PRAKTIKUM 6

Ekualisasi/Perataan Histogram

Tujuan :

Mahasiswa mengetahui cara membuat program Operasi Perbaikan citra dengan metode Ekualisasi atau perataan histogram.

Dasar Teori :

Ekualisasi histogram merupakan suatu cara yang bertujuan untuk memperoleh histogram yang intensitasnya terdistribusi secara seragam pada citra. Namun, dalam praktik, hasilnya tidak benar-benar seragam (Jain, 1989). Pendekatan yang dilakukan adalah untuk mendapatkan aras keabuan yang lebih luas pada daerah yang memiliki banyak piksel dan mempersempit aras keabuan pada daerah yang berpiksel sedikit. Efeknya dapat digunakan untuk meningkatkan kontras secara menyeluruh. Perlu diketahui, ekualisasi histogram termasuk sebagai pemetaan nonlinear.

Misalnya, histogram untuk setiap aras keabuan dinyatakan dengan

hist [i+1]

Dalam hal ini, i bernilai 0, 1, 2, .., L-1, dengan L menyatakan jumlah aras keabuan. Akumulasi histogram untuk piksel yang memiliki aras k dinyatakan dengan

$$c[k+1]= \sum_{i=1}^{k} \llbracket hist[k+1], k=0,1,2,\ldots,L-1 \rrbracket \qquad (6.1)$$

Selanjutnya, aras k akan diganti dengan a dengan ketentuan sebagai berikut:

$$a\_k=round((L-1)\ \ c[k+1]/N), k=0,1,2,\ldots,L-1 \qquad (6.2)$$

Dalam hal ini, N menyatakan jumlah piksel pada citra.

Untuk memahami proses dalam ekualisasi histogram, lihatlah contoh pada Tabel 6.1.

Tabel 6.1 Proses ekualisasi histogram

I	Aras	hist[i]	c[i]	a(i)
1	0	2	2	0

2	1	2	4	0
3	2	4	8	0
4	3	12	20	2
5	4	18	38	4
6	5	14	52	6
7	6	10	62	7
8	7	2	64	7
	L=8	N=64		

Pada contoh di atas, yang diarsir dengan warna hijau muda menyatakan keadaan awal citra. Dalam hal ini, citra mengandung N=64 piksel (8x8) dengan jumlah aras keabuan berupa 8. Selanjutnya, berdasarkan nilai hist[i] maka c[i] dihitung. Selanjutnya, a[i] dapat dihitung berdasar Persamaan 3.9. Dalam hal ini, setiap nilai

0 atau 1 pada citra akan diganti dengan 0;

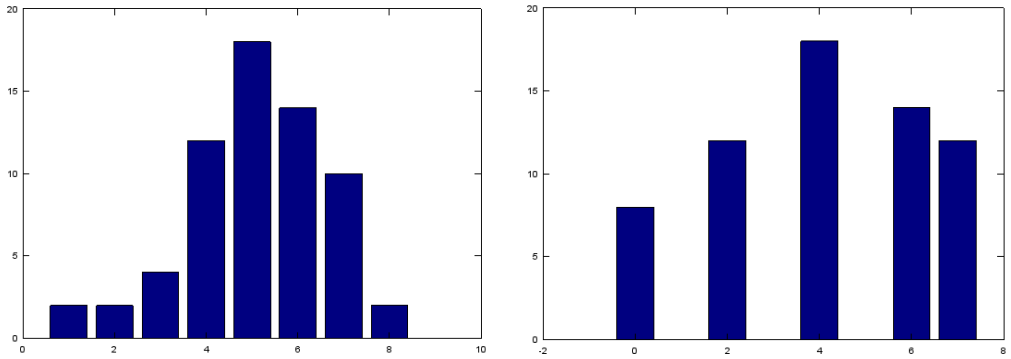
3 akan diganti dengan 2;

4 tidak diganti (tetap);

5 diganti dengan 6;

6 dan 7 diganti dengan 7.

Gambar 3.18 memperlihatkan keadaan sebelum dan sesudah ekualisasi histogram. Tampak bahwa di sekitar batang histogram yang paling tinggi terjadi perenggangan dan perbedaan dengan yang lebih rendah mengecil.



(a) Histogram awal (b) Histogram setelah ekualisasi

Gambar 6.1 Efek ekualisasi histogram

Algoritma untuk melakukan penggantian nilai intensitas pada citra ditunjukkan berikut ini.

ALGORITMA 6.1 – Melaksanakan ekualisasi histogram citra aras keabuan

Masukan:

- f(M, N) : citra berukuran M baris dan N kolom
- n : jumlah piksel dalam citra

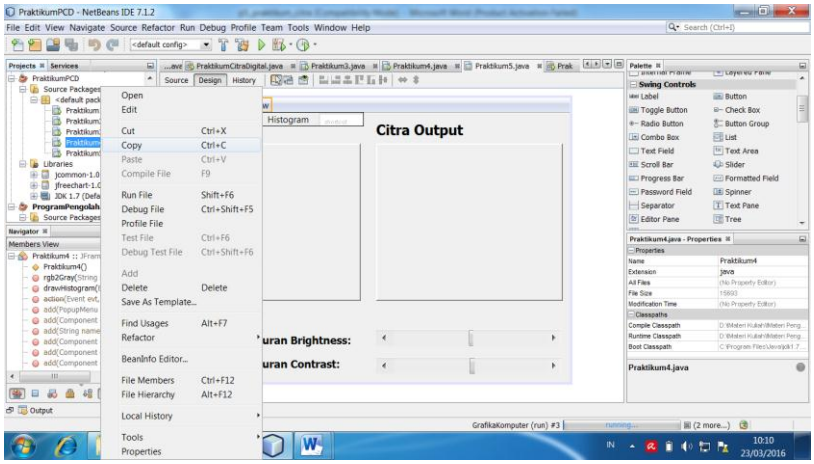
Keluaran

- g(M, N) : citra yang telah mengalami ekualisasi histogram

1. Hitung faktor penyekalaan: α ← 255 / n
2. Hitung histogram citra menggunakan Algoritma 3.1 dengan hasil berupa hist
3. c[1] ← α * hist[1]
4. FOR i←1 TO L-1
5. c[i+1] ← c[i] + round(α * hist[i+1])
6. END-FOR
7. FOR i ← 1 TO M
8. FOR j ← 1 TO N
9. g(y,x) ← c[f(y, x)]
10. END-FOR
11. END-FOR

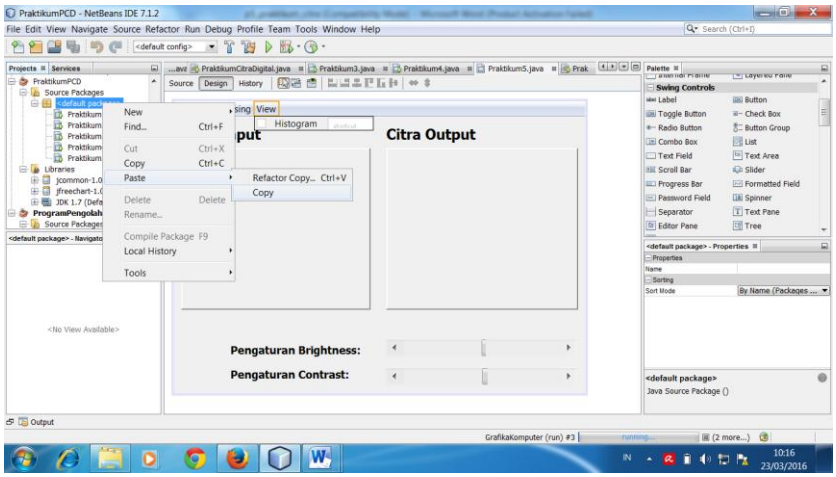
Langkah Praktikum :

1). Copy class JFrame Form pada praktikum sebelumnya dengan cara klik kanan pada folder Source Package Praktikum6 lalu pilih Copy seperti tampak pada gambar 6.2 berikut:



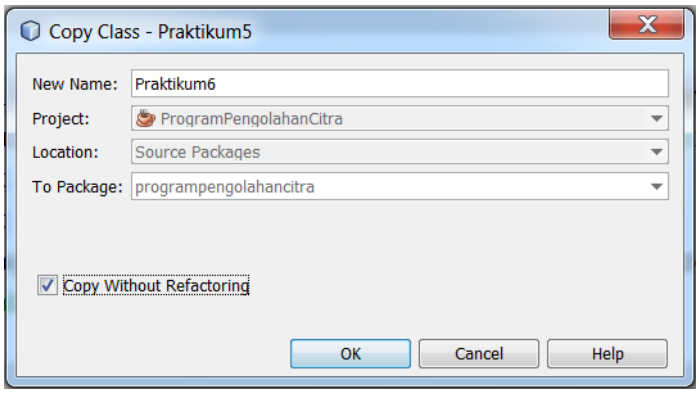
Gambar 6.2. Copy class JFrame Form

2). Kemudian klik kanan pada folder source packages pada <default Packages>, kemudian pilih Paste kemudian pilih Copy, seperti tampak pada gambar 6.3.



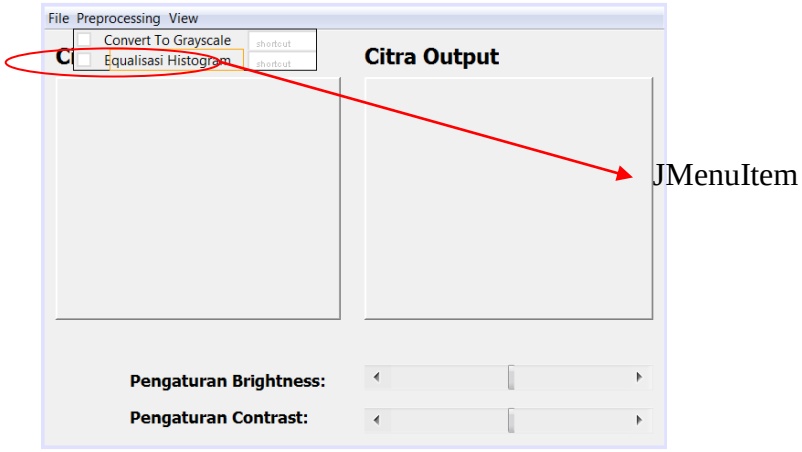
Gambar 6.3. Paste ke class JFrame Form baru

3). Rubah nama class menjadi Praktikum6 kemudian centang pilihan Copy Without Refactoring. Kemudian klik OK. Seperti tampak pada gambar 6.4. kemudian rubah nama class dari Praktikum5_1 menjadi Praktikum6.



Gambar 6.4. Jendela Dialog Copy Class

4). Klik Design, kemudian tambahkan 1 buah komponen JMenuItem dan tambahkan pada Menu Preprocessing kemudian rubah text nya menjadi Equalisasi Histogram. Seperti tampak dalam gambar 6.5. berikut:



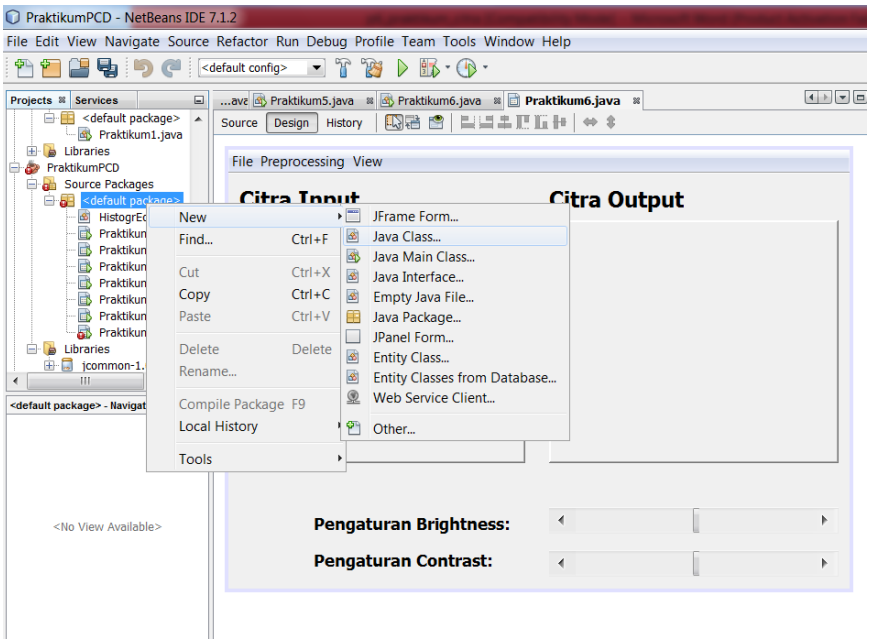
Gambar 6.5. Design Form

5). Tambahkan Kode metode untuk proses Equalisasi Histogram pada bagian Source Praktikum6 sebagai berikut.

```
private void jMenuItem5ActionPerformed(java.awt.event.ActionEvent evt) {
    HistogrEqualisation histEq = new HistogrEqualisation();
    BufferedImage gbInput = loadImage(sumber);
    BufferedImage gbEq;
    int ukuranX = gbInput.getWidth();
    int ukuranY = gbInput.getHeight();
    int size = ukuranX * ukuranY;
    int grayScale [][] = new int[ukuranX][ukuranY];
    int histogram [] = new int[256];
    int cdf [] = new int[256];
    float equalized [] = new float[256];
    float picEqualized [][] = new float[ukuranX][ukuranY];
```

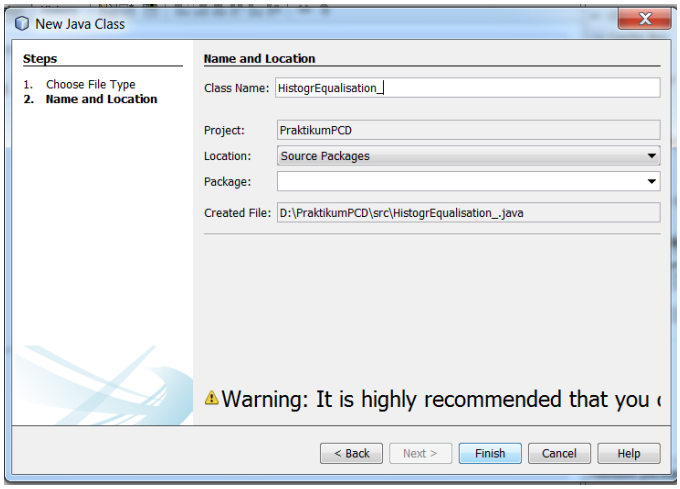
```
        grayScale = histEq.ArrayGray(sumber);
        histogram = histEq.histogram(grayScale,ukuranX, ukuranY);
        cdf = histEq.getCDF(histogram);
        equalized = histEq.equalization(cdf, size);
        picEqualized = histEq.gbEqualiz(grayScale, equalized, ukuranX, ukuranY);
        gbEq = histEq.gbHasil(picEqualized, ukuranX, ukuranY);
        int counter = 0;
        int x = jLabel2.getWidth();
        int y = jLabel2.getHeight();
        ImageIcon imageIcon = new ImageIcon(resize(gbEq, x, y));
        jLabel2.setIcon(imageIcon);
    }
```

6). Klik kanan Pada Folder Package Project PraktikumPCD , kemudian klik new dan pilih Java Class seperti gambar 6.6. berikut:



Gambar 6.6. Menambahkan Java Class baru

7). Pada jendela dialog New Java Class, Isi nama class dengan HistogrEqualisation, lalu klik Finish seperti gambar 6.7. berikut:



Gambar 6.7. Menentukan nama Java Class baru

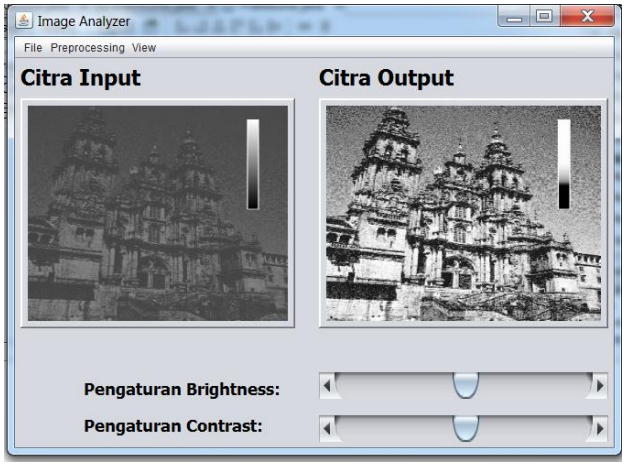
8). Ketik source code class HistogrEqualisation dengan kode berikut:

```
import java.awt.Graphics;
import java.awt.image.BufferedImage;
import java.awt.image.WritableRaster;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;

public class HistogrEqualisation {
    public static BufferedImage loadImage(String ref) {
        BufferedImage bimg = null;
        try {
            bimg = ImageIO.read(new File(ref));
        } catch (Exception e) {
            e.printStackTrace();
        }
        return bimg;
    }

    public int[][] ArrayGray(String sumber) {
        BufferedImage gbRgb = loadImage(sumber);
        int ukuranX = gbRgb.getWidth();
        int ukuranY = gbRgb.getHeight();
        int[][] arrayGray = new int[ukuranX][ukuranY];
        for (int x = 0; x < ukuranX; x++) {
            for (int y = 0; y < ukuranY; y++) {
                int rgb = gbRgb.getRGB(x, y);
                int alpha = (rgb << 24) & 0xff;
                int merahg = (rgb >> 16) & 0xff;
                int hijaug = (rgb >> 8) & 0xff;
                int birug = (rgb >> 0) & 0xff;
                float gray = (float) ((0.5 * merahg) + (0.3 * hijaug) + (0.2 * birug));
                int grayhasil = Math.round(gray);
                arrayGray[x][y] = grayhasil;
            }
        }
        return arrayGray;
    }

 public int[] histogram(int[][] grayScale, int ukuranX, int ukuranY) {
    int[] pixNum = new int[256];
    int size = grayScale.length;
```

```
        for (int c = 0; c < 256; c++) {
           for (int i = 0; i < ukuranX; i++) {
                for (int j = 0; j < ukuranY; j++) {
                    if (grayScale[i][j] == c) {
                      sum++;
                    }
                    pixNum[c] = sum;
                }
            }

        }
        return pixNum;
    }

    //CDF = Cumulative Distributif Function
    public int[] getCDF(int[] histogram) {
        int[] cdf = new int[256];
        int cum = 0;
        for (int i = 0; i < 256; i++) {
            cum += histogram[i];
            cdf[i] = cum;
        }
        return cdf;
    }

    public int getMinCDF(int[] cdf) {
        int minCDF = 257;
        for (int i = 0; i < 256; i++) {
            if (cdf[i] < minCDF && cdf[i] != 0) {
                minCDF = cdf[i];
            }
        }
        return minCDF;
    }

    public int getMaxCDF(int[] cdf) {
        int maxCDF = 0;
        for (int i = 0; i < 256; i++) {
            if (cdf[i] > maxCDF) {
                maxCDF = cdf[i];
            }
        }
        return maxCDF;
    }

    public float[] equalization(int[] cdf, int pictSize) {
        int min = getMinCDF(cdf);
        float e[] = new float[256];
        System.out.println("minimum: " + min);
        System.out.println("pictSize: " + pictSize);
        for (int i = 0; i < 256; i++) {
            e[i] = (float) ((((float) cdf[i] - min) / (float) pictSize) * 255);
            System.out.println(e[i]);
        }
        for (int i = 0; i < 256; i++) {
            if (e[i] < 0) {
                e[i] = 0;
            }
            if (e[i] > 255) {
                e[i] = 255;
            }
        }
        return e;
```

```
    }

    public float[][] gbEqualiz(int[][] grayScale, float[] equalization, int w, int h) {
        int size = w * h;
        float[][] newGS = new float[w][h];
        int counter = 0;
        for (int i = 0; i < w; i++) {
            for (int j = 0; j < h; j++) {
                newGS[i][j] = equalization[(int) grayScale[i][j]];
                counter++;
            }
        }
        return newGS;
    }

    public BufferedImage gbHasil(float[][] newGS, int w, int h) {
        int size = w * h;
        int counter = 0;
        BufferedImage im = new BufferedImage(w, h, BufferedImage.TYPE_BYTE_GRAY);
        WritableRaster raster = im.getRaster();
        for (int i = 0; i < w; i++) {
            for (int j = 0; j < h; j++) {
                float gray = newGS[i][j];
                raster.setSample(i, j, 0, gray);
                counter++;
            }
        }
        return im;
    }
}
```

9). Jalankan program sehingga tampilannya seperti gambar 6.8. berikut:



Gambar 6.8. Tampilan aplikasi hasil proses Equalisasi Histogram