

MODUL 4 – Arithmetic Operation, Warna Terdekat, Error Diffusion, Histogram

A. TUJUAN

- Mahasiswa dapat membuat aplikasi denoising dengan menggunakan operasi aritmatika sederhana
- Mahasiswa dapat melakukan pencarian warna terdekat ke palet warna yang telah ditentukan
- Mahasiswa dapat membuat aplikasi error diffusion
- Mahasiswa dapat menampilkan histogram dari citra

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

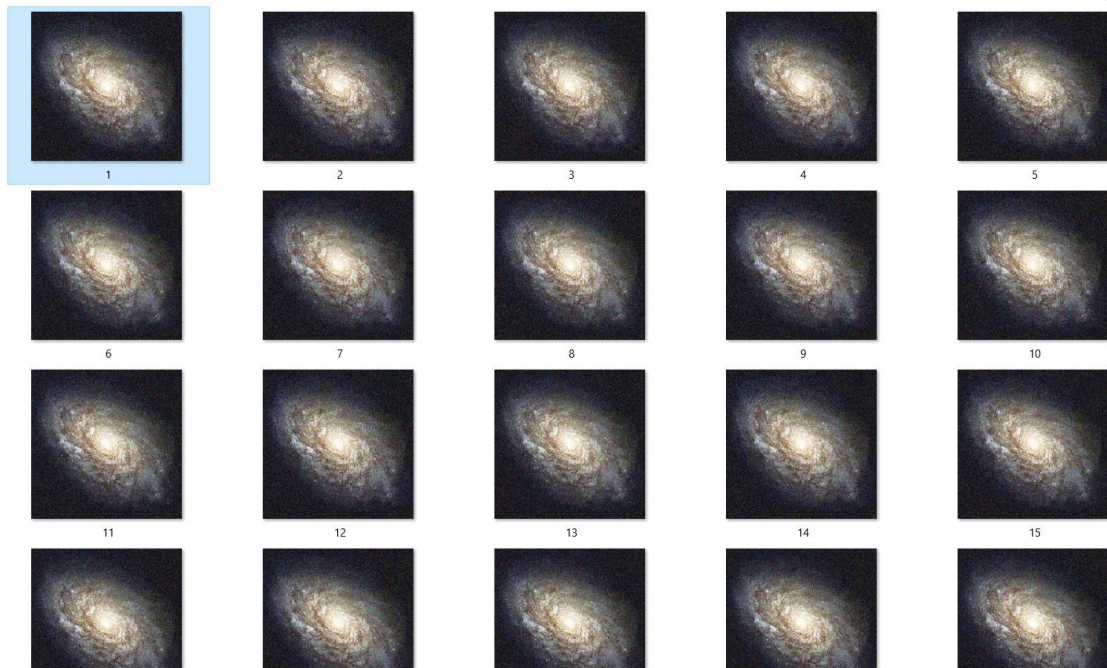
C. ALOKASI WAKTU: 6 jam pelajaran

D. DASAR TEORI

- Arithmetic Operation

Arithmetic Operation atau operasi matematika biasa dapat dikenakan pada citra untuk berbagai keperluan yang bermanfaat. Operasi ini adalah operasi penjumlahan, pengurangan, perkalian, pembagian, atau gabungan operasi aritmetika pada citra. Contoh yang paling umum adalah operasi Averaging untuk denoising image. Operasi ini melakukan perhitungan nilai rata-rata tiap pixel yang berkoordinat sama dalam sebuah kumpulan citra. Operasi ini biasanya dilakukan untuk menghilangkan noise dari faktor eksternal yang terjadi saat akuisisi citra.

Gambar berikut sengaja diberikan Gaussian noise secara acak dibuat sebanyak 100 kali:

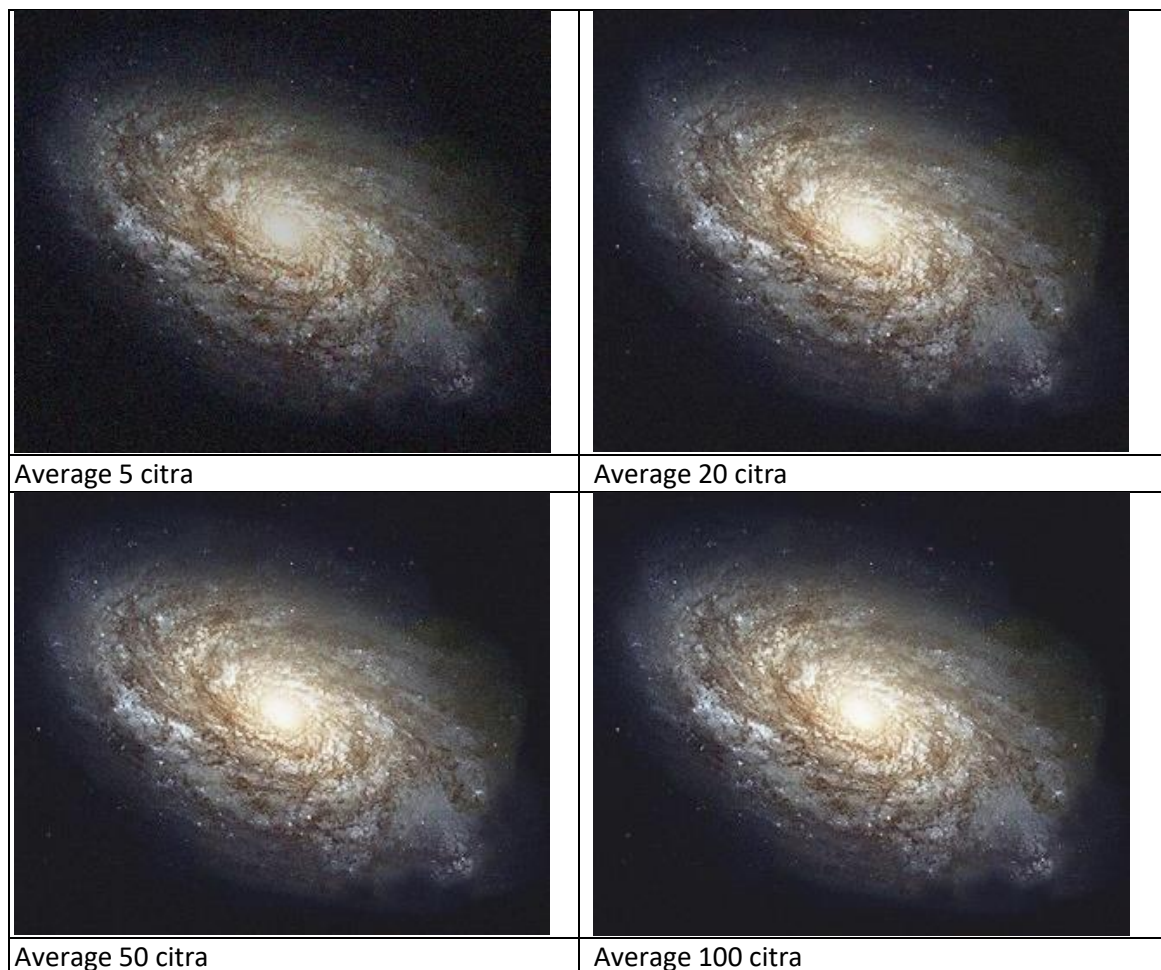


Kumpulan Gambar yang diberikan Gaussian Noise secara acak



Salah satu gambar yang diperbesar dan telah diberikan Gaussian Noise

Berikut adalah hasil image yang telah didenoising menggunakan averaging pada 5 citra, 20 citra, 50 citra, dan 100 citra.



- Mencari Warna Terdekat

Adakalanya jumlah warna yang terlalu banyak perlu dikurangi menjadi sedikit warna karena beberapa alasan, diantaranya:

1. Kemungkinan keterbatasan perangkat tampilan (display device) yang tidak terlalu baik.
2. Kemungkinan ingin membatasi jumlah memory dari proses komputasi citra.
3. Kemungkinan ingin melakukan kompresi citra dengan ratio kompresi tinggi. Hal ini bisa dilakukan karena dengan jumlah warna sedikit, kemungkinan redundansi nilai warna semakin tinggi.
4. Kemungkinan keterbatasan jumlah tinta warna saat citra akan dicetak.

Untuk mencari warna terdekat dapat dilakukan dengan menghitung jarak antara warna pixel asli dengan warna palet yang ingin digunakan sebagai pengganti. Pengukuran jarak dilakukan pada semua palet warna yang ditentukan. Pixel asli akan diganti dengan palet warna yang memiliki kedekatan paling minimal (paling mirip) dengan warna pixel asli.

Langkah awal yang dilakukan adalah dengan menset nilai minimum awal menjadi nilai tertinggi dari jarak kedekatan yang mungkin, dalam hal ini:

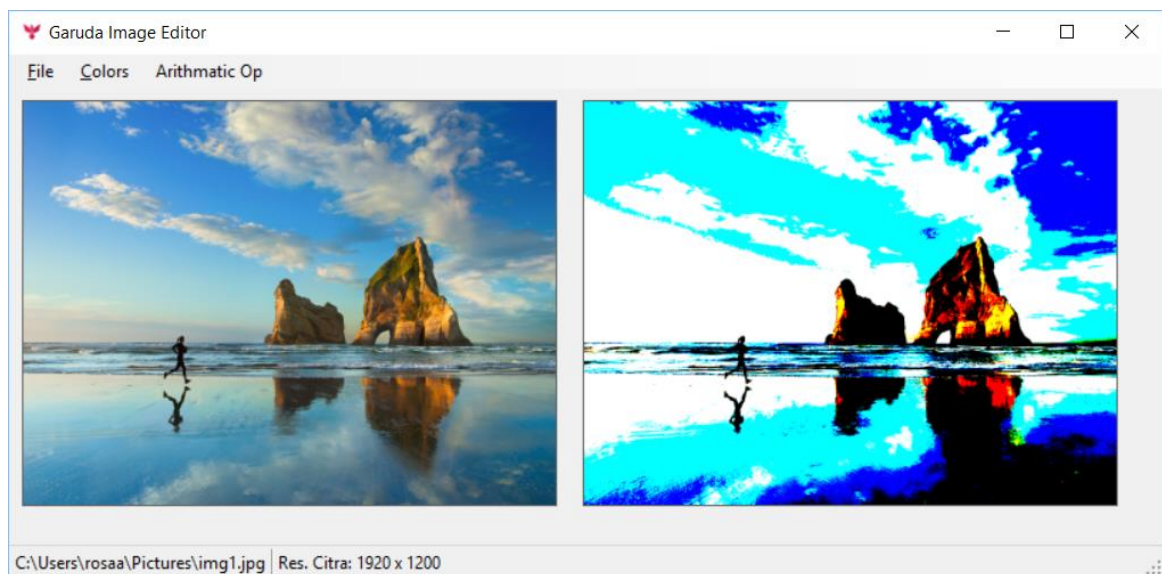
$$\text{minDist} = 255 \times 255 + 255 \times 255 + 255 \times 255$$

Untuk lebih jelasnya, perhatikan algoritma berikut:

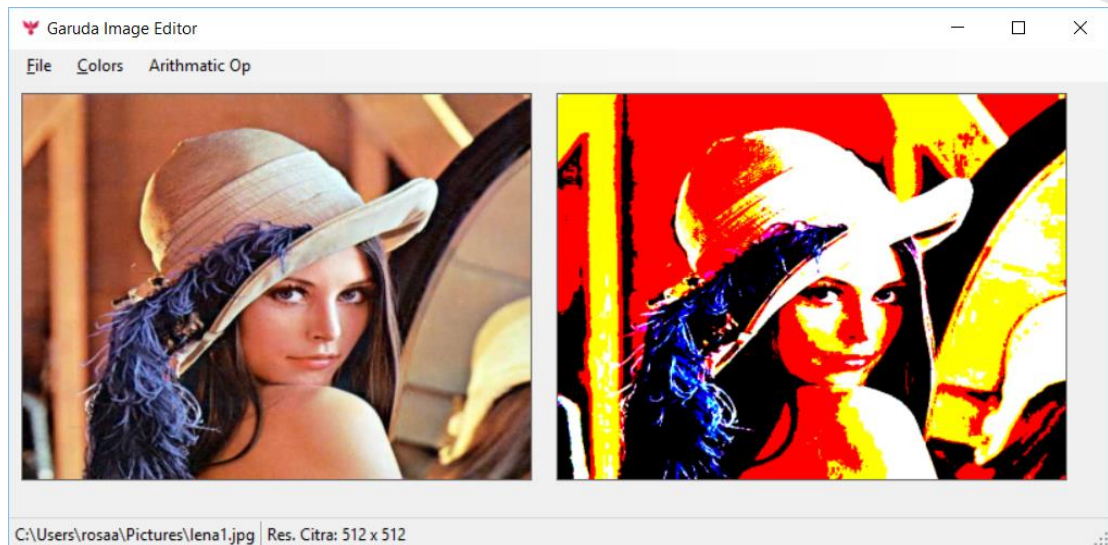
```

1  minDist = 255*255 + 255*255 + 255*255
2  For warnaPalet = 0 to warnaPalet.terakhir
3      mDiff = merah(warnaAsli) – merah(warnaPalet)
4      hDiff = hijau(warnaAsli) – hijau(warnaPalet)
5      bDiff = biru(warnaAsli) – biru(warnaPalet)
6      Jarak = mDiff*mDiff + hDiff*hDiff + bDiff*bDiff
7      If Jarak < minDist
8          minDist = Jarak
9          warnaTerdekat = warnaPalet
10     End If
11 Next
    
```

Gambar berikut menunjukkan proses penyederhanaan citra 16juta warna kedalam 8 warna (hitam, merah, hijau, kuning, biru, magenta, cyan, dan putih)



Berikut adalah contoh yang lain:



- Image Dithering (Error Diffusion)

Error Diffusion adalah salah satu tipe halftoning dimana proses kuantisasi didistribusi pada piksel tetangga yang belum diproses. Penggunaan utamanya adalah untuk mengubah kedalaman image menjadi lebih kecil, tetapi kualitas gambar tidak terlalu buruk.

Proses ini biasanya digunakan pada teknologi cetak, karena jumlah warna tinta yang tidak terlalu banyak sehingga ilusi warna terbentuk dari teknik ini.

Error diffusion digolongkan sebagai area operation, karena yang dilakukan oleh error diffusion pada satu pixel akan mempengaruhi hasil perhitungan operasi pixel lainnya. Error diffusion juga memiliki kemampuan untuk memperkuat tepi dari citra. Hal ini dapat membuat teks dalam citra menjadi lebih mudah terbaca daripada teknik halftoning yang lain.

Beberapa metode Error Diffusion diantaranya adalah:

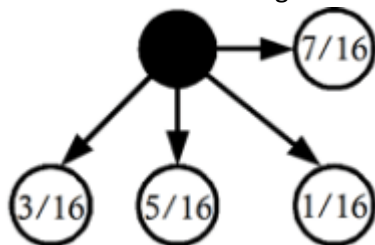
1. Floyd and Steinberg

Floyd dan Steinberg mendeskripsikan system pembentuk error diffusion pada citra digital dengan menggunakan kernel sederhana:

$$\frac{1}{16} \begin{bmatrix} - & \# & 7 \\ 3 & 5 & 1 \end{bmatrix}$$

Dimana – menunjukkan pixel yang sudah diproses, # menunjukkan pixel saat ini yang sedang diproses.

Atau dalam bentuk diagram node ditunjukkan dengan gambar berikut:

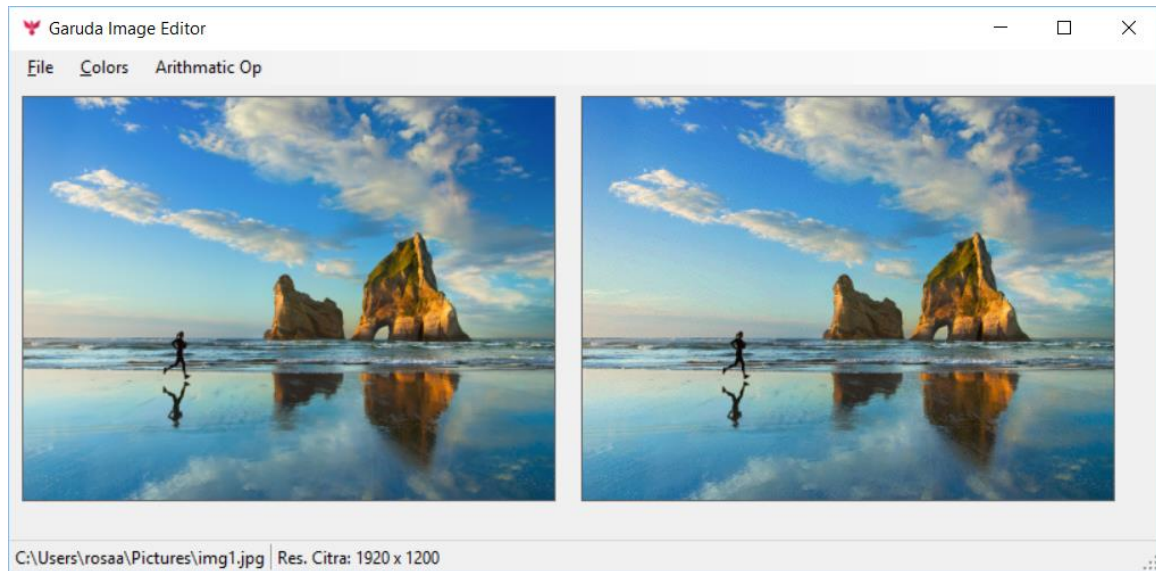


2. Jarvis, Judice dan Ninke dari Bell Labs mendeskripsikan system yang sama yang mereka istilahkan “minimized average error”, menggunakan kernel dengan ukuran yang lebih besar:

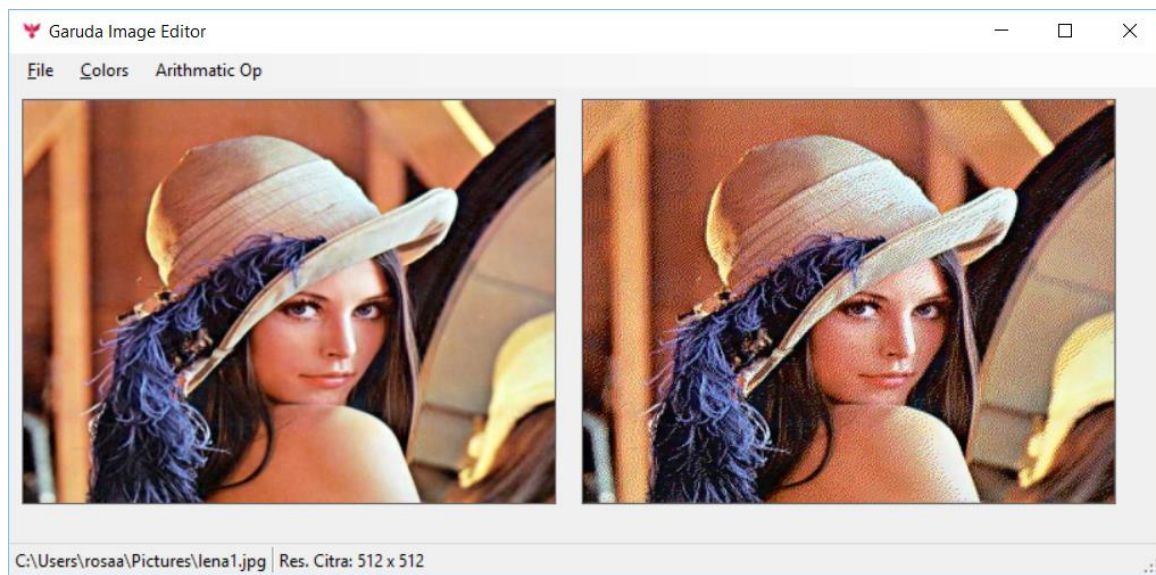
$$\frac{1}{48} \begin{bmatrix} - & - & \# & 7 & 5 \\ 3 & 5 & 7 & 5 & 3 \\ 1 & 3 & 5 & 3 & 1 \end{bmatrix}$$

Jika ingin menyederhanakan jumlah warna tanpa error diffusion, maka cara yang termudah adalah dengan mencari warna terdekatnya. Jika input warna lebih dekat ke merah, maka warnanya akan diubah kemerah. Jika lebih dekat ke cyan, maka warnanya diubah ke cyan, dan seterusnya. Hasilnya terlihat pada gambar pada sub bab warna terdekat.

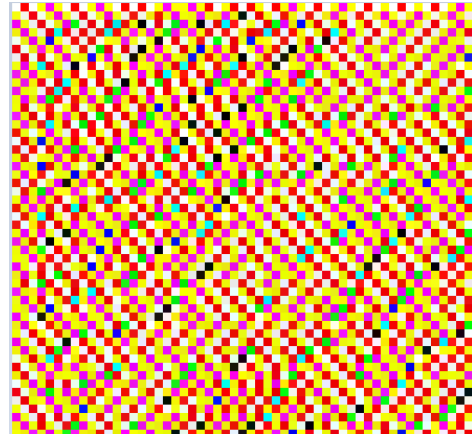
Perhatikan gambar berikut sebelum dan sesudah disederhanakan kedalam 8 warna menggunakan Floyd & Steinberg Error Diffusion.



Berikut adalah contoh yang lain:

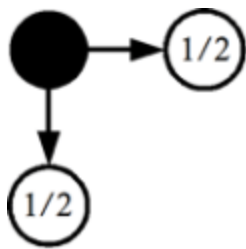


Untuk lebih jelasnya, perhatikan gambar bagian kiri atas yang telah diperbesar:



Error diffusion bekerja dengan membandingkan warna asli pixel dengan warna terdekat yang ditentukan dan menghitung selisihnya. Selisih ini dinamakan dengan error. Porsi error ini kemudian dibagikan ke pixel tetangga sehingga menyebabkan errornya terdifusi dan dinamakan “Error Diffusion”.

Bentuk paling sederhana dari error diffusion terlihat dari gambar berikut:



Dengan bentuk error diffusion seperti diatas, setengah error dari pixel yang diproses (ditunjukkan dengan titik hitam) terdifusi setengahnya ke pixel sebelah kanan dan setengahnya lagi ke pixel bawahnya. Pada error diffusion warna, proses ini harus dilakukan disemua channel red, green, dan blue.

Bagian penting yang perlu digarisbawahi adalah jumlah total error yang terdifusi tidak boleh melebihi satu. Hal penting lain yang perlu dicatat adalah ketika porsi error terdifusi ke tetangganya, tetangga tersebut memiliki nilai pixel antara 0 – 255. Jika nilainya diluar 0 – 255, maka nilai akhir perlu di truncate.

Berikut adalah pseudo-code untuk error difusi sederhana yang harus dilakukan setelah sebuah pixel(x,y) diubah nilainya ke warna terdekat:

1	Error = warna asli – warnaTerdekat
2	SetPixel(x+1, y) = Truncate(GetPixel(x+1,y) + 0.5 * error)
3	SetPixel(x, y+1) = Truncate(GetPixel(x, y+1) + 0.5 * error)

Berikut adalah pseudo-code untuk error difusi Floyd and Steinberg yang harus dilakukan setelah sebuah pixel(x,y) diubah nilainya ke warna terdekat:

1	Error = warna asli – warnaTerdekat
---	------------------------------------

2	$\text{SetPixel}(x+1, y) = \text{Truncate}(\text{GetPixel}(x+1, y) + 7/16 * \text{error})$
3	$\text{SetPixel}(x-1, y+1) = \text{Truncate}(\text{GetPixel}(x-1, y+1) + 3/16 * \text{error})$
4	$\text{SetPixel}(x, y+1) = \text{Truncate}(\text{GetPixel}(x, y+1) + 5/16 * \text{error})$
5	$\text{SetPixel}(x+1, y+1) = \text{Truncate}(\text{GetPixel}(x+1, y+1) + 1/16 * \text{error})$

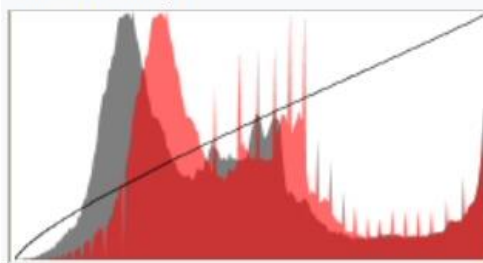
- Histogram Image

Citra Histogram adalah tipe histogram yang berfungsi sebagai representasi grafis dari distribusi intensitas warna pixel pada citra digital. Grafik histogram akan menampilkan jumlah pixel dari tiap intensitas warna pixel. Dengan memperhatikan grafik histogram, seseorang dapat menilai distribusi warna pixel secara cepat.

Axis horizontal dari grafik histogram mewakili variasi nilai pixel, sedangkan axis vertikalnya mewakili jumlah pixel. Sebelah kiri dari axis horizontal mewakili area hitam dan gelap, bagian tengah mewakili area abu-abu, dan bagian kanan mewakili area terang dan putih.



Sunflower image



Histogram of sunflower image

Berikut adalah algoritma dalam membuat histogram dari citra digital

1	Siapkan dua form, form 1 berisi 1 chart yang digunakan untuk menampilkan histogram citra grayscale, form 2 berisi 3 chart yang digunakan untuk menampilkan histogram citra R, G, dan B.
2	Buat variable penyimpan histogram R, G, dan B Variabel dapat disimpan dalam array 2 dimensi atau menggunakan Dictionary pada Visual Studio. Tipe data untuk axis horizontal dapat menggunakan int atau Byte, sedangkan pada axis vertical, disarankan menggunakan tipe data Double. Hal ini dilakukan karena nilai histogramnya akan kita normalisasi. Pada percobaan ini data histogram kita normalisasi karena selanjutnya akan dilakukan proses perbaikan citra menggunakan histogram. Percobaan perbaikan citra menggunakan histogram akan kita lakukan pada jobsheet berikutnya.
3	Buat variable bertipe Bitmap untuk menyimpan nilai citra masukan
4	Beri nilai awal untuk tiap intensitas pixel dengan nilai 0

5	Untuk setiap baris dan kolom citra, periksa nilai pixel dan tambahkan 1 untuk tiap nilai pixel yang diperiksa
6	Periksa nilai dictionary, jika histogram dari R sama persis dengan G //Tampilkan form 1
7	Simpan nilai histogram dari dictionary ke dalam variable <i>kunci</i> bertipe data List
8	Tampilkan form 1
9	Untuk setiap nilai kunci
10	Normalisasi nilai histogram
11	Tambahkan data histogram ke dalam chart
12	Selesai
13	Jika tidak sama antara R dan G //Tampilkan form 2
14	Lakukan langkah yang sama seperti ketika menampilkan form 1
15	selesai

E. LATIHAN PRAKTIKUM

1. Arithmetic Operation

Arithmetic Operation yang dilakukan pada percobaan kali ini adalah denoising dengan menggunakan banyak citra noise. Disediakan 100 citra yang telah diberikan Gaussian noise, citra tersebut berada pada folder yang sama. Toolstrip Average Denoising akan menunjuk dan membuka folder dimana 100 citra tersebut berada, kemudian diproses menggunakan average denoising.

Perhatikan dan isikan script yang kurang dari average denoising berikut. Script pada baris kedua hingga lima belas digunakan untuk membaca semua file image berekstensi .jpg. pada satu folder kemudian disimpan pada List bertipe image bernama pictureArray. Baris 16 digunakan untuk menampilkan image pertama dari List pictureArray ke dalam pictureBox input image. Variable jumGambar digunakan sebagai variable berapa jumlah image yang dirata-rata. Karena jumlah citra dalam satu folder 100, maka jumGambar maksimal bernilai 100. Bitmap b digunakan sebagai media penyimpanan data image sementara selama waktu perulangan. Bitmap c digunakan sebagai media penyimpanan data image akhir hasil perhitungan.

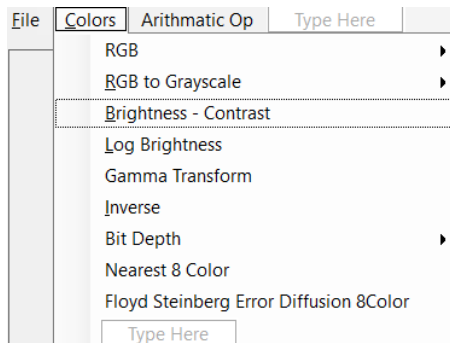
```

1 private void averageDenoisingToolStripMenuItem_Click(object sender, EventArgs e)
2 {
3     FolderBrowserDialog folderDlg = new FolderBrowserDialog();
4     folderDlg.ShowNewFolderButton = true;
5     if (folderDlg.ShowDialog() == DialogResult.OK)
6     {
7         Environment.SpecialFolder root = folderDlg.RootFolder;
8     }
9     List<Image> pictureArray = new List<Image>();
10    foreach (string item in
11        Directory.GetFiles(folderDlg.SelectedPath, "*.jpg", SearchOption.AllDirectories))
12    {
13        Image _image = Image.FromFile(item);
14        pictureArray.Add(_image);
15    }
16    pbInput.Image = pictureArray[0];
17    Bitmap b = new Bitmap((Bitmap)this.pbInput.Image);
18    Bitmap c = new Bitmap((Bitmap)this.pbInput.Image);
19    statusLabel2.Text = "Res. Citra: " + pbInput.Image.Width + " x " +
20    pbInput.Image.Height;
21    progressBar1.Visible = true;
22    int R, G, B, newR, newG, newB;
23    //nilai 50 berikut menunjukkan jumlah citra, yang diproses adalah 50 citra
24    int jumGambar = 50;
25    for (int i = 0; i < b.Width; i++)
26    {
27        for (int j = 0; j < b.Height; j++)
28        {
29            R = 0;
30            G = 0;
31            B = 0;
32            for (int k = 0; k < jumGambar - 1; k++)
33            {
34                b = (Bitmap)pictureArray[k];
35                Color c1 = b.GetPixel(i, j);
36                R = .....;
37                G = .....;
38                B += .....;
39            }
40            c.SetPixel(i, j, Color.FromArgb(.....,.....,.....));
41        }
42        progressBar1.Value = Convert.ToInt16(100 * (i + 1) / c.Width);
43    }
44    progressBar1.Visible = false;
45    this.pbOutput.Image = c;
46 }
47
48

```

2. Warna Terdekat

Untuk membuat warna terdekat, kita buat button / list menu 8 nearest color, yang jika ditekan maka proses perhitungan warna terdekat akan dilakukan.



Pada percobaan ini warna terdekat akan dibuat dalam fungsi warnaTerdekat dengan keluaran sebagai tipe data double dan masukannya adalah warna pixel yang diproses.

```

1 private static double warnaTerdekat(int pValueR, int pValueG, int pValueB)
2 {
3     double minDistance = .....;
4     int palColor, rDiff, gDiff, bDiff;
5     double pValueR1=0;
6     double distance;
7     //set warna pallete: hitam, merah, hijau, kuning, biru, cyan, magenta, putih
8     int[,] paletteColor = new int[,] { { 0, 0, 0 }, { ..... }, { ..... },
9     { ..... }, { ..... }, { ..... }, { ..... }, { ..... } };
10    for (palColor=0;palColor<=paletteColor.GetLength(0)-1;palColor++)
11    {
12        rDiff = .....;
13        gDiff = .....;
14        bDiff = .....;
15        distance = .....;
16        if (distance<minDistance)
17        {
18            minDistance = .....;
19            pValueR1 = .....;
20        }
21    }
22    return pValueR1;
23 }
24

```

Berikut adalah script untuk toolstrip menu nearest 8 color:


```

1 private void nearest8ColorToolStripMenuItem_Click(object sender, EventArgs e)
2 {
3     if (pbInput.Image == null)
4         MessageBox.Show("Tidak ada citra yang akan diolah");
5     else
6     {
7         double baru;
8         int[,] paletteColor = new int[,] { { 0, 0, 0 }, { ..... }, { ..... },
9         { ..... }, { ..... }, { ..... }, { ..... }, { ..... } };
10        Bitmap b = new Bitmap((Bitmap)this.pbInput.Image);
11        this.pbOutput.Image = b;
12        progressBar1.Visible = true;
13        for (int i = 0; i < b.Width; i++)
14        {
15            for (int j = 0; j < b.Height; j++)
16            {
17                Color c1 = b.GetPixel(i, j);
18                baru = warnaTerdekat(.....);
19                b.SetPixel(i, j, Color.FromArgb(.....));
20            }
21            progressBar1.Value = Convert.ToInt16(100 * (i + 1) / b.Width);
22        }
23        progressBar1.Visible = false;
24        this.pbOutput.Image = b;
25    }
26 }
27
28

```

3. Error Diffusion

Berikut adalah script untuk Floyd and Steinberg Error Diffusion:

```

1 private void floydSteinbergErrorDiffusion8ColorToolStripMenuItem_Click(object
2 sender, EventArgs e)
3 {
4     if (pbInput.Image == null)
5         MessageBox.Show("Tidak ada citra yang akan diolah");
6     else
7     {
8         int[,] paletteColor = {{0, 0, 0}, { ..... }, { ..... }, { ..... }, {
9         ..... }, { ..... }, { ..... }, { ..... }};
10        Bitmap b = new Bitmap((Bitmap)this.pbInput.Image);
11        this.pbOutput.Image = b;
12        int merah, hijau, biru;
13        double baru, errorR, errorG, errorB;
14        progressBar1.Visible = true;
15        for (int i = 0; i <= b.Width - 2; i++)
16        {
17            for (int j = 0; j <= b.Height - 2; j++)
18            {
19                merah = b.GetPixel(i, j).R;
20                hijau = b.GetPixel(i, j).G;
21                biru = b.GetPixel(i, j).B;
22                .....
23            }
24            progressBar1.Value = Convert.ToInt16(100 * (i + 1) / b.Width);
25        }
26        progressBar1.Visible = false;
27        this.pbOutput.Refresh();
28    }
29 }
30 }

```

4. Membuat Histogram

Siapkan ToolStrip Menu yang digunakan untuk menampilkan nilai histogram. Perhatikan gambar berikut. Pada menu dibawah ini, anda akan mencoba menampilkan histogram citra masukan, histogram citra keluaran, dan histogram citra masukan dan keluaran sekaligus.

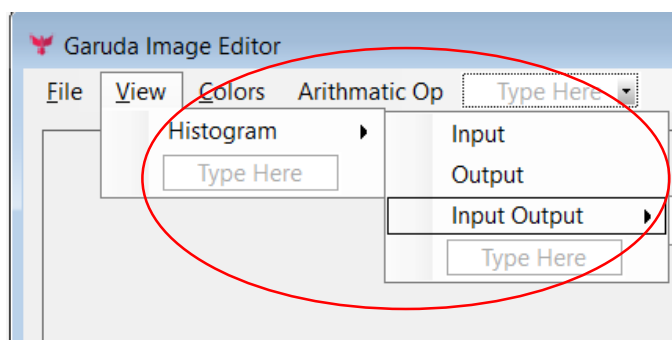
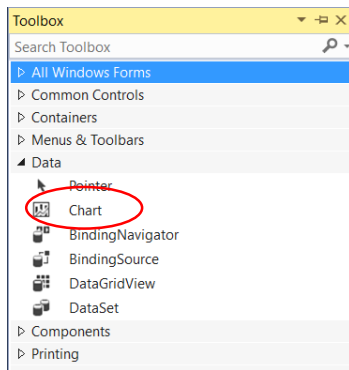
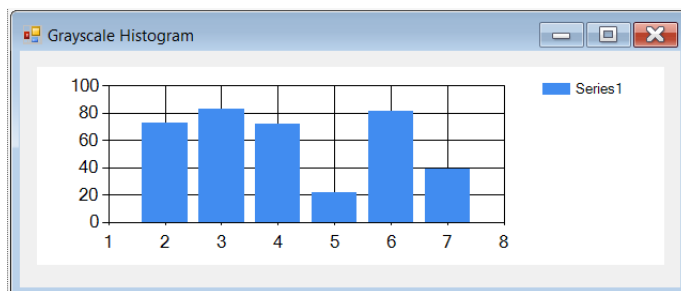


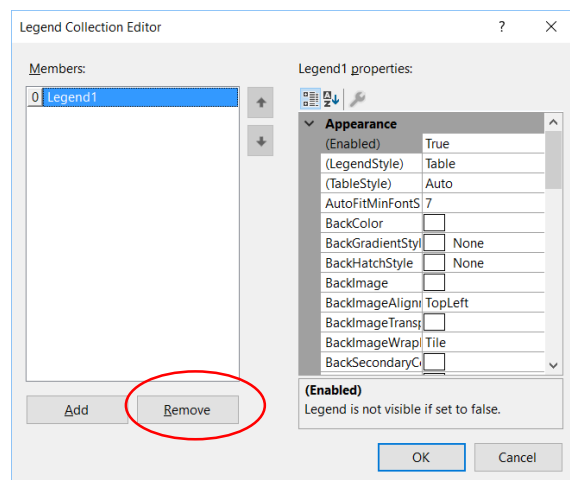
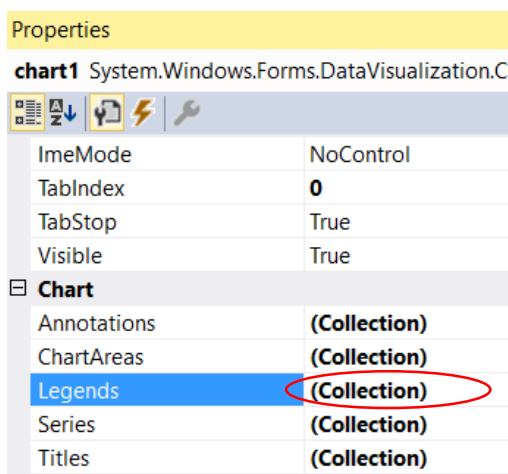
Chart ditambahkan dengan menggunakan Toolbox Chart yang ada pada menu Data didalam Toolbox. Perhatikan gambar berikut.



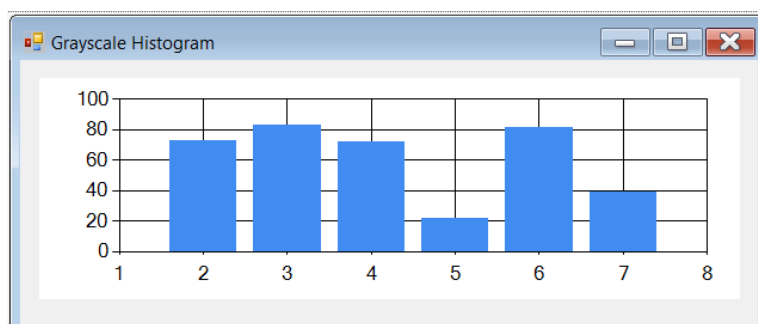
Drag Chart kedalam Form baru yang digunakan untuk menampilkan Histogram citra keabuan.



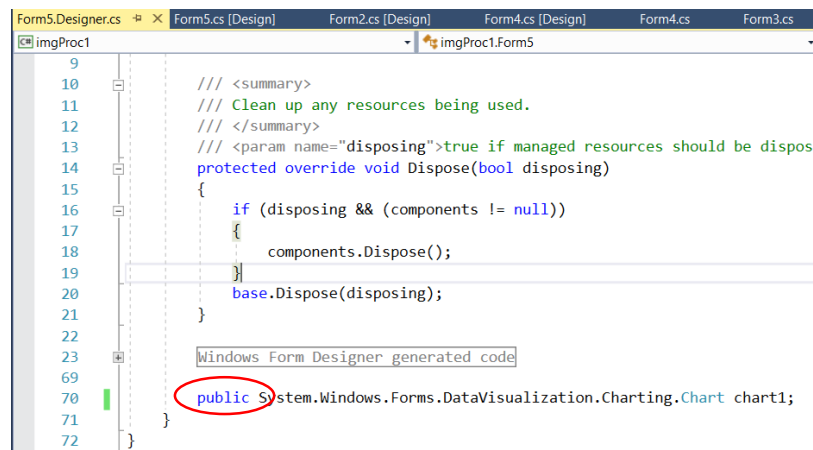
Pada grafik histogram, Legend tidak perlu ditampilkan sehingga enable legend dihilangkan (di Remove). Perhatikan gambar berikut.



Gambar berikut adalah tampilan chart setelah legend di-remove.



Set chart sebagai public pada form tersebut. Perhatikan gambar berikut.



Berikut adalah script untuk Grayscale Histogram:

```

1 private void inputToolStripMenuItem_Click(object sender, EventArgs e)
2 {
3     if (pbInput.Image == null)
4         MessageBox.Show("Tidak ada citra yang akan diolah");
5     else
6     {
7         //Data histogram disimpan kedalam Dictionary bertipe data Byte untuk
8         //intensitas pixel dan bertipe Double untuk nilai histogram tiap pixel
9         .....
10        //Variabel b (Bitmap) digunakan untuk menyimpan nilai citra masukan
11        Bitmap b = new Bitmap((Bitmap)this.pbInput.Image);
12        //Form5 adalah form untuk chart Histogram Grayscale
13        //Form6 adalah form untuk chart Histogram RGB
14        Form5 frm5 = new Form5();
15        Form6 frm6 = new Form6();
16        //Nilai awal pixel 0 - 255 diset bernilai 0
17        .....
18        //Untuk tiap baris dan kolom citra, nilai histogram ditambahkan
19        for (int i = 0; i <= 255; i++)
20        {
21            for (int j = 0; j <= 255; j++)
22            {
23                Color c1 = b.GetPixel(i, j);
24                //jika pada baris i kolom j, pixel bernilai n, maka nilai n pada
25                //dictionary ditambah 1
26                .....
27            }
28            progressBar1.Value = Convert.ToInt16(100 * (i + 1) / b.Width);
29        }
30        progressBar1.Visible = false;
31        //jika histogram R == G, maka kemungkinan besar citra grayscale
32        //jika grayscale maka form5 yang ditampilkan
33        if (histoR.Count == histoG.Count && !histoR.Except(histoG).Any())
34        {
35            //Variabel intensitas pixel pada dictionary dimapping ke dalam List kunci1
36            //bertipe Byte
37            .....
38            frm5.Show();
39            //tiga baris dibawah ini sengaja diset di code, agar tahu setting
40            //propertiesnya
41            frm5.chart1.Series["Series1"].Color = Color.Gray;
42            frm5.chart1.ChartAreas["ChartArea1"].AxisX.LabelStyle.Enabled = false;
43            frm5.chart1.ChartAreas["ChartArea1"].AxisY.LabelStyle.Enabled = false;
44            //Untuk tiap data dalam List kunci1

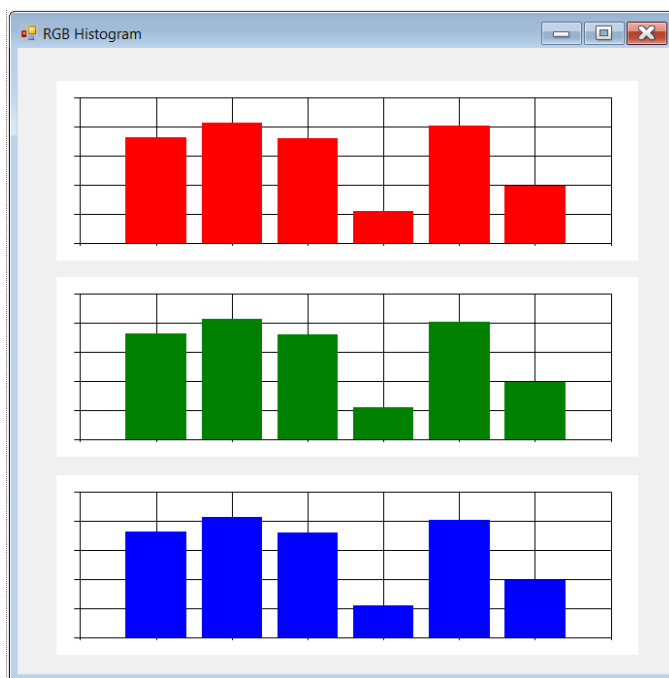
```

```

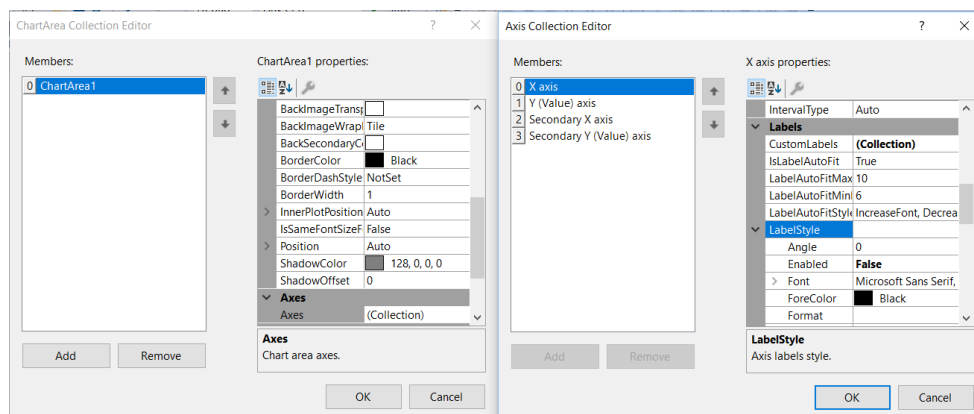
45     foreach (Byte key in kunci1)
46     {
47         //nilai histogram dinormalisasi
48         .....
49     }
50 } else
51 {
52     //lakukan proses histogram citra warna
53 }
54 }
55 }

```

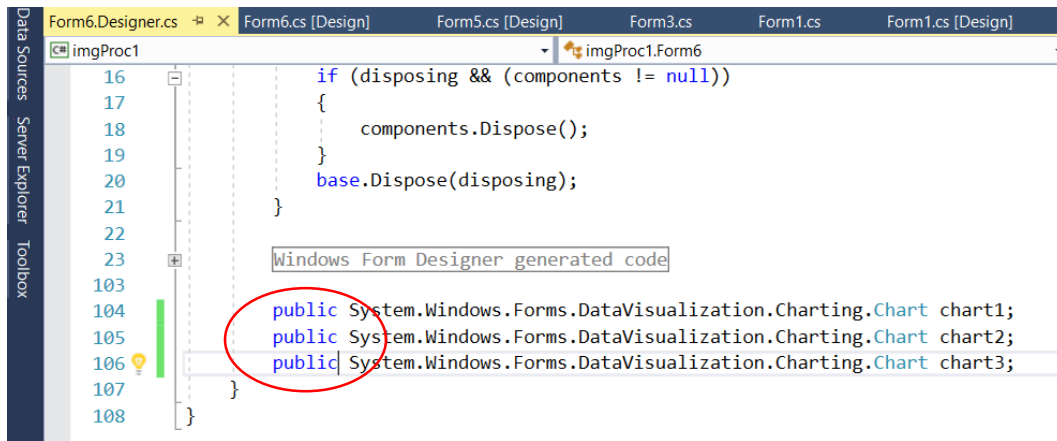
Pada tahap berikutnya, anda diminta untuk membuat histogram citra warna. Gambar berikut adalah form awal yang dibuat untuk menampilkan histogram warna dari citra



Set Color dan ChartArea dari chart sesuai dengan setting pada histogram grayscale yang ditunjukkan pada code grayscale diatas. (Color-> Red utk chart1, Color->Green untuk chart 2, Color->Blue untuk chart 3, LabelStyle->False).



Set tiap chart sebagai public, Perhatikan gambar berikut.



```
Form6.Designer.cs  Form6.cs [Design]  Form5.cs [Design]  Form3.cs  Form1.cs  Form1.cs [Design]
imgProc1
16  if (disposing && (components != null))
17  {
18      components.Dispose();
19  }
20  base.Dispose(disposing);
21  }
22
23  Windows Form Designer generated code
103
104  public System.Windows.Forms.DataVisualization.Charting.Chart chart1;
105  public System.Windows.Forms.DataVisualization.Charting.Chart chart2;
106  public System.Windows.Forms.DataVisualization.Charting.Chart chart3;
107
108  }
```

A. TUGAS PRAKTIKUM

Buat Histogram citra warna.

--- SELAMAT BELAJAR ---