MODUL 3 – Operasi Citra Sederhana – Gamma Correction, Grayscale, Grayscale Operation, Image Depth

A. TUJUAN

- Mahasiswa dapat membuat aplikasi Gamma Correction
- Mahasiswa dapat membuat citra level warna Grayscale
- Mahasiswa dapat melakukan operasi Grayscale only
- Mahasiswa dapat membuat Citra dengan image depth yang ditentukan

B. PETUNJUK

1. Awali setiap kegiatan praktikum dengan berdoa
2. Baca dan pahami tujuan, dasar teori, dan latihan-latihan praktikum dengan baik
3. Kerjakan tugas-tugas praktikum dengan baik, sabar dan jujur
4. Tanyakan kepada dosen apabila ada hal-hal yang kurang jelas

C. ALOKASI WAKTU: 6 jam pelajaran

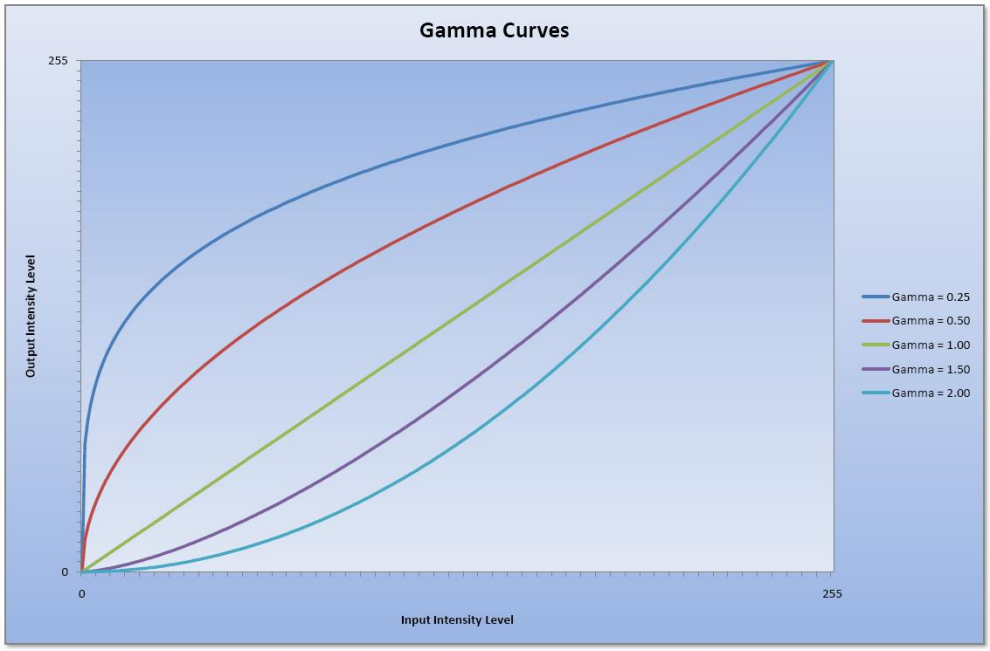
D. DASAR TEORI

- Gamma Correction

  Gamma (Power-Law Transformation), di simbolkan dengan huruf yunani γ, dijelaskan sebagai hubungan antara masukan dan keluaran yang dihasilkan. Masukan yang dimaksud adalah nilai intensitas RGB dari citra. Hubungan antara masukan dan keluaran yang dimaksud adalah keluarannya proporsional dengan masukan yang dipangkatkan dengan nilai Gamma. Rumus dari Gamma Correction adalah sebagai berikut:

  $$I' = 255\ x\ \left(\frac{I}{255}\right)^{\gamma}$$

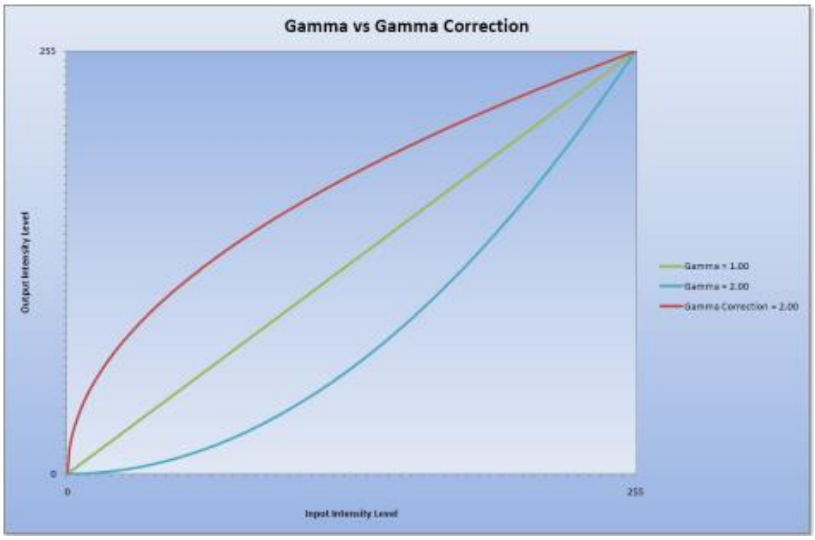
  Berikut adalah ilustrasi hasil keluaran kurva Gamma pada beberapa nilai Gamma.



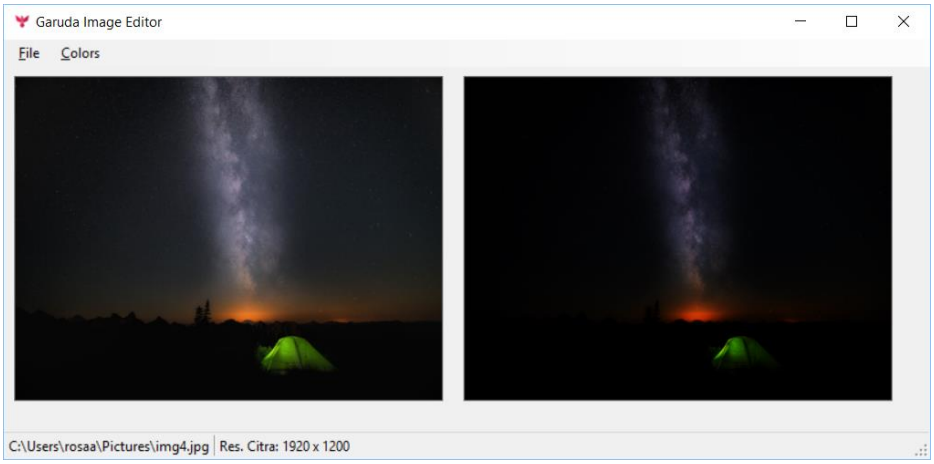
Perhatikan bahwa dengan Gamma bernilai 1, maka nilai masukan sama dengan nilai keluaran dan ditunjukkan dengan garis lurus. Untuk menghitung Gamma Correction, nilai masukan dipangkatkan dengan inverse nilai Gamma. Rumus untuk Gamma Correction adalah sebagai berikut:

$$I' = 255\ x\ \left(\frac{I}{255}\right)^{\frac{1}{\gamma}}$$

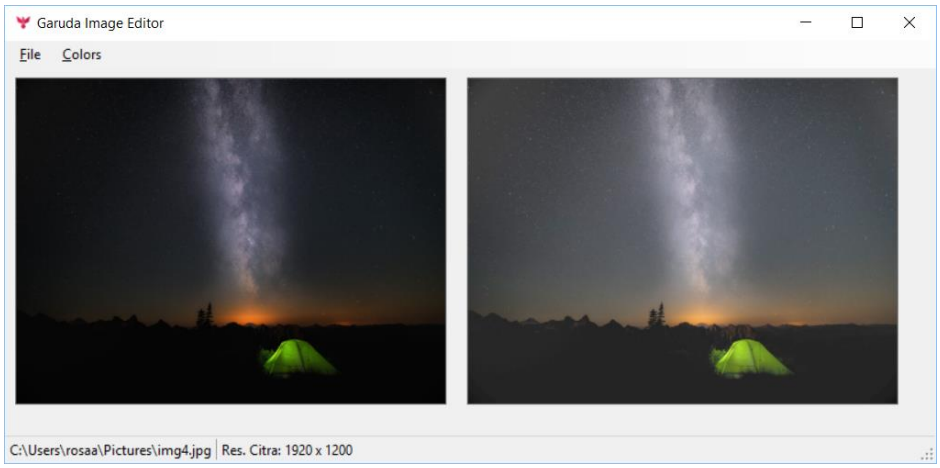
Berikut adalah grafik perbandingan antara kurva gamma dengan kurva gamma correction:



Nilai yang digunakan sebagai Gamma tergantung dari aplikasi yang digunakan. Berikut adalah beberapa contoh gambar RGB sebelum dan setelah diproses menggunakan Gamma Correction.



Gamma Correction dengan nilai Gamma 0.5



Gammar Correction dengan nilai Gamma 6

- Grayscale

  Grayscale ada citra keabuan dengan nilai intensitas kedalaman pixel 8 bit. Grayscale biasanya dipakai untuk merepresentasikan intensitas cahaya pada jalur tunggal dari spectrum electromagnet (inframerah, cahaya visible, ultraviolet, dsb). Beberapa metode grayscaling yang sering dipakai adalah Averaging, Lightness, dan Luminance. Averaging ditentukan dengan merata-rata nilai dari tiap channel R, G, dan B. Lightness ditentukan dengan merata-rata nilai maksimum dan nilai minimum seluruh channel R, G, dan B. Luminance (Luminosity) ditentukan dengan memberikan pembobotan yang disesuaikan dengan spectrum visible yang paling banyak ada pada citra alami.

  $$Grayscale_{avg} = \frac{(R + G + B)}{3}$$

  $$Grayscale_{Lightness} = \frac{\max[R, G, B] + \min[R, G, B]}{2}$$

  $$Grayscale_{Luminance} = 0.21R + 0.72G + 0.07B$$

Original image

Lightness

Average

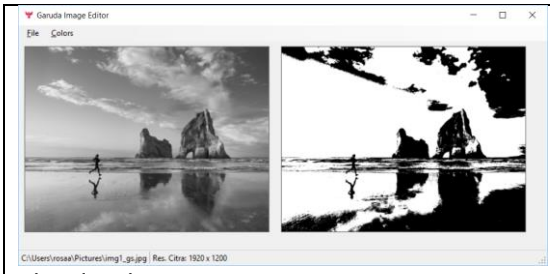
Luminosity

- Grayscale Operation
  Grayscale Operation adalah proses pengolahan warna pada channel grayscale saja. Seperti kita ketahui, pada C# nilai grayscale 8bit dipetakan pada channel RGB dengan membuat nilai sama pada R, G, dan B nya. Hal ini sebenarnya trik saja agar kita tetap bisa membuat Grayscale pada warna RGB. Kelemahan dari trik ini adalah proses komputasi transformasi citra grayscale akan sama lamanya dengan citra RGB biasa. Untuk mengatasinya, kita dapat membuat agar citra hanya sekali saja diproses. Caranya adalah dengan memeriksa apakah nilai R,G,B sama untuk beberapa pixel. Jika sama, dianggap bahwa citranya adalah citra grayscale. Sebagai contoh pada proses gamma correction diatas, kita akan membuat gamma correction pada RGB dan pada Grayscale dengan proses komputasi yang lebih cepat jika dilakukan pada grayscale.

- Bit Depth
  Operasi ini digunakan untuk menunjukkan pada anda tentang kuantisasi citra. Kuantisasi yang dilakukan adalah pada nilai kedalaman warna 1 – 7bit. Transformasi ini sifatnya adalah simulasi saja, bit tidak benar benar berjumlah 1-7bit, hanya variasi warnanya saja sejumlah 1-7bit. Tentukan level kedalaman terlebih dahulu, kemudian nilai warna baru adalah hasil dari warna lama yang dioperasikan dengan level yang telah ditentukan. Berikut adalah rumus yang digunakan:

  $$level = \frac{255}{2^{bit\_depth} - 1}$$

  $$C' = round(\left(\frac{C}{level}\right) * level)$$


1bit depth


2bit depth


3bit depth


4bit depth


5bit depth


6bit depth

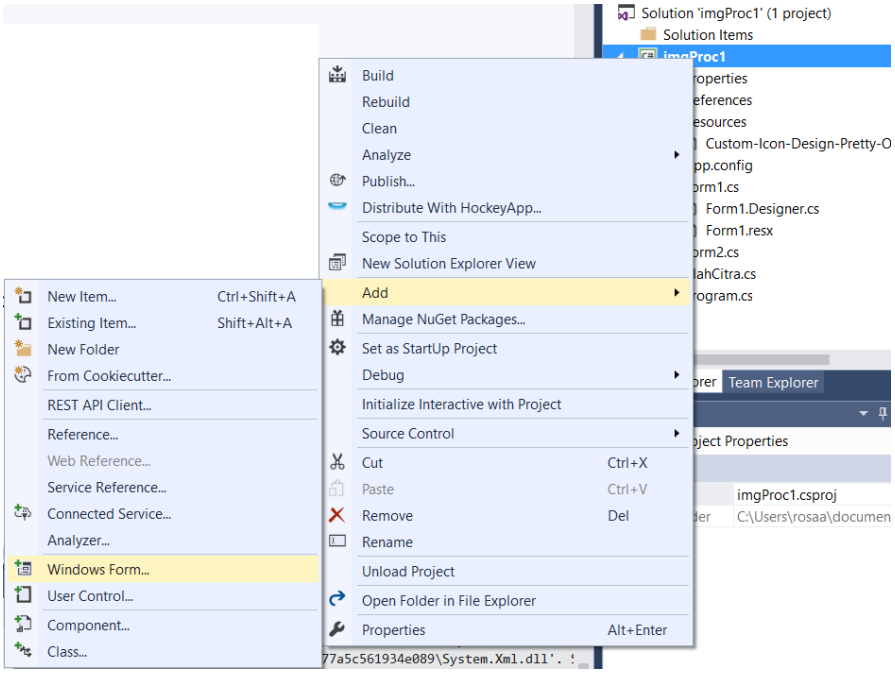

7bit depth

E. LATIHAN PRAKTIKUM

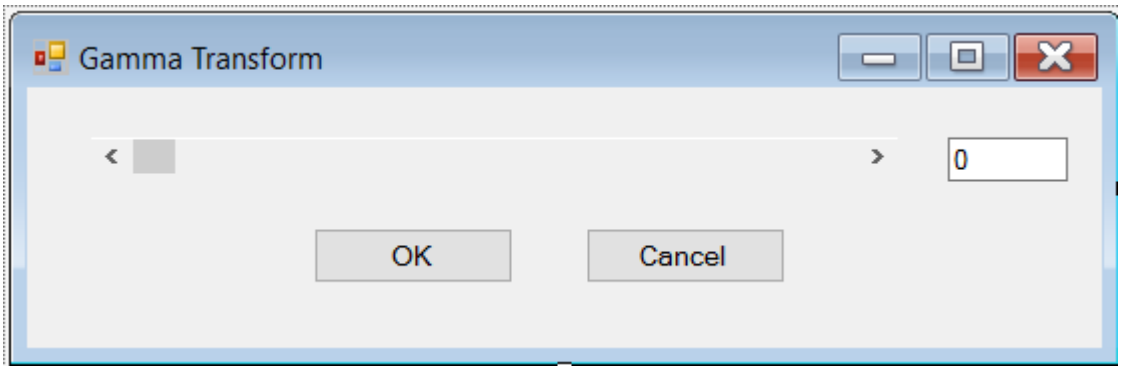
1. Gamma Correction

Percobaan ini akan meminta anda membuat Gamma Correction. Pada percobaan ini, nilai Gamma akan diset menggunakan Horizontal Scroll Bar atau Textbox dari form baru.

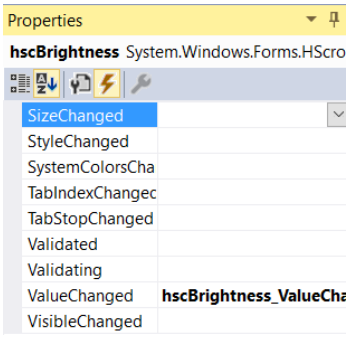
1. Siapkan Form 4 yang digunakan sebagai kendali dari Gamma Transform untuk form pertama (main form). Dari Solution Exlorer, pilih project, klik kanan Add Windows Form.



2. Buat Form control seperti pada contoh berikut.

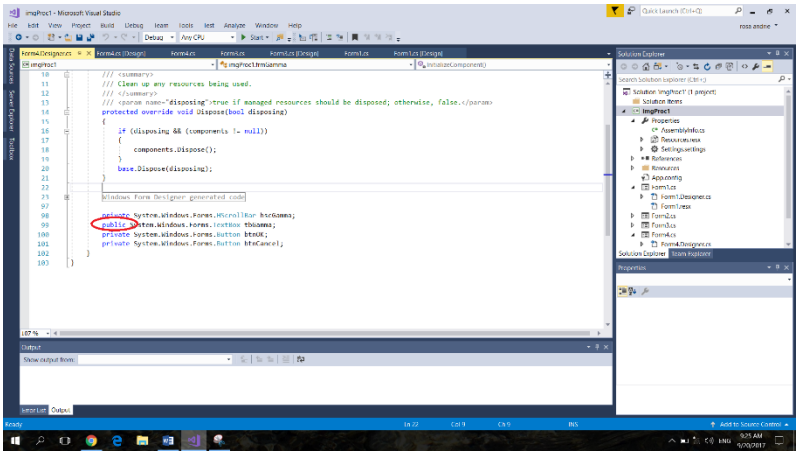


- Set Form Text menjadi “Gamma Transform”. Beri nama Form2 dengan nama frmGamma.
- Tambahkan satu Horizontal Scroll Bar diset name sebagai “hscGamma. Set Value sebagai “0”. Set nilai Maksimumnya dengan nilai 800, dan nilai minimumnya dengan nilai 0. Hal ini dilakukan agar nilai gamma berada pada 0 – 8. Set event ValueChanged dari hscGamma. Event ini digunakan untuk mengubah nilai textbox jika terjadi perubahan nilai pada ScrollBar. Pilih ScrollBar, klik tombol Event (gambar petir) pada properties, Double click pada ValueChanged, dan tambahkan kode berikut. Karena nilai ScrollBar adalah numeric, maka harus dikonversi menjadi String menggunakan method *ToString();* .




```
private void hscGamma_ValueChanged(object sender, EventArgs e)
        {
            tbGamma.Text = Convert.ToString(hscGamma.Value*0.01);
        }
```

- Tambahkan satu TextBox dibagian kanan dari hscGamma, diset name sebagai "tbGamma". Set textnya menjadi "0". Nilai angka pada TextBox sesuai dengan nilai angka dari Horizontal Scroll Bar. Jika nilai Scroll Bar berubah, maka nilai textboxnya juga berubah. Jika nilai textboxnya diubah, maka nilai scroll bar juga ikut berubah. Jika nilai textbox "null" atau kosong atau "-", maka nilai scrollbar adalah "0". Jika nilai TextBox lebih kecil dari 0, maka akan diset menjadi 0. Jika nilai TextBox lebih besar dari 8, maka akan diset menjadi 8. Set "tbGamma" sebagai public Control, agar dapat diakses oleh form utama. Perhatikan Screenshot untuk mengubah Control private menjadi public. Saat textbox berubah, maka nilai ScrollBar juga berubah. Tambahkan kode berikut pada event "TextChanged":



Nilai ScrollBar sesuai dengan nilai TextBox jika berada pada range 0 hingga 8. Karena TextBox bertipe Text dan ScrollBar bertipe Numeric, maka nilai TextBox harus dikonversi ke Numeric (int16). Jika diinputkan selain angka, atau diluar range yang ditentukan, maka akan muncul error MessageBox. Lakukan langkah yang sama pada tbContrast.

```
private void tbGamma_TextChanged(object sender, EventArgs e)
        {
            if ((tbGamma.Text == "") || (tbGamma.Text == "-"))
            {
                hscGamma.Value = 0;
                tbGamma.Text = "0";
            }
            else if ((Convert.ToInt16(tbGamma.Text) <= 8) &&
(Convert.ToInt16(tbGamma.Text) >= 0))
            {
                hscGamma.Value = Convert.ToInt16(tbGamma.Text);
            }
            else
            {
                MessageBox.Show("Input nilai Error");
                tbGamma.Text = "0";
            }
        }

        private void hscGamma_ValueChanged(object sender, EventArgs e)
        {
            tbGamma.Text = Convert.ToString(hscGamma.Value*0.01);
        }
    }
```

- Tambahkan Satu Button set text menjadi "OK". Set name sebagai "tbOK". Set Property DialogResult dari Button menjadi "OK". Jika tombol OK ditekan, maka frmBrightness akan ditutup. Double click pada Tombol OK, ketikkan kode berikut. Kode ini digunakan untuk menutup form jika nilai Brightness sudah diset.

```
private void tbOK_Click(object sender, EventArgs e)
{
   this.Close();
}
```

3. Double-click frmGamma pada File Form4.cs [Design]. kode berikut digunakan untuk memberi nilai awal tbGamma menjadi "0":

```
private void frmGamma_Load(object sender, EventArgs e)
        {
            tbGamma.Text = hscGamma.Value.ToString();
        }
```

4. Pada Form Utama, tambahkan kode berikut saat menuStrip "Gamma Transform" di klik. Bagian awal dari kode berfungsi untuk memeriksa apakah citra yang akan diolah sudah dibuka atau belum, jika belum maka akan muncul MessageBox yang memberi pesan "Tidak ada citra yang akan diolah". FrmGamma di buat sebagai object baru "frm4" seperti terlihat pada baris 425. Jika tombol OK dari frm4 ditekan (baris 426), maka proses Gamma Transform akan dijalankan.

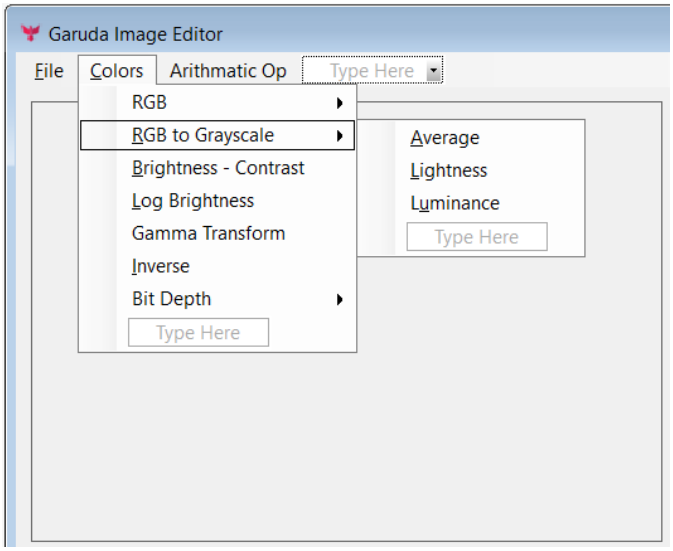
```
417         private void gammaTransformToolStripMenuItem_Click(object sender, EventArgs e)
418         {
419             if (pbInput.Image == null)
420             {
421                 MessageBox.Show("Tidak Ada citra yang akan diolah");
422             }
423             else
424             {
425                 frmGamma frm4 = new frmGamma();
426                 if (frm4.ShowDialog() == DialogResult.OK)
427                 {
428                     Bitmap b = new Bitmap((Bitmap)this.pbInput.Image);
429                     double nilaiGamma = Convert.ToDouble(frm4.tbGamma.Text);
430                     progressBar1.Visible = true;
431                     int r1, g1, b1;
432                     
433                     double merah, hijau, biru;
434                     for (int i = 0; i < 10; i++)
435                     {
436                         for (int j = 0; j < 10; j++)
437                         {
```

```
for (int i = 0; i < b.Width; i++)
{
   for (int j = 0; j < b.Height; j++)
   {
        merah = b.GetPixel(i,j).R;
        hijau = b.GetPixel(i,j).G;
        biru = b.GetPixel(i,j).B;
        r1 = ..........;
        g1 = ..........;
        b1 = ..........;
        b.SetPixel(i, j, Color.FromArgb(r1, g1, b1));
   }
}
```

Gambar diatas adalah citra yang diproses menggunakan Gamma Correction nilai gamma 3.

2. Membuat Grayscale

Grayscale yang dibuat pada percobaan kali ini adalah grayscale Average, Lightness, dan Luminance. Untuk tiap tipe grayscale, anda dapat buat menustrip yang sesuai. Perhatikan gambar berikut:



Karena tidak memerlukan form baru untuk setting parameter, maka Grayscale dapat langsung dilakukan pada form utama. Karena kita bekerja pada tiga channel (RGB) dan tidak bisa digunakan satu channel saja (dalam kasus ini Grayscale), maka untuk membuat grayscale dengan mudah, kita hanya perlu membuat nilai R,G, dan Bnya sama. Perhatikan kode baris 15 yang digunakan untuk membuat citra grayscale. Pada variable grayAvg, anda tinggal masukkan rumus grayscale average yang sudah dituliskan dibagian atas dari modul ini.

Untuk Grayscale Average:

```
1   if (pbInput.Image == null)
2   {
3      MessageBox.Show("Tidak Ada citra yang akan diolah");
4   }
5   else
6   {
7      Bitmap b = new Bitmap((Bitmap)this.pbInput.Image);
8      progressBar1.Visible = true;
9      for (int i = 0; i < b.Width; i++)
10     {
11        for (int j = 0; j < b.Height; j++)
12        {
13           Color c1 = b.GetPixel(i, j);
14           int grayAvg = ..........................................;
15           b.SetPixel(i, j, Color.FromArgb(grayAvg, grayAvg, grayAvg));
16        }
17        progressBar1.Value = Convert.ToInt16(100 * (i + 1) / b.Width);
18     }
19     progressBar1.Visible = false;
20     this.pbOutput.Image = b;
21  }
```

Untuk Grayscale Lightness:

```
if (pbInput.Image == null)
{
   MessageBox.Show("Tidak Ada citra yang akan diolah");
}
else
{
   Bitmap b = new Bitmap((Bitmap)this.pbInput.Image);
   progressBar1.Visible = true;
   for (int i = 0; i < b.Width; i++)
   {
      for (int j = 0; j < b.Height; j++)
      {
         Color c1 = b.GetPixel(i, j);
         int nilaiMax = ……;
         int nilaiMin = ……;
         int gLight = ……;
         b.SetPixel(i, j, Color.FromArgb(gLight, gLight, gLight));
      }
      progressBar1.Value = Convert.ToInt16(100 * (i + 1) / b.Width);
   }
   progressBar1.Visible = false;
   this.pbOutput.Image = b;
}
```

Untuk Grayscale Luminance:

```
Bitmap b = new Bitmap((Bitmap)this.pbInput.Image);
progressBar1.Visible = true;
for (int i = 0; i < b.Width; i++)
{
   for (int j = 0; j < b.Height; j++)
   {
      Color c1 = b.GetPixel(i, j);
      int gLum = ……;
      b.SetPixel(i, j, Color.FromArgb(gLum, gLum, gLum));
   }
   progressBar1.Value = Convert.ToInt16(100 * (i + 1) / b.Width);
}
progressBar1.Visible = false;
this.pbOutput.Image = b;
}
```

3. Membuat Grayscale Operation

Percobaan kali ini dilakukan untuk membuat operasi sesuai dengan channel warnanya. Jika citra masukan adalah RGB, maka komputasi dilakukan tiga kali yaitu pada R, G, dan B. Jika masukan adalah grayscale, maka komputasi hanya cukup dilakukan satu kali walapun kita bekerja pada tiga channel. Hal ini dilakukan untuk mempercepat komputasi jika citra masukannya adalah grayscale. Untuk melakukan hal ini, maka aplikasi harus tahu citra masukannya adalah RGB atau grayscale. Caranya adalah dengan melakukan pemeriksaan nilai pixel pada 10x10 resolusi awal dari total resolusi citra. Jika nilai R, G, dan B nya sama, maka kemungkinan besar citranya adalah grayscale. Kita akan lakukan percobaan ini pada Gamma Transform.

Berikut adalah script dari Gamma Transform yang telah dimodifikasi sehingga bisa membedakan citra masukan RGB atau grayscale. Baris 11-19 adalah script yang digunakan untuk memeriksa citra masukan dalam bentuk Grayscale atau RGB. Baris 27-41 adalah operasi Gamma Transform yang telah dimodifikasi agar dapat melakukan komputasi sesuai dengan citra masukannya.

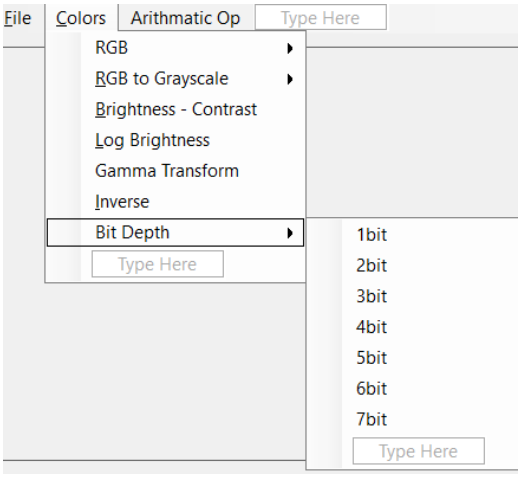
```
1   Bitmap b = new Bitmap((Bitmap)this.pbInput.Image);
2   double nilaiGamma = Convert.ToDouble(frm4.tbGamma.Text);
3   progressBar1.Visible = true;
4   int r1, g1, b1;
5   bool gray=false;
6   double merah, hijau, biru;
7   for (int i = 0; i < 10; i++)
8   {
9      for (int j = 0; j < 10; j++)
10     {
11        if …….(cek nilai pixel r,g,
12  dan b apakah sama)
13        {
14           …….;
15        }
16        else
17        {
18           …….;
19           …….;
20        }
21     }
22  }
23  for (int i = 0; i < b.Width; i++)
24  {
25     for (int j = 0; j < b.Height; j++)
26     {
27         merah = b.GetPixel(i,j).R;
28         hijau = b.GetPixel(i,j).G;
29         biru = b.GetPixel(i,j).B;
30         if (gray == false)
31         {
32            r1 =
33            g1 =
34            b1 =
35            b.SetPixel(i, j, Color.FromArgb(r1, g1, b1));
36         }
37         else
38         {
39            r1 =
40            b.SetPixel(i, j, Color.FromArgb(r1, r1, r1));
41         }
42     }
43     progressBar1.Value = Convert.ToInt16(100 * (i + 1) / b.Width);
44  }
```

4. Image Depth

Percobaan ini digunakan sebagai simulasi dari proses kuantisasi citra. Pada kuantisasi citra, pixel dapat direpresentasikan dengan n-bit kedalaman (default menggunakan 8-bit). Pada pixel 8-bit, warna yang memungkinkan adalah 256 warna, dari 0 (0000 0000) hingga 255(1111 1111). Pada pixel 7-bit, warna yang memungkinkan adalah 128 warna, dari 0 (000 0000) hingga 127 (111 1111). Kemungkinan warna didapat dari pangkat 2 jumlah bit. Jika 7bit, maka jumlah warnanya adalah 2^7 = 128, dst. Karena Visual Studio 2017 bekerja hanya

pada 8 bit, maka percobaan ini hanya memanipulasi warna sehingga jumlah warnanya sesuai dengan kedalamannya. Untuk kasus 7-bit, maka dua warna 8-bit diwakili oleh satu warna 7-bit. Contoh pixel warna 0 dan 1 pada 8-bit, diwakili oleh warna 0 pada 7-bit. pixel warna 2 dan 3 pada 8-bit, diwakili oleh warna 1 pada 7-bit, dst.

Gambar berikut adalah menu strip yang dibuat:



Berikut adalah script manipulasi 7 bit-depth image.

```
Bitmap b = new Bitmap((Bitmap)this.pbInput.Image);
double level = ……;
progressBar1.Visible = true;
for (int i = 0; i < b.Width; i++)
{
  for (int j = 0; j < b.Height; j++)
  {
     Color c1 = b.GetPixel(i, j);
     int R = ……;
     int G = ……;
     int B = ……;
     b.SetPixel(i, j, Color.FromArgb(R, G, B));
  }
  progressBar1.Value = Convert.ToInt16(100 * (i + 1) / b.Width);
}
```

A. TUGAS PRAKTIKUM

1. Kerjakan tahapan-tahapan percobaan yang dituliskan pada modul ini.
2. Buat simulasi 2 hingga 7 bit-depth.

--- SELAMAT BELAJAR ---