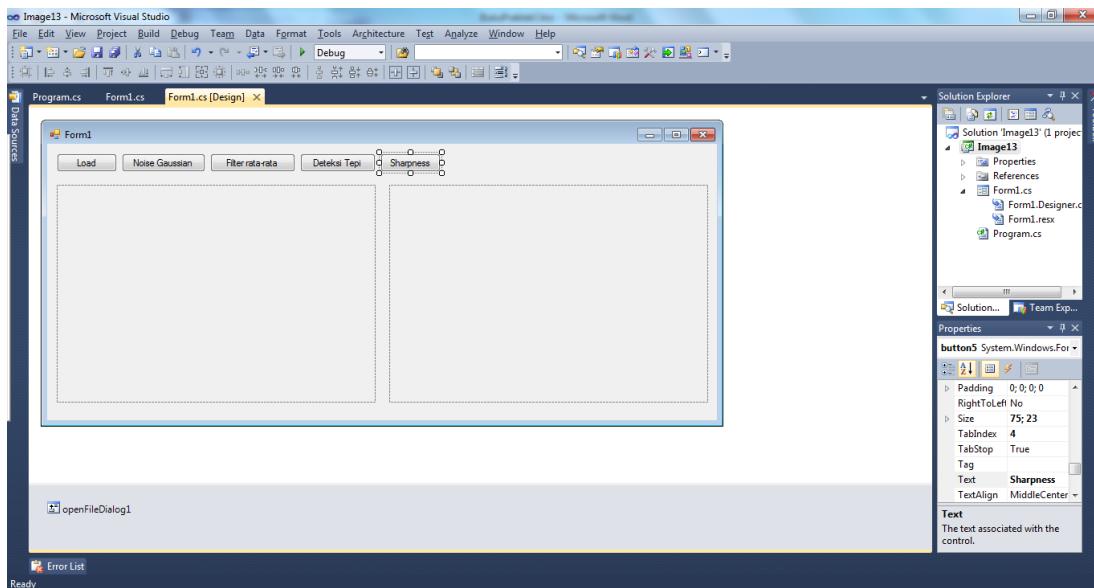
14.1.4. Sharpness

Sharpness pada gambar berwarna bisa dilakukan dengan menggabungkan flter rata-rata dan metode prewitt di atas.

14.2. Petunjuk Praktikum

Langkah-langkah berikut adalah membuat aplikasi filter gambar berwarna yang terdiri dari menambahkan noise gaussian, mengurangi noise dengan filter rata-rata, mendeteksi tepi dengan metode prewitt dan sharpness:

- (1) Buka Visual Studio .Net 2010. Bua project baru dengan File→Project→New Project.
- (2) Pilih Visual C# [Windows Form Application]. Beri nama “Image13” dan Solution name mengikuti dari nama, dan tekan [Ok].
- (3) Setelah keluar form baru dari project yang dibuat, pilih komponen 5 buah button, 2 buah Picturebox dan 1 buah openFileDialog.



Gambar 13.1. Layout project Image12

- (4) Pada Button1, ubah text menjadi “Load”. Pada Button2, ubah text menjadi “Noise Gaussian”. Pada Button3, ubah text menjadi “Filter Rata-rata”. Pada Button4, ubah text menjadi “Deteksi Tepi”. Pada Button5, ubah text menjadi “Sharpness”. Pada Picturebox1, atur size modenya dengan StretchImage. Atur tampilannya seperti gambar 10.1 di atas.
- (5) Tambahkan obyek Bitmap di Class Form1, yaitu objBitmap dan objBitmap1.

```
Bitmap objBitmap;
Bitmap objBitmap1;
```

- (6) Double click pada button1 (Load), tambahkan program berikut:

```
 DialogResult d = openFileDialog1.ShowDialog();
if (d == DialogResult.OK)
{
    objBitmap = new Bitmap(openFileDialog1.FileName);
    pictureBox1.Image = objBitmap;
}
```

(7) Double click pada button2 (Noise Gaussian), tambahkan program berikut:

```

objBitmap1 = new Bitmap(objBitmap);
Random r = new Random();
Color wb;
for (int x = 0; x < objBitmap.Width; x++)
    for (int y = 0; y < objBitmap.Height; y++)
    {
        int p = r.Next(0, 100);
        Color w = objBitmap.GetPixel(x, y);
        wb = w;
        if (p < 20)
        {
            int nr = r.Next(0, 200);
            int rb = w.R + nr - 100;
            if (rb < 0) rb = 0;
            if (rb > 255) rb = 255;
            int gb = w.G + nr - 100;
            if (gb < 0) gb = 0;
            if (gb > 255) gb = 255;
            int bb = w.B + nr - 100;
            if (bb < 0) bb = 0;
            if (bb > 255) bb = 255;
            wb = Color.FromArgb(rb, gb, bb);
        }
        objBitmap1.SetPixel(x, y, wb);
    }
pictureBox2.Image = objBitmap1;

```

(8) Double click pada button3 (Filter rata-rata), tambahkan program berikut:

```

objBitmap = new Bitmap(objBitmap1);
pictureBox1.Image = objBitmap;
for (int x = 1; x < objBitmap.Width-1; x++)
    for (int y = 1; y < objBitmap.Height-1; y++)
    {
        Color w1 = objBitmap.GetPixel(x - 1, y - 1);
        Color w2 = objBitmap.GetPixel(x - 1, y);
        Color w3 = objBitmap.GetPixel(x - 1, y + 1);
        Color w4 = objBitmap.GetPixel(x, y - 1);
        Color w5 = objBitmap.GetPixel(x, y);
        Color w6 = objBitmap.GetPixel(x, y + 1);
        Color w7 = objBitmap.GetPixel(x + 1, y - 1);
        Color w8 = objBitmap.GetPixel(x + 1, y);
        Color w9 = objBitmap.GetPixel(x + 1, y + 1);
        int r = (int)((w1.R + w2.R + w3.R + w4.R + w5.R
+ w6.R + w7.R + w8.R + w9.R) / 9);
        int g = (int)((w1.G + w2.G + w3.G + w4.G + w5.G
+ w6.G + w7.G + w8.G + w9.G) / 9);
        int b = (int)((w1.B + w2.B + w3.B + w4.B + w5.B
+ w6.B + w7.B + w8.B + w9.B) / 9);
        Color wb = Color.FromArgb(r, g, b);
        objBitmap1.SetPixel(x, y, wb);
    }

```

```

        }
        pictureBox2.Image = objBitmap1;
    
```

- (9) Double click pada button4 (Deteksi Tepi), tambahkan program berikut:

```

objBitmap1 = new Bitmap(objBitmap);
for (int x = 1; x < objBitmap.Width - 1; x++)
    for (int y = 1; y < objBitmap.Height - 1; y++)
    {
        Color w1 = objBitmap.GetPixel(x - 1, y - 1);
        Color w2 = objBitmap.GetPixel(x - 1, y);
        Color w3 = objBitmap.GetPixel(x - 1, y + 1);
        Color w4 = objBitmap.GetPixel(x, y - 1);
        Color w5 = objBitmap.GetPixel(x, y);
        Color w6 = objBitmap.GetPixel(x, y + 1);
        Color w7 = objBitmap.GetPixel(x + 1, y - 1);
        Color w8 = objBitmap.GetPixel(x + 1, y);
        Color w9 = objBitmap.GetPixel(x + 1, y + 1);
        int rh = (int)(-w1.R-w4.R-w7.R+w3.R+w6.R+w9.R);
        int gh = (int)(-w1.G-w4.G-w7.G+w3.G+w6.G+w9.G);
        int bh = (int)(-w1.B-w4.B-w7.B+w3.B+w6.B+w9.B);
        int rv = (int)(-w1.R-w2.R-w3.R+w7.R+w8.R+w9.R);
        int gv = (int)(-w1.G-w2.G-w3.G+w7.G+w8.G+w9.G);
        int bv = (int)(-w1.B-w2.B-w3.B+w7.B+w8.B+w9.B);
        int r = (int)(rh + rv);
        if (r < 0) r = -r;
        if (r > 255) r = 255;
        int g = (int)(gh + gv);
        if (g < 0) g = -g;
        if (g > 255) g = 255;
        int b = (int)(bh + bv);
        if (b < 0) b = -b;
        if (b > 255) b = 255;
        Color wb = Color.FromArgb(r, g, b);
        objBitmap1.SetPixel(x, y, wb);
    }
    pictureBox2.Image = objBitmap1;

```

- (10) Double click pada button4 (Sharpness), tambahkan program berikut:

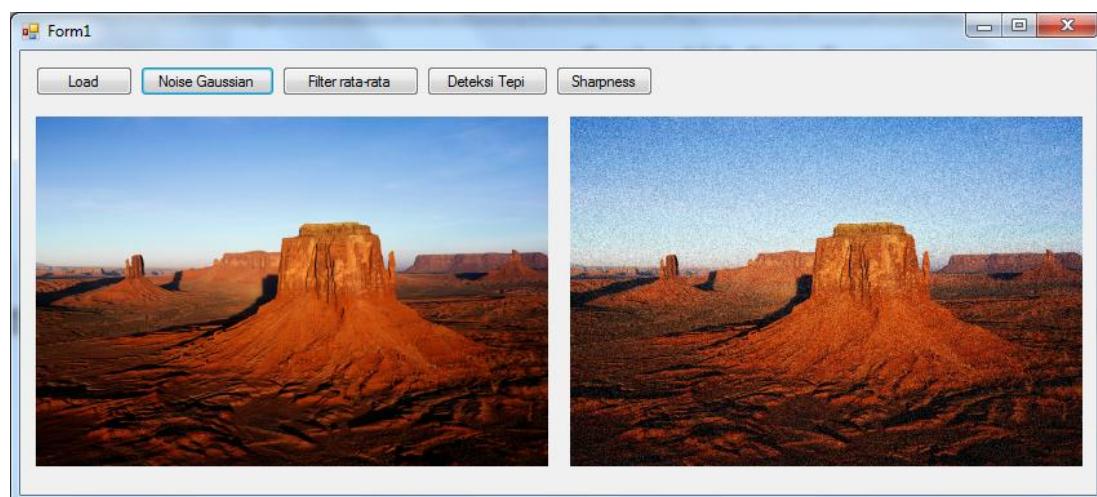
```

objBitmap1 = new Bitmap(objBitmap);
for (int x = 1; x < objBitmap.Width - 1; x++)
    for (int y = 1; y < objBitmap.Height - 1; y++)
    {
        Color w1 = objBitmap.GetPixel(x - 1, y - 1);
        Color w2 = objBitmap.GetPixel(x - 1, y);
        Color w3 = objBitmap.GetPixel(x - 1, y + 1);
        Color w4 = objBitmap.GetPixel(x, y - 1);
        Color w5 = objBitmap.GetPixel(x, y);
        Color w6 = objBitmap.GetPixel(x, y + 1);
        Color w7 = objBitmap.GetPixel(x + 1, y - 1);
        Color w8 = objBitmap.GetPixel(x + 1, y);
        Color w9 = objBitmap.GetPixel(x + 1, y + 1);
    }

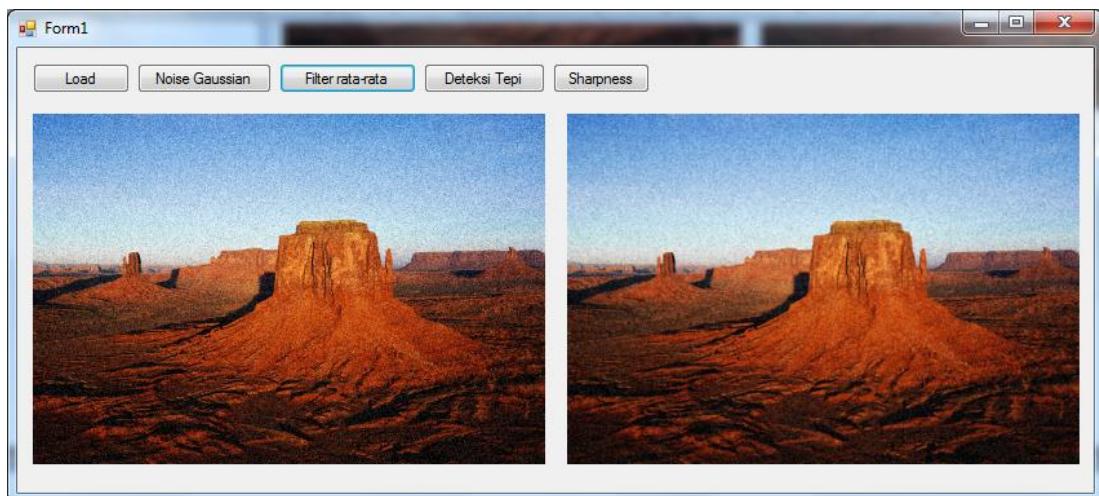
```

```
        int rh = (int)(-w1.R - w4.R - w7.R + w3.R +
w6.R + w9.R);
        int gh = (int)(-w1.G - w4.G - w7.G + w3.G +
w6.G + w9.G);
        int bh = (int)(-w1.B - w4.B - w7.B + w3.B +
w6.B + w9.B);
        int rv = (int)(-w1.R - w2.R - w3.R + w7.R +
w8.R + w9.R);
        int gv = (int)(-w1.G - w2.G - w3.G + w7.G +
w8.G + w9.G);
        int bv = (int)(-w1.B - w2.B - w3.B + w7.B +
w8.B + w9.B);
        int rr = (int)((w1.R + w2.R + w3.R + w4.R +
w5.R + w6.R + w7.R + w8.R + w9.R) / 9);
        int gr = (int)((w1.G + w2.G + w3.G + w4.G +
w5.G + w6.G + w7.G + w8.G + w9.G) / 9);
        int br = (int)((w1.B + w2.B + w3.B + w4.B +
w5.B + w6.B + w7.B + w8.B + w9.B) / 9);
        int r = (int)(rr + rh + rv);
        if (r < 0) r = -r;
        if (r > 255) r = 255;
        int g = (int)(gr + gh + gv);
        if (g < 0) g = -g;
        if (g > 255) g = 255;
        int b = (int)(br + bh + bv);
        if (b < 0) b = -b;
        if (b > 255) b = 255;
        Color wb = Color.FromArgb(r, g, b);
        objBitmap1.SetPixel(x, y, wb);
    }
    pictureBox2.Image = objBitmap1;
```

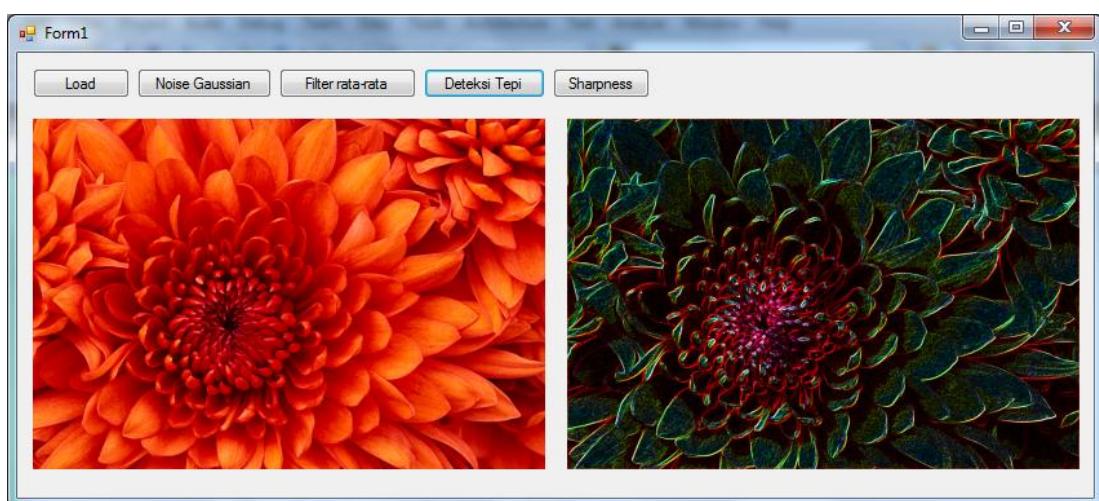
Hasil dari program di atas adalah:



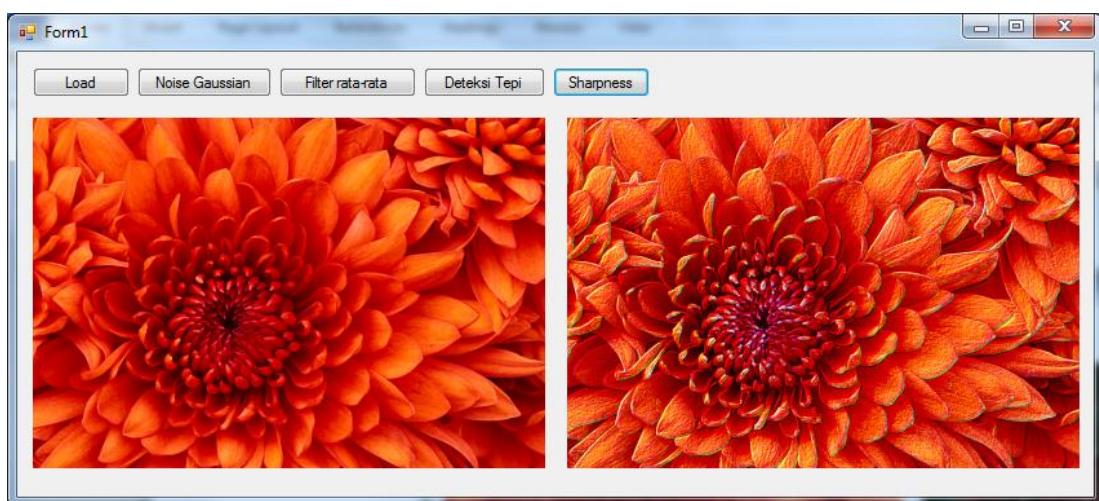
Gambar 14.2. Noise Gaussian



Gambar 14.3. Filter rata-rata



Gambar 14.4. Deteksi tepi



Gambar 14.5. Sharpness

14.3. Laporan Praktikum

Dari program di atas, tulis dan jawab pertanyaan-pertanyaan berikut dalam bentuk laporan praktikum:

- (1) Tuliskan semua kode program dan jelaskan bagian-bagian yang dianggap penting.
- (2) Tambahkan program untuk membuat noise speckle.
- (3) Tambahkan program filter gaussian untuk mereduksi noise.
- (4) Tambahkan program metode sobel untuk deteksi tepi pada gambar berwarna.
- (5) Tambahkan program sharpness dengan menggabungkan filter gaussian dan metode sobel.