MODUL 7 – Histogram Equalization dan Fuzzy Histogram Equalization

A. TUJUAN

- Mahasiswa dapat membuat aplikasi perbaikan citra digital dengan metode Histogram Equalization
- Mahasiswa dapat membuat aplikasi perbaikan citra digital dengan metode Fuzzy Histogram Equalization

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

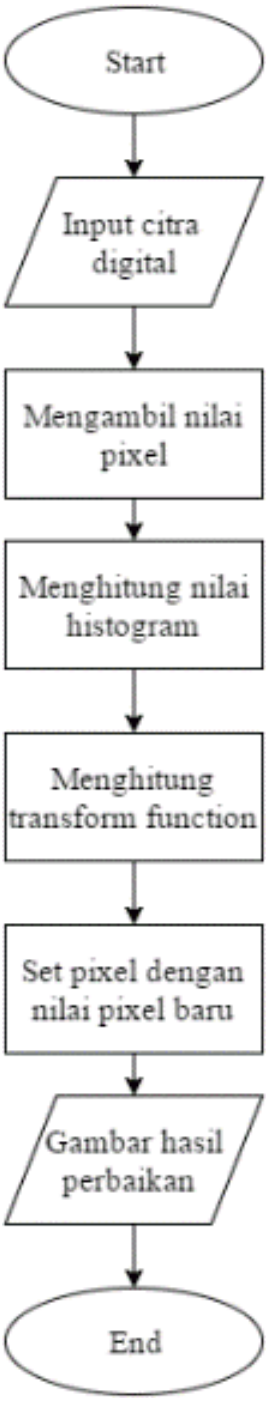
D. DASAR TEORI

- Histogram Equalization

- *Histogram Equalization* adalah persamaan yang diimplementasikan pada sebuah citra *digital* di mana distribusi histogramnya akan lebih menyebar, dalam hal ini walaupun tidak dapat dibuktikan bahwa bentuk histogram-nya akan seragam namun dengan *Histogram Equalization* dapat dipastikan histogram-nya akan lebih merata. Perataan histogram diperoleh dengan mengubah derajat keabuan sebuah piksel (r) dengan derajat keabuan yang baru (s) dengan sebuah fungsi transformasi (T) (Gonzalez & Woods, 2002). Histogram Equalization dilakukan dengan membuat transform function dari histogram image yang telah dinormalisasi. Berikut adalah persamaan untuk transform function:

  $$G(z_q) = (L-1)\sum_{1=0}^{q} p_z(z_i)$$

  Gambar 1 menunjukkan proses perbaikan citra digital menggunakan Histogram Equalization.



Gambar 1. Flowchart jalannya perbaikan citra digital dengan HE

- Perhitungan manual untuk *Histogram Equalization* kali ini menggunakan 3-bit *image* (L = 8) dengan size 64x64 piksel (MN = 4096) dan distribusi intensitas akan diperlihatkan pada Tabel 1, dimana level intensitas adalah berupa nilai integer pada jarak [0, L – 1] = [0, 7]. Pada Gambar 2 ditunjukkan original histogram sebelum dilakukan proses ekualisasi. Pada Gambar 3 menunjukkan hasil transform function Pada Gambar 4 ditunjukkan histogram hasil penghitungan dengan metode HE. Penghitungan *Histogram Equalization* menggunakan rumus:

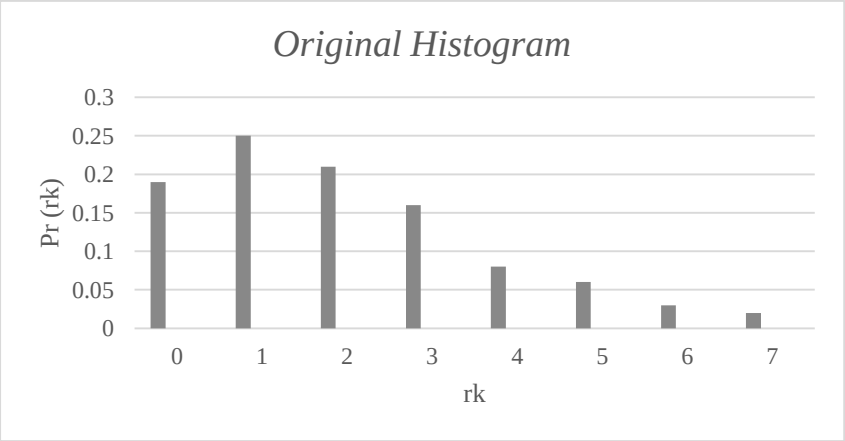
$$s_0 = T(r_0) = 7\sum_{j=0}^{1} p_r(r_j) = 7\ p_r\ (r_0) = 1.33$$

$$s_1 = T(r_1) = 7\sum_{j=0}^{1} p_r(r_j) = 7\ p_r\ (r_0) + \ 7\ p_r\ (r_1) = 3.08$$

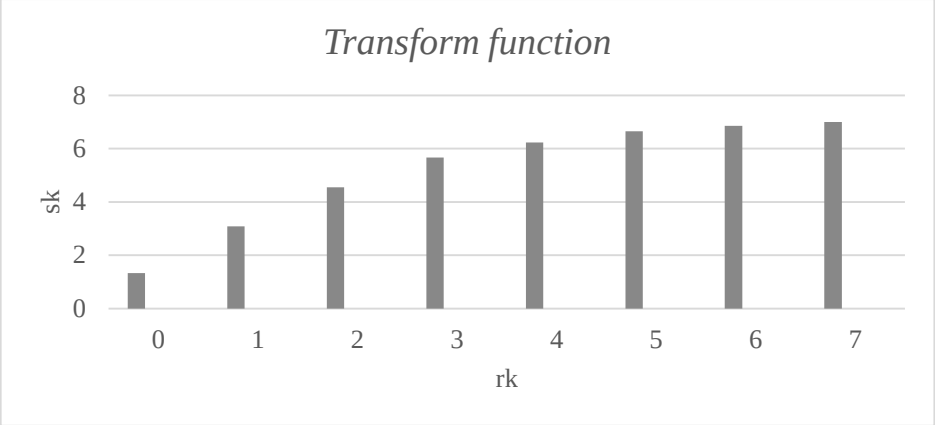
dan $s_2 = 4.55,\ s_3 = 5.67,\ s_4 = 6.23,\ s_5 = 6.65,\ s_6 = 6.86,\ s_7 = 7.00$

Tabel 1 Distribusi intensitas dan nilai histogram untuk 3-bit *image*, 64 x 64 citra *digital*

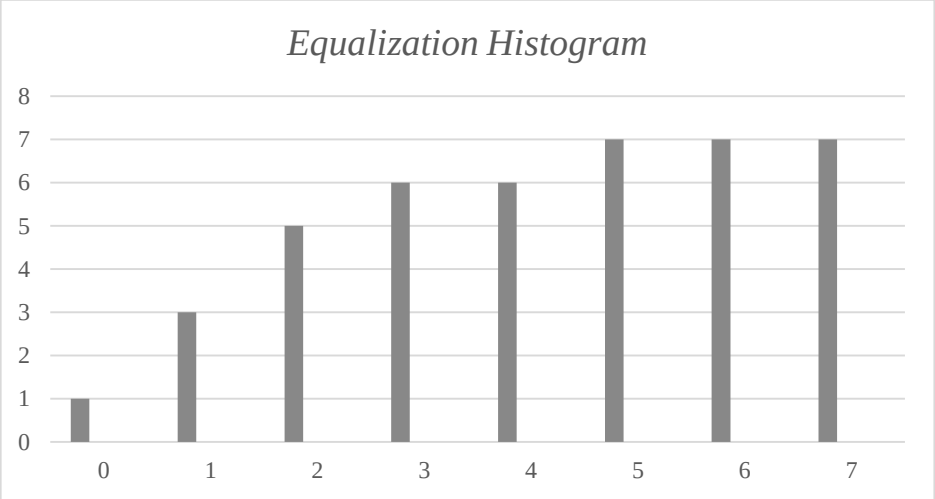
r_k	n_k	$p_{r\ (r_k)= n_{k/MN}}$
$r_{0=0}$	790	0.19
$r_{1=1}$	1023	0.25
$r_{2=2}$	850	0.21
$r_{3=3}$	656	0.16
$r_{4=4}$	329	0.08
$r_{5=5}$	245	0.06
$r_{6=6}$	122	0.03
$r_{7=7}$	81	0.02



Gambar 2. Original histogram sebelum dilakukan proses ekualisasi



Gambar 3. Transform function



Gambar 4. Histogram hasil penghitungan dengan metode Equalization Histogram

Pada point ini, s memiliki nilai berbentuk pecahan maka nilai tersebut dibulatkan pada nilai integer yang paling dekat.

s_0 = 1.33 → 1

s_1 = 3.08 → 3

s_2 = 4.55 → 5

s_3 = 5.67 → 6

s_0 = 6.23 → 6

s_0 = 6.65 → 7

s_0 = 6.86 → 7

s_0 = 7.00 → 7

Berikut adalah pseudo-code untuk membuat Histogram Equalization:

- Membuat transform-function

```
List<byte> kunci = histo.Keys.ToList(); //variable kunci
digunakan menyimpan nilai dari histogram berupa byte
foreach (byte key in kunci)
{
   histo[key] = histo[key] / (image1.Width * image1.Height);
   jum += 255 * histo[key];
   s[key] = jum;
}
jum = 0; //pengulangan sebanyak 3 kali untuk RGB
```

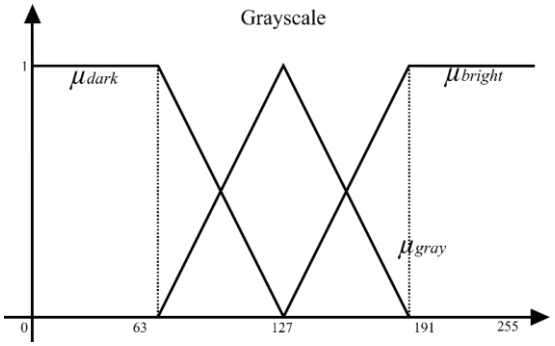
- Proses mengubah nilai pixel ke nilai baru sesuai transform function

```
for (baris = 0; baris < image1.Width; baris++)
{
  for (kolom = 0; kolom < image1.Height; kolom++)
  {
    R = image1.GetPixel(baris, kolom).R;
    G = image1.GetPixel(baris, kolom).G;
    B = image1.GetPixel(baris, kolom).B;
    int s = Convert.ToInt16(s1[R]);
    int ss = Convert.ToInt16(s2[G]);
    int sss = Convert.ToInt16(s3[B]);
    image1.SetPixel(baris, kolom, Color.FromArgb(s, ss, sss));
  }
}
```

- **Fuzzy Histogram Equalization**

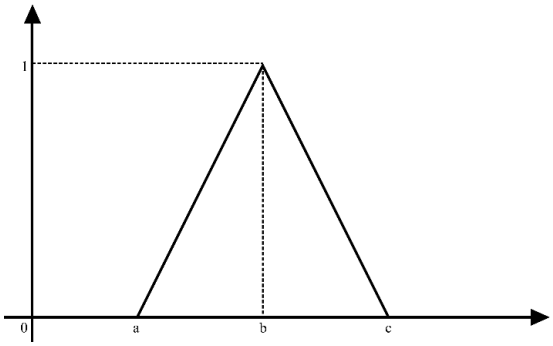
- *Fuzzy Histogram Equalization* adalah suatu proses perataan histogram, dimana distribusi nilai derajat keabuan pada suatu citra dibuat rata. Tujuan perbaikan citra dengan menggunakan FHE untuk **mempertahankan bentuk** histogram aslinya. Pada tahap ini proses perataan histogram menggunakan metode Fuzzy. Dalam melakukan perataan histogram menggunakan FHE, nilai derajat keabuannya ditentukan oleh metode fuzzy. *Fuzzy Image Processing* mempunyai tiga tahap utama: *image fuzzification*, *image rule evaluation / inference*, dan *image defuzzification*. Fuzzy Histogram Equalization terkadang lebih dipilih daripada Histogram Equalization karena beberapa alasan, diantaranya adalah lebih cocok digunakan pada image *low contrast*. Alasan lain yang jauh lebih penting adalah jika pada Histogram Equalization, histogramnya telah berbeda jauh antara citra keluaran dengan citra masukan. Pada Fuzzy Histogram Equalization histogramnya berubah, tetapi bentuk histogram masih mirip antara citra masukan dengan citra keluaran. Karena mirip, jika dilakukan operasi berikutnya (transformasi citra), tidak akan perubahan citra yang signifikan.

- *Fuzzification* adalah mengubah masukan-masukan yang nilai kebenarannya bersifat pasti (crisp input) ke dalam bentuk fuzzy input. Hasil dari proses *Fuzzification* ini adalah fuzzy input. Oleh sebab itu langkah pertama yang harus dilakukan adalah mendesain himpunan-himpunan fuzzy yang disajikan dalam bentuk fungsi keanggotaan. Fungsi keanggotaan *grayscale* diwakili oleh *dark*, *gray* dan *bright*. Pada Gambar 5 ditunjukkan fungsi keanggotaan *grayscale*.



Gambar 5. Fungsi keanggotaan grayscale

Dalam proses *Fuzzification* ini akan dihitung fungsi keanggotaan menggunakan rumus tertentu untuk menghasilkan fuzzy output. Rumus fungsi keanggotaan yang digunakan dalam penelitian ini adalah rumus fungsi keanggotaan segitiga. Gambar 6 menunjukkan fungsi keanggotaan segitiga.



Gambar 6. Fungsi keanggotaan segitiga

Rumus mencari fuzzy input menggunakan fungsi keanggotan segitiga:

(x-a) / (b-a)	a < x ≤ b
-(x-c) / (c-b)	b < x ≤ c

dengan nilai x adalah inputan.

- *Image* Rule Evaluation / Inference
  Inference: melakukan penalaran menggunakan fuzzy input dan fuzzy rules yang telah ditentukan sehingga menghasilkan fuzzy output. Secara sintaks, suatu fuzzy rule (aturan fuzzy) dituliskan sebagai:

IF antecedent THEN consequent

Setelah dilakukan proses pembentukan membership function dilakukan proses pembentukan rule atau proses inference. Disini nantinya akan terbentuk 3 penilaian untuk membership function *Grayscale* yaitu *dark*, *gray*, dan *bright*.

Rule untuk *Grayscale* adalah:

IF a piksel is *Dark*, THEN make it *dark*er

IF a piksel is *Gray*, THEN make it *gray*

IF a piksel is *Bright*, THEN make it *bright*er

Dari proses rule / inference ini akan menghasilkan fuzzy output. Salah satu model penalaran yang banyak dipakai adalah penalaran max-min. Pada penelitian ini, menggunakan min atau mencari nilai terkecil.

$$\mu_{A\cap B}[x] \quad = \min(\mu_A[x], \mu_B[x])$$

- *Image DeFuzzification*

  DeFuzzification adalah proses pemetaan dari mengubah fuzzy output menjadi crisp value berdasarkan fungsi keanggotaan yang telah ditentukan. Terdapat berbagai metode *deFuzzification* yang telah berhasil diaplikasikan untuk berbagai macam masalah, Dalam penelitian ini, pada proses defuzzifikasi menggunakan metode Weight of Average. Metode ini mengambil nilai rata-rata dengan menggunakan pembobotan berupa derajat keanggotaan. Sehingga y* didefinisikan sebagai:

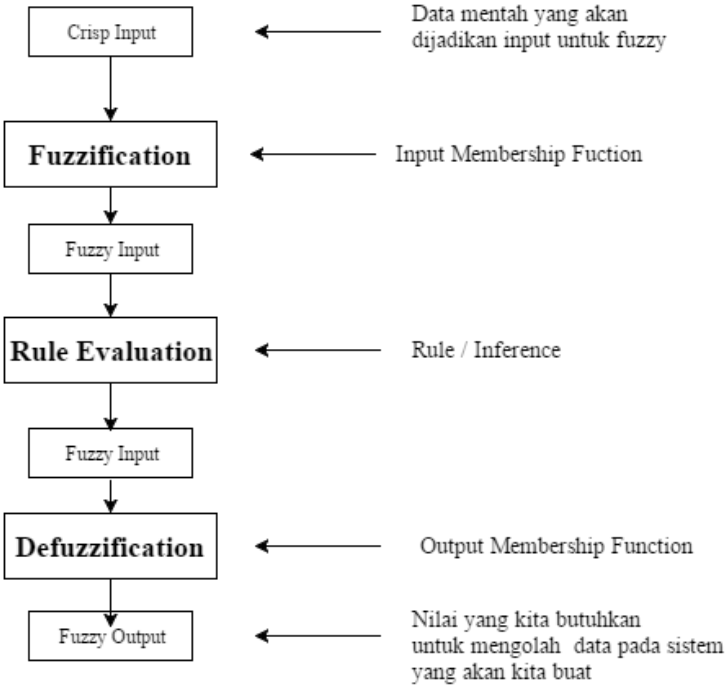
$$y* = \sum \frac{\mu(y)y}{\mu(y)}$$

  di mana y adalah nilai crisp dan μ(y) adalah derajat keanggotan dari nilai crisp y. Hasil dari proses Defuzzifikasi ini adalah fuzzy output yang pada akhirnya nilai dari fuzzy output akan diproses pada sistem.

  Dari sekian banyak metode deffuzzifikasi, dalam proses perbaikan citra ini menggunakan metode Weight of Average (WoA). Pada proses perbaikan citra ini membutuhkan nilai numerik untuk nilai piksel baru guna diimplementasikan pada piksel yang akan diperbaiki, maka menggunakan metode WoA pada proses deffuzikfikasi dinilai paling tepat. Rumus deffuzifikasi menggunakan metode WoA adalah:

$$v0 = \frac{\mu dark(Z0)\,x\,Vd \;+\; \mu gray(Z0)\,x\,Vg \;+\; \mu bright(Z0)\,x\,Vb}{\mu dark(Z0) \;+\; \mu gray(Z0) \;+\; \mu bright(Z0)}$$

Gambar 7 berikut menunjukkan diagram proses fuzzy secara keseluruhan



Gambar 7. Diagram proses fuzzy secara keseluruhan

Berikut adalah pseudo-code untuk membuat Fuzzy Histogram Equalization:

- Proses Fuzzifikasi dan Rule Evaluation

1	`GrayscaleAvg = R + G + B / 3`
2	`cAbu = GrayscaleAvg`

3	`cAbu = (R + G + B) / 3;`
	`//Grayscale`
	`if (cAbu >= 0 && cAbu <= 63)`
	`{`
	`  batasAbuA = 0;`
	`  batasAbuB = 63;`
	`  woa = 0;`
	`  wob = 0;`
	`}`
	`else if (cAbu >= 64 && cAbu <= 126)`
	`{`
	`  batasAbuA = 63;`
	`  batasAbuB = 127;`
	`  woa = 0;`
	`  wob = 127;`
	`}`
	`else if (cAbu >= 128 && cAbu <= 190)`
	`{`
	`  batasAbuA = 127;`
	`  batasAbuB = 191;`
	`  woa = 127;`

```
    wob = 255;
}

else if (cAbu >= 191 && cAbu <= 255)
{
  batasAbuA = 191;
  batasAbuB = 255;
  woa = 255;
  wob = 255;
}
```

```
4  //Proses Fuzzification untuk menghasilkan fuzzy input
   hasilFuzzy1 = (double)(cAbu - batasAbuA) / (double)(batasAbuB -
   batasAbuA);

   hasilFuzzy2 = -(double)(cAbu - batasAbuB) / (double)(batasAbuB
   - batasAbuA);

   FuzzificationAbu1 = Math.Round(hasilFuzzyAbu1, 6);
   FuzzificationAbu2 = Math.Round(hasilFuzzyAbu2, 6);
```

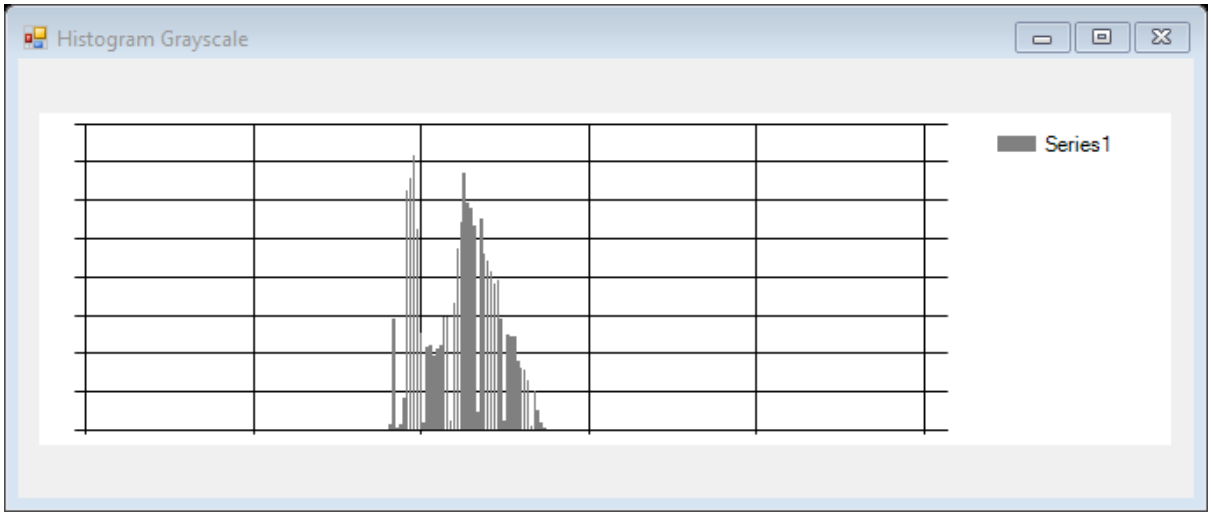
- Proses Defuzzifikasi

```
1  //Proses Defuzzifikasi
2  if (cAbu >= 0 && cAbu <= 63)
3  {
4     woaAbu = 0;
5  }
6  else if (cAbu == 127)
7  {
8     woaAbu = 127;
9  }
10 else if (cAbu >= 191 && cAbu <= 255)
11 {
12    woaAbu = 255;
13 }
14 else if (cAbu >= 64 && cAbu <= 126)
15 {
16    woaAbu = (double)((hasilFuzzyAbu1 * wob) + (hasilFuzzyAbu2 * woa)) /
17 (double)(hasilFuzzyAbu1 + hasilFuzzyAbu2);
18 }
19 else if (cAbu >= 128 && cAbu <= 190)
20 {
21    woaAbu = (double)((hasilFuzzyAbu1 * wob) + (hasilFuzzyAbu2 * woa)) /
22 (double)(hasilFuzzyAbu1 + hasilFuzzyAbu2);
23 }
24 else
25 {
26    woaAbu = 0;
27 }
28    int woaAbuFix = Convert.ToInt16(Math.Round(woaAbu, 0));
```

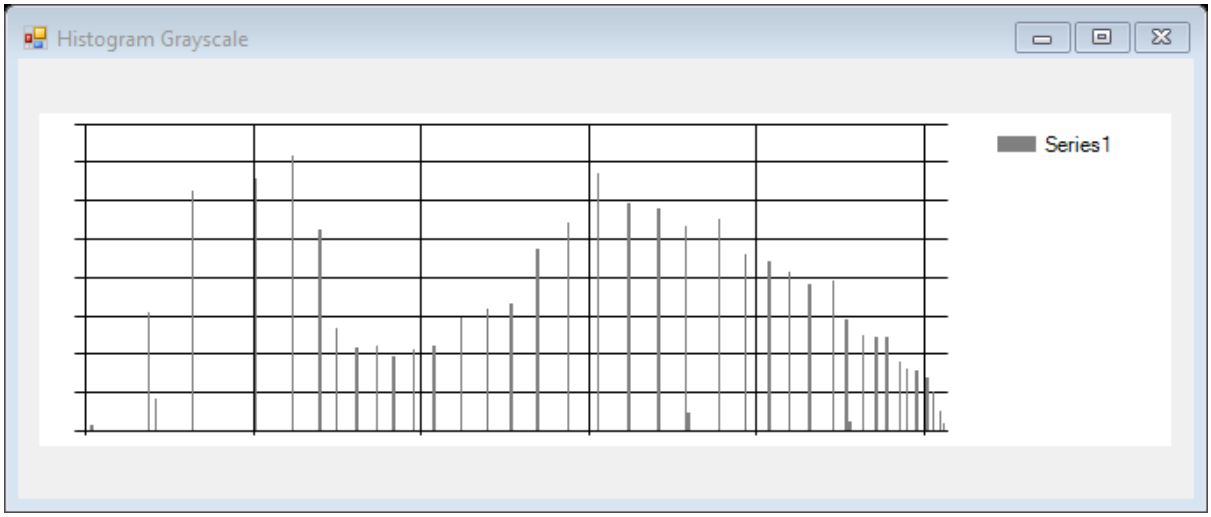
```
29      image2.SetPixel(baris, kolom, Color.FromArgb(woaAbuFix, woaAbuFix,
30  woaAbuFix));
```

Contoh Aplikasi Perbaikan dengan HE dan FHE

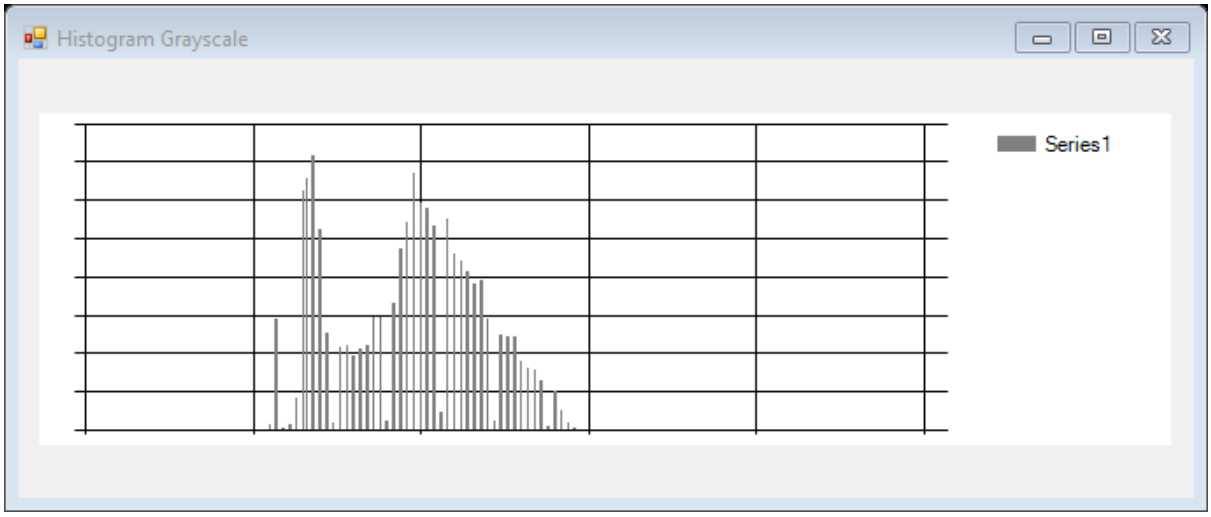
Histogram Input



Histogram HE Grayscale

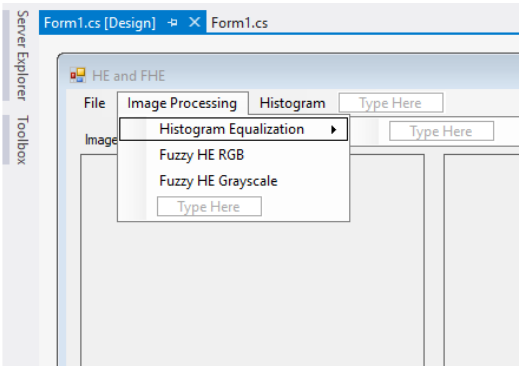


Histogram FHE Grayscale



E. Praktikum Histogram Equalization

- Histogram Equalization



```
1   //Histogram Equalization
2   private void histogramEqualizationToolStripMenuItem_Click(object sender,
3   EventArgs e)
4   {
5     if ((pb1.Image == null))
6     {
7       MessageBox.Show("Error!! Empty Input Image");
8     }
9     else
10    {
11      Dictionary<byte, double> histoR = new Dictionary<byte, double>();
12      Dictionary<byte, double> histoG = new Dictionary<byte, double>();
13      Dictionary<byte, double> histoB = new Dictionary<byte, double>();
14      Bitmap image1 = new Bitmap(pb1.Image);
15      pb2.Image = image1;
16      int baris, kolom;
17      Form3 frm3 = new Form3(); //RGB
18      Form2 frm2 = new Form2(); //Grayscale
19      byte c1, c2, c3;
20      double[] s1 = new double[256];
21      double[] s2 = new double[256];
22      double[] s3 = new double[256];
23      double jum = 0;
24      //Proses inisiasi nilai awal histogram
25      for (int counter = 0; counter <= 255; counter++)
26      {
27        histoR[(Byte)counter] = 0.0;
28        histoG[(Byte)counter] = 0.0;
29        histoB[(Byte)counter] = 0.0;
30      }
31      // Proses mendapatkan nilai histogram
32      for (baris = 0; baris < image1.Width; baris++)
33      {
34        for (kolom = 0; kolom < image1.Height; kolom++)
35        {
36          c1 = image1.GetPixel(baris, kolom).R;
37          c2 = image1.GetPixel(baris, kolom).G;
38          c3 = image1.GetPixel(baris, kolom).B;
39          if (histoR.ContainsKey(c1))
40          {
41             histoR[c1] += 1;
42          }
43          if (histoG.ContainsKey(c2))
44          {
45             histoG[c2] += 1;
46          }
47          if (histoB.ContainsKey(c3))
48          {
49             histoB[c3] += 1; //kerja histogram
50          }
51        }
52    }
```

```
1   //Proses menghitung nilai transform function,
2   List<byte> kunci1 = histoR.Keys.ToList();
3   List<byte> kunci2 = histoG.Keys.ToList();
4   List<byte> kunci3 = histoB.Keys.ToList();
5
6   foreach (byte key in kunci1)
7   {
8     histoR[key] = histoR[key] / (image1.Width * image1.Height);
9     jum += 255 * histoR[key];
10    s1[key] = jum;
11  }
12    jum = 0;
13    foreach (byte key in kunci2)
14  {
15    histoG[key] = histoG[key] / (image1.Width * image1.Height);
16    jum += 255 * histoG[key];
17    s2[key] = jum;
18  }
19    jum = 0;
20    foreach (byte key in kunci3)
21  {
22    histoB[key] = histoB[key] / (image1.Width * image1.Height);
23    jum += 255 * histoB[key];
24    s3[key] = jum;
25  }
26  progressBar1.Show();
27  //Proses mengubah nilai pixel ke nilai baru sesuai transform function
28  for (baris = 0; baris < image1.Width; baris++)
29  {
30    for (kolom = 0; kolom < image1.Height; kolom++)
31    {
32    c1 = image1.GetPixel(baris, kolom).R;
33    c2 = image1.GetPixel(baris, kolom).G;
34    c3 = image1.GetPixel(baris, kolom).B;
35    int s = Convert.ToInt16(s1[c1]);
36    int ss = Convert.ToInt16(s2[c2]);
37    int sss = Convert.ToInt16(s3[c3]);
38    image1.SetPixel(baris, kolom, Color.FromArgb(s, ss, sss));
39    }
40  progressBar1.Value = Convert.ToInt32(Math.Floor((double)(100 * (baris + 1) /
41  image1.Width)));
42  }
43  progressBar1.Hide();
44  pb2.Refresh();
45  }
46  }
```

Tugas

1. Buat Fuzzy Histogram Equalization