PRAKTIKUM 7
Deteksi Tepi

7.1. Tujuan :

Mahasiswa mengetahui cara membuat program segmentasi citra dengan metode deteksi tepi menggunakan filter Laplacian.

7.2. Dasar Teori :

7.2.1. Segmentasi

Segmentasi citra merupakan proses yang ditujukan untuk mendapatkan objek-objek yang terkandung di dalam citra atau membagi citra ke dalam beberapa daerah dengan setiap objek atau daerah memiliki kemiripan atribut. Pada citra yang mengandung hanya satu objek, objek dibedakan dari latarbelakangnya.

7.2.2. Deteksi tepi

Deteksi tepi berfungsi untuk memperoleh tepi objek. Deteksi tepi memanfaatkan perubahan nilai intensitas yang drastis pada batas dua area. Definisi tepi di sini adalah “himpunan piksel yang terhubung yang terletak pada batas dua area” (Gonzalez & Woods, 2002). Perlu diketahui, tepi sesungguhnya mengandung informasi yang sangat penting. Informasi yang diperoleh dapat berupa bentuk maupun ukuran objek.

7.2.3. Operator Laplacian

Operator Laplacian merupakan contoh operator yang berdasarkan pada turunan kedua. Operator ini bersifat omnidirectional, yakni menebalkan bagian tepi ke segala arah. Namun, operator Laplacian memiliki kelemahan, yakni peka terhadap derau, memberikan ketebalan ganda, dan tidak mampu mendeteksi arah tepi (Gonzalez & Woods, 2002). Contoh cadar ditunjukkan pada Gambar 7.1.

	x-1	x	x+1
y-1	z_1	z_2	z_3
y	z_4	z_5	z_6
y+1	z_7	z_8	z_9

(a) Posisi pada citra f

0	-1	0
-1	4	-1
0	-1	0

(b) #1

-1	-1	-1
-1	8	-1
-1	-1	-1

(c) #2

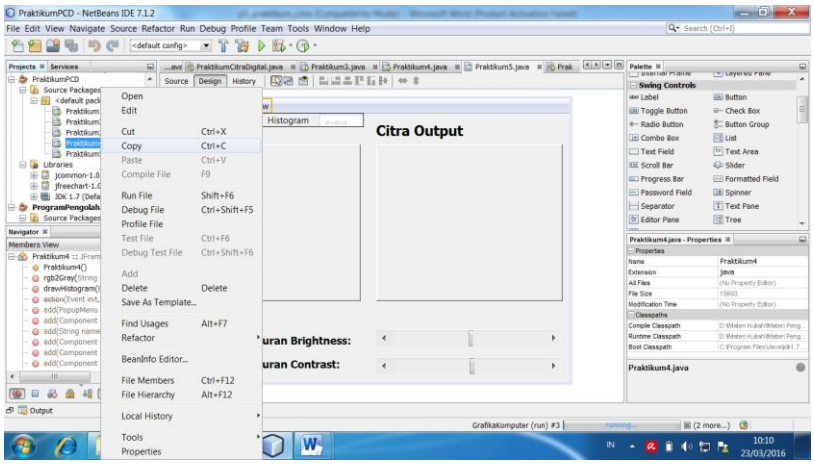
Gambar 7.1 Operator Laplacian

Berdasarkan filter #1 pada Gambar 7.1(b), nilai operator Laplacian pada (y, x) didefinisikan sebagai

$l(y,x)=4\ f(y,x)-[f(y-1,x)+f(y,x-1)+f(y,x+1)+f(y+1,x)]$

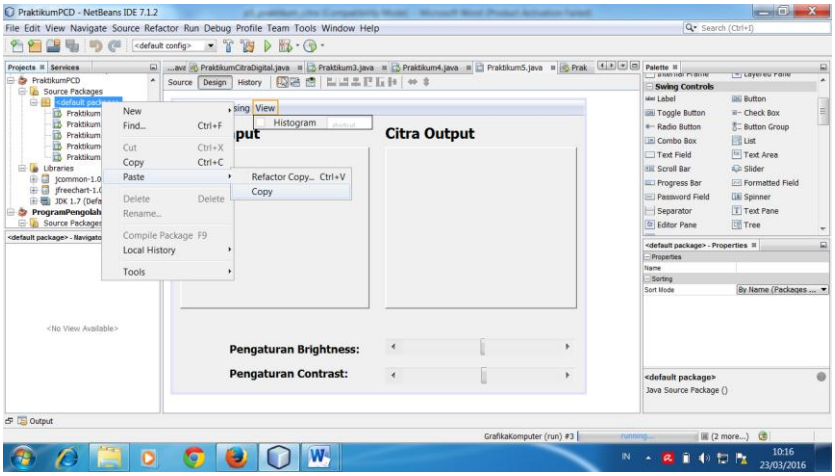
7.2. Langkah Praktikum :

1). Copy class JFrame Form pada praktikum sebelumnya dengan cara klik kanan pada folder Source Package Praktikum6 lalu pilih Copy seperti tampak pada gambar 7.2 berikut:



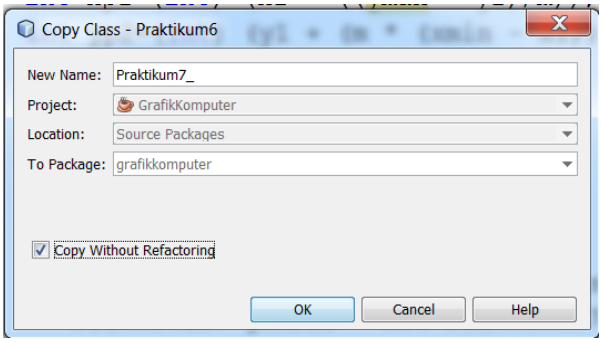
Gambar 7.2. Copy class JFrame Form

2). Kemudian klik kanan pada folder source packages pada <default Packages>, kemudian pilih Paste kemudian pilih Copy, seperti tampak pada gambar 7.3.



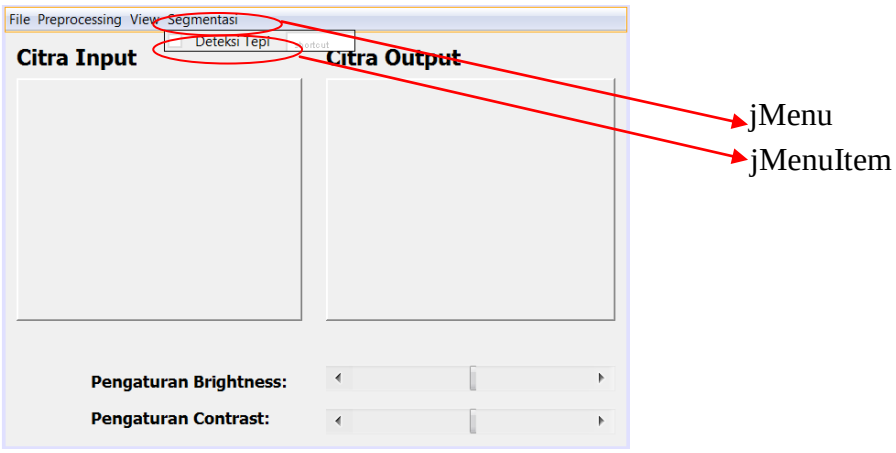
Gambar 7.3. Paste ke class JFrame Form baru

3). Rubah nama class menjadi Praktikum6 kemudian centang pilihan Copy Without Refactoring. Kemudian klik OK. Seperti tampak pada gambar 7.4. kemudian rubah nama class dari Praktikum6_1 menjadi Praktikum6.



Gambar 7.4. Jendela Dialog Copy Class

4). Klik Design, kemudian tambahkan 1 buah komponen jMenu pada jMenuBar dan rubah text nya menjadi Segmentasi kemudian tambahkan jMenuItem baru pada Menu Segmentasi tersebut kemudian rubah text nya menjadi Deteksi tepi. Seperti tampak dalam gambar 7.5. berikut:



Gambar 7.5. Design Form

5). Tambahkan Kode metode untuk proses segmentasi pada bagian Source Praktikum7 pada event method jMenuItem Deteksi tepi sebagai berikut.

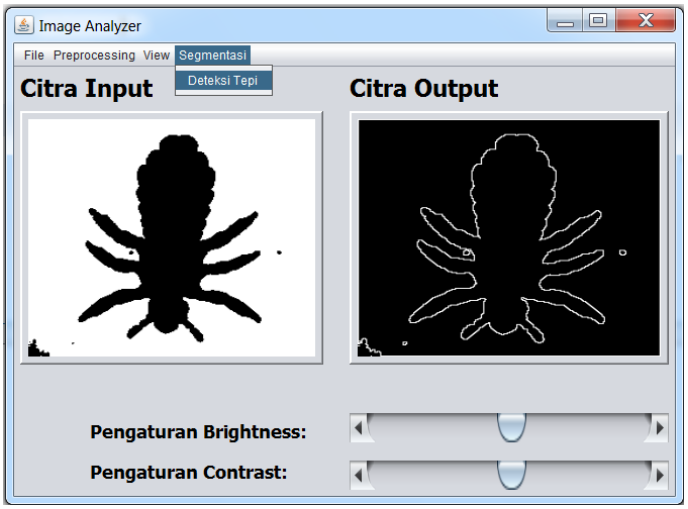
```
private void jMenuItem8ActionPerformed(java.awt.event.ActionEvent evt) {
BufferedImage grayscale = deteksitepi(sumber);
int x = jLabel2.getWidth();
int y = jLabel2.getHeight();
ImageIcon imageIcon = new ImageIcon(resize(grayscale, x, y));
jLabel2.setIcon(imageIcon);
}
```

6). Tambahkan methode baru untuk proses deteksi tepi dengan kode sebagai berikut:

```
public BufferedImage deteksitepi(String sumber) {
    BufferedImage prosesGambar;
    BufferedImage loadIng = loadImage(sumber);
    ukuranX = loadIng.getWidth();
    ukuranY = loadIng.getHeight();
    prosesGambar = new BufferedImage(ukuranX, ukuranY, BufferedImage.TYPE_BYTE_GRAY);
    Graphics g = prosesGambar.getGraphics();
    g.drawImage(loadIng, 0, 0, null);
    WritableRaster raster = prosesGambar.getRaster();
    for (int x = 1; x < (ukuranX-1); x++) {
        for (int y = 1; y < (ukuranY-1); y++) {
            int rgb11 = loadIng.getRGB((x-1),(y-1));
            int p11 = (rgb11 >> 8) & 0xff;
            int rgb12 = loadIng.getRGB(x, (y-1));
            int p12 = (rgb12 >> 8) & 0xff;
            int rgb13 = loadIng.getRGB((x+1),(y-1));
            int p13 = (rgb13 >> 8) & 0xff;
            int rgb21 = loadIng.getRGB((x-1),(y));
            int p21 = (rgb21 >> 8) & 0xff;
            int rgb22 = loadIng.getRGB(x, y);
            int p22 = (rgb22 >> 8) & 0xff;
            int rgb23 = loadIng.getRGB((x+1), y);
            int p23 = (rgb23 >> 8) & 0xff;
            int rgb31 = loadIng.getRGB((x-1),(y+1));
            int p31 = (rgb31 >> 8) & 0xff;
            int rgb32 = loadIng.getRGB(x,(y+1));
            int p32 = (rgb32 >> 8) & 0xff;
            int rgb33 = loadIng.getRGB((x+1),(y+1));
            int p33 = (rgb33 >> 8) & 0xff;
```

```
                int tepi = (int)((p11*-1)+(p12*-1)+(p13*-1)+
                                 (p21*-1)+(p22 *8)+(p23*-1)+
                                 (p31*-1)+(p32*-1)+(p33*-1));
                if (tepi < 0) {
                    tepi = 0;
                }
                if (tepi > 255) {
                    tepi = 255;
                }
                raster.setSample(x, y, 0, tepi);
            }
        }
        return prosesGambar;
    }
```

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



Gambar 7.6. Tampilan aplikasi hasil proses Deteksi tepi