PRAKTIKUM 10
Deskripsi Citra dengan metode Thinning

10.1. Tujuan :

Mahasiswa mengetahui cara membuat program morfologi citra dengan menggunakan operasi Dilasi dan erosi pada sebuah citra biner

10.2. Dasar Teori :

Dalam *computer vision*, kita tahu bahwa thinning (penipisan) yang kita gunakan untuk menghasilkan kerangka (skeleton) suatu objek, dan biasanya objek berupa citra biner (hitam putih). Kerangka dianggap mewakili bentuk objek dalam jumlah yang relatif kecil dari pixel. Tiga hal yang perlu diingat dalam teori thinning ini adalah :

1. Tidak semua objek harus di lakukan thinning (penipisan). Penipisan berguna untuk objek yang terdiri dari garis, baik itu lurus maupun melengkung (curve), dan tidak berguna untuk objek yang memiliki bentuk yang menyertakan daerah yang signifikan. Misalnya, lingkaran dapat dilakukan thinning karena diwakili oleh garis melengkung; sedangkan sebuah disk (piringan) tidak perlu dilakukan thinning karena menyertakan komponen-komponen didalamnya.
2. Apa yang diterapkan sebagai kerangka pada suatu situasi tidak dapat diterapkan di situasi yang berbeda. Thinning pada umumnya suatu langkah awal (preproses) untuk suatu citra yang akan diproses lebih lanjut (contoh, lingkaran yang diproses lebih lanjut menggunakan Hough Transform akan terlebih dahulu di thinning untuk meningkatkan akurasi hasil deteksi).
3. Thinning adalah proses identifikasi kerangka (skeleton), dan tidak ditentukan oleh algoritma yang digunakan

10.2.1. Algoritma Thinning

Sekarang, mari kita bahas salah satu algoritma thinning yaitu Zhang Suen. Algoritma ini merupakan algoritma thinning yang sederhana. Pertama, di bawah ini adalah kondisi yang diberikan oleh algoritma Zhang Suen :

a) 2 <= N(P1) <= 6

b) S(P1) = 1

c) P2 * P4 * P6 = 0

d) P4 * P6 * P8 = 0

e) 2 <= N(P1) <= 6

f) S(P1) = 1

g) P2 * P4 * P8 = 0

h) P2 * P6 * P8 = 0

Konsep: 1. Jangan lepaskan titik terakhir
2. Jangan melanggar konektivitas
3. Jangan menyebabkan erosi berlebihan

Hanya berlaku untuk kontur piksel: pixel "1" memiliki setidaknya satu dari 8 tetangga piksel bernilai "0"

Notation:

p_9	p_2	p_3
p_8	p_1	p_4
p_7	p_6	p_5

$N(p_1) = p_2 + p_3 + \ldots + p_8 + p_9$

T(p1) = jumlah transisi 0-1 dalam urutan piksel tetangga p2, p3, p4, p5, p6 , p7, p8, p9, p2.

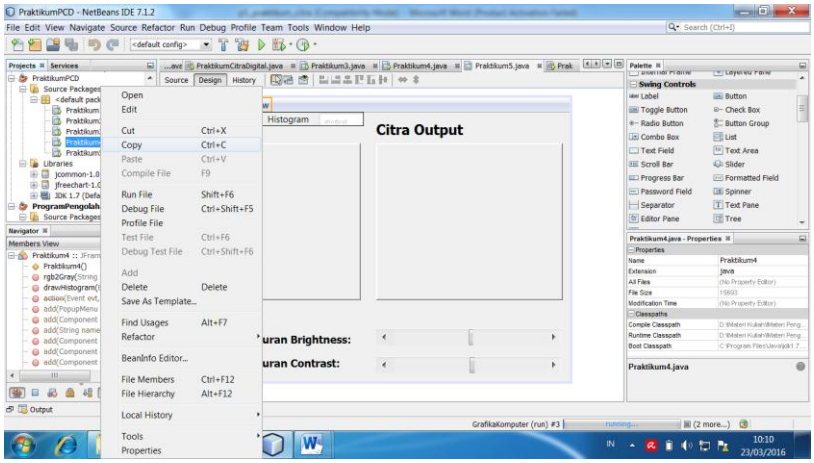
Di atas telah dipaparkan kondisi yang diberikan oleh algoritma Zhang Suen, mari kita mulai untuk memahami algoritma ini. Kita akan membuat sebuah aturan, sehingga aturan ini harus diikuti. Kita akan menggunakan citra biner, dimana pixel putih sebagai foreground dan pixel hitam sebagai background. Dan yang perlu diperhatikan adalah kita menggunakan aturan delapan tetangga (eight neighbor). Ikuti aturan ini dan dibawah ini adalah algoritma thinning, ulangin seluruh langkah ini hingga tidak ada perubahan pixel pada citra :

1. Tandai semua pixel foreground (berwarna putih) yang memenuhi kondisi a – d
2. Ubah semua pixel yang telah ditandai pada langkah nomor 1 menjadi pixel background (warna hitam)
3. Tandai semua pixel foreground (berwarna putih) yang memenuhi kondisi e – h

4. Ubah semua pixel yang telah ditandai pada langkah nomor 3 menjadi pixel background (warna hitam)

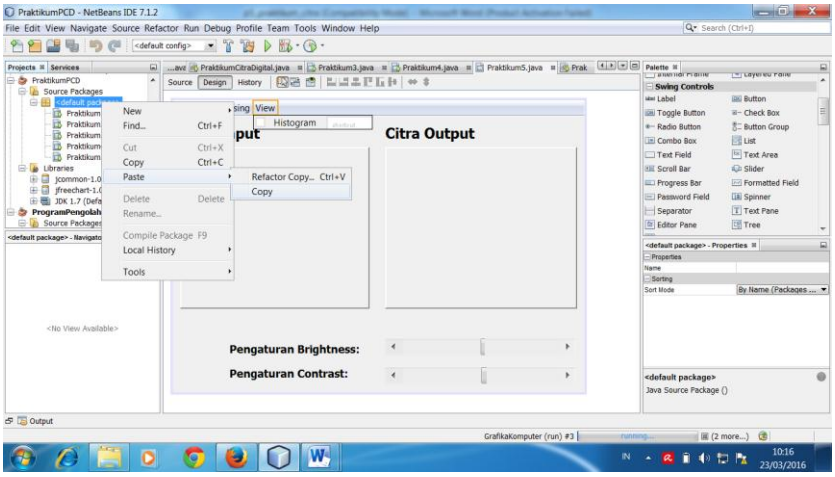
10.3. Langkah Praktikum :

1). Copy class JFrame Form pada praktikum sebelumnya dengan cara klik kanan pada folder Source Package Praktikum10 lalu pilih Copy seperti tampak pada gambar 10.1 berikut:



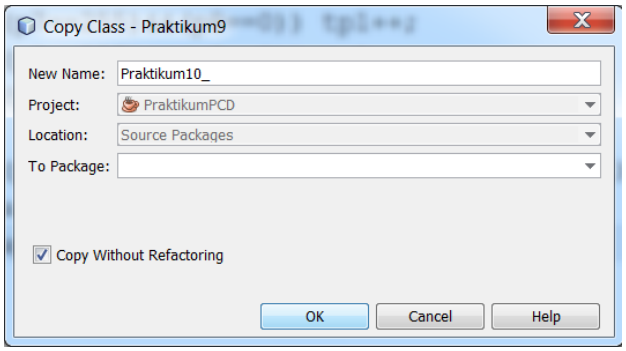
Gambar 10.1. Copy class JFrame Form

2). Kemudian klik kanan pada folder source packages pada <default Packages>, kemudian pilih Paste kemudian pilih Copy, seperti tampak pada gambar 10.2.



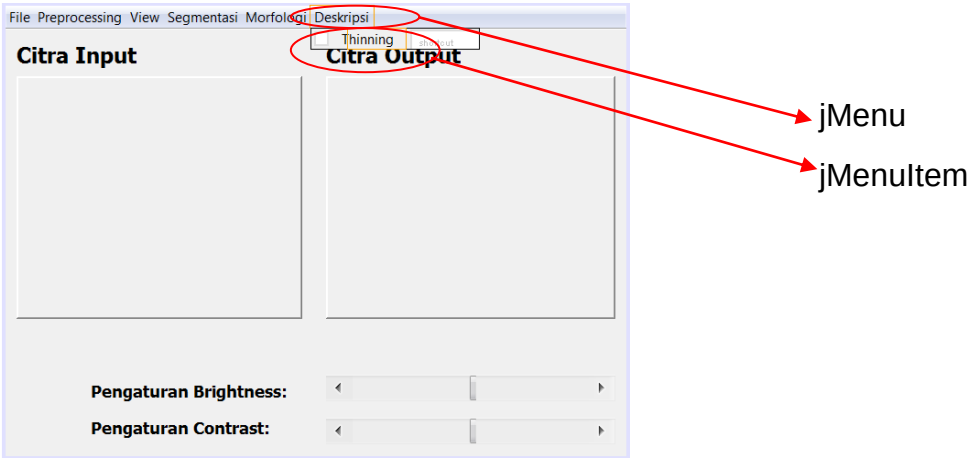
Gambar 10.2. Paste ke class JFrame Form baru

3). Rubah nama class menjadi Praktikum10 kemudian centang pilihan Copy Without Refactoring. Kemudian klik OK. Seperti tampak pada gambar 10.3. kemudian rubah nama class dari Praktikum9_1 menjadi Praktikum10.



Gambar 10.3. Jendela Dialog Copy Class

4). Klik Design, kemudian tambahkan 1 buah komponen jMenu pada jMenuBar dan rubah text nya menjadi Deskripsi kemudian tambahkan sebuah jMenuItem baru pada Menu Deskripsi tersebut kemudian rubah text nya menjadi Thinning Seperti tampak dalam gambar 10.4 berikut:



Gambar 10.3. Design Form

5). Tambahkan Kode metode untuk proses segmentasi pada bagian Source Praktikum10 pada event method jMenuItem thinning sebagai berikut.

```
private void jMenuItem10ActionPerformed(java.awt.event.ActionEvent evt) {
  BufferedImage grayscale = thinning(sumber);
  int x = jLabel2.getWidth();
  int y = jLabel2.getHeight();
  ImageIcon imageIcon = new ImageIcon(resize(grayscale, x, y));
  jLabel2.setIcon(imageIcon);
}
```

7). Tambahkan methode baru untuk proses thinning dengan kode sebagai berikut:

```
public BufferedImage thinning(String sumber) {
  BufferedImage prosesGambar;
  BufferedImage loadIng = loadImage(sumber);
  ukuranX = loadIng.getWidth();
  ukuranY = loadIng.getHeight();
  int Bufferku [][]= new int [ukuranX][ukuranY];
  prosesGambar = new BufferedImage(ukuranX, ukuranY,
                 BufferedImage.TYPE_BYTE_GRAY);
  Graphics g = prosesGambar.getGraphics();
  g.drawImage(loadIng, 0, 0, null);
  WritableRaster raster = prosesGambar.getRaster();
  double hapus=0;
  for (int x = 0; x < ukuranX; x++) {
    for (int y = 0; y < ukuranY; y++) {
       int rgb22 = loadIng.getRGB(x, y);
       int pixel = (rgb22 >> 8) & 0xff;
       Bufferku[x][y]=pixel;
    }
  }
  do {
  // Step 1 & 2
       hapus=0;
       for (int x = 1; x < (ukuranX-1); x++) {
         for (int y = 1; y < (ukuranY-1); y++) {
             int p9 = Bufferku[x-1][y-1];
             int p2 = Bufferku[x][y-1];
             int p3 = Bufferku[x+1][y-1];
             int p8 = Bufferku[x-1][y];
             int p1 = Bufferku[x][y];
             int p4 = Bufferku[x+1][y];
             int p7 = Bufferku[x-1][y+1];
             int p6 = Bufferku[x][y+1];
             int p5 = Bufferku[x+1][y+1];

             int np1=0;
             if (p2 == 0) np1++;
             if (p3 == 0) np1++;
             if (p4 == 0) np1++;
             if (p5 == 0) np1++;
             if (p6 == 0) np1++;
             if (p7 == 0) np1++;
             if (p8 == 0) np1++;
             if (p9 == 0) np1++;

             int tp1=0;
             if ((p2==255)&&(p3==0)) tp1++;
             if ((p3==255)&&(p4==0)) tp1++;
             if ((p4==255)&&(p5==0)) tp1++;
             if ((p5==255)&&(p6==0)) tp1++;
             if ((p6==255)&&(p7==0)) tp1++;
             if ((p7==255)&&(p8==0)) tp1++;
             if ((p8==255)&&(p9==0)) tp1++;
             if ((p9==255)&&(p2==0)) tp1++;
```

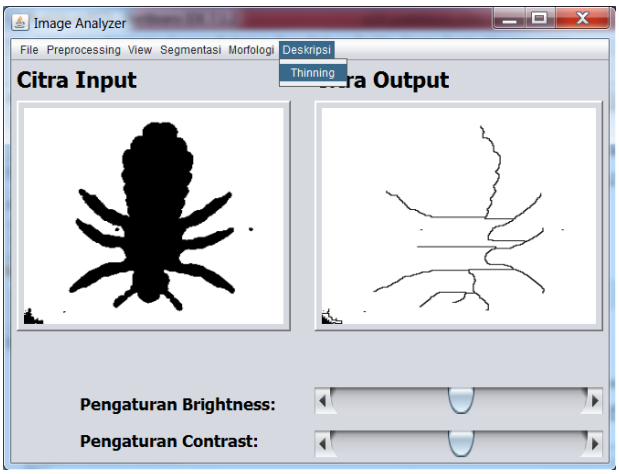
```
                if
    ((p1==0)&&((np1>=2)&&(np1<=6))&&(tp1==1)&&((p2*p4*p6)==0)&&((p4*p6*p8)==0)) {
                    Bufferku[x][y]=255;
                    hapus++;
                }
             }
          }
          hapus=0;
          // Step 3 & 4
          for (int x = 1; x < (ukuranX-1); x++) {
            for (int y = 1; y < (ukuranY-1); y++) {
                int p9 = Bufferku[x-1][y-1];
                int p2 = Bufferku[x][y-1];
                int p3 = Bufferku[x+1][y-1];
                int p8 = Bufferku[x-1][y];
                int p1 = Bufferku[x][y];
                int p4 = Bufferku[x+1][y];
                int p7 = Bufferku[x-1][y+1];
                int p6 = Bufferku[x][y+1];
                int p5 = Bufferku[x+1][y+1];

                int np1=0;
                if (p2 == 0) np1++;
                if (p3 == 0) np1++;
                if (p4 == 0) np1++;
                if (p5 == 0) np1++;
                if (p6 == 0) np1++;
                if (p7 == 0) np1++;
                if (p8 == 0) np1++;
                if (p9 == 0) np1++;

                int tp1=0;
                if ((p2==255)&&(p3==0)) tp1++;
                if ((p3==255)&&(p4==0)) tp1++;
                if ((p4==255)&&(p5==0)) tp1++;
                if ((p5==255)&&(p6==0)) tp1++;
                if ((p6==255)&&(p7==0)) tp1++;
                if ((p7==255)&&(p8==0)) tp1++;
                if ((p8==255)&&(p9==0)) tp1++;
                if ((p9==255)&&(p2==0)) tp1++;

                if
    ((p1==0)&&((np1>=2)&&(np1<=6))&&(tp1==1)&&((p2*p4*p8)==0)&&((p2*p6*p8)==0)) {
                     Bufferku[x][y]=255;
                     hapus++;
                }
             }
          }
      }  while (hapus==0);
      for (int x = 0; x < ukuranX; x++) {
        for (int y = 0; y < ukuranY; y++) {
           raster.setSample(x, y, 0, Bufferku[x][y]);
        }
      }
      return prosesGambar;
  }
```

8). Jalankan program sehingga tampilannya seperti gambar 10.4. berikut:



Gambar 10.4. Tampilan aplikasi hasil proses deskripsi dengan metode thinning