**BAB VII**

**Deteksi Tepi Citra**

1. **Hasil Pembelajaran Umum:**

* Memiliki pengertian dasar tentang konsep fundamental citra, model citra, representasi dan operasi citra secara praktis.

1. **Hasil Pembelajaran Khusus:**

* Mampu memahami dan menjelaskan prinsip dan metode deteksi tepi pada citra
* Mampu memahami dan menerapkan pemrograman Deteksi tepi pada citra.
  1. **Deteksi Tepi**

Pada beberapa kasus penerapan pengolahan citra dibutuhkan garis objek pada citra. Garis objek pada citra mempunyai cirri – ciri nilai intensitas pixel yang berbeda sacara signifikan dengan nilai intensitas pixel tetangga. Proses deteksi tepi adalah proses mendapatkan garis tepi objek pada suatu citra. Untuk mendapatkan tepi objek, kita harus memfilter citra menggunakan High Pass Filter.

Contoh:

Suatu citra F(x,y) mempunyai data sebagai berikut:

F(x,y) =

Difilter menggunakan High Pass Filter dengan matrik filter sebagai berikut:

H =

Z(x,y) = F ⊗ H

Z(1,1) = f(1,1).h(1,1) + f(1,2).h(1,2)

= 1 + -1

= 0

Z(1,2) = f(1,2).h(1,1) + f(1,3).h(1,2)

= 1 + -1

= 0

Z(1,3) = f(1,3).h(1,1) + f(1,4).h(1,2)

= 1 + -1

= 0

Z(1,4) = f(1,4).h(1,1) + f(1,5).h(1,2)

= 1 + -1

= 0

Z(1,5) = f(1,5).h(1,1) + f(1,6).h(1,2)

= 1 + 0

= 1

Z(2,1) = f(2,1).h(1,1) + f(2,1).h(1,2)

= 1 + -1

= 0

Z(2,2) = f(2,2).h(1,1) + f(2,3).h(1,2)

= 1 + -1

= 0

Z(2,3) = f(2,3).h(1,1) + f(2,4).h(1,2)

= 1 + -1

= 0

Z(2,4) = f(2,4).h(1,1) + f(2,5).h(1,2)

= 1 + 0

= 1

Z(2,5) = f(2,5).h(1,1) + f(2,6).h(1,2)

= 0 + 0

= 0

Z(3,1) = f(3,1).h(1,1) + f(3,2).h(1,2)

= 1 + -1

= 0

Z(3,2) = f(3,2).h(1,1) + f(3,3).h(1,2)

= 1 + -1

= 0

Z(3,3) = f(3,3).h(1,1) + f(3,3).h(1,2)

= 1 + 0

= 1

Z(3,4) = f(3,4).h(1,1) + f(3,5).h(1,2)

= 0 + 0

= 0

Z(3,5) = f(3,5).h(1,1) + f(3,6).h(1,2)

= 0 + 0

= 0

Z(4, 1) = f(4,1).h(1,1) + f(4,2).h(1,2)

= 1 + -1

= 0

Z(4, 2) = f(4,2).h(1,1) + f(4,3).h(1,2)

= 1 + 0

= 1

Z(4, 3) = f(4,3).h(1,1) + f(4,4).h(1,2)

= 0 + 0

= 0

Z(4, 4) = f(4,4).h(1,1) + f(4,5).h(1,2)

= 0 + 0

= 0

Z(4, 5) = f(4,5).h(1,1) + f(4,6).h(1,2)

= 0 + 0

= 0

Z(5, 1) = f(5,1).h(1,1) + f(5,2).h(1,2)

= 1 + 0

= 1

Z(5,2) = f(5,2).h(1,1) + f(5,2).h(1,2)

= 0 + 0

= 0

Z(5,3) = f(5,3).h(1,1) + f(5,4).h(1,2)

= 0 + 0

= 0

Z(5,4) = f(5,4).h(1,1) + f(5,5).h(1,2)

= 0 + 0

= 0

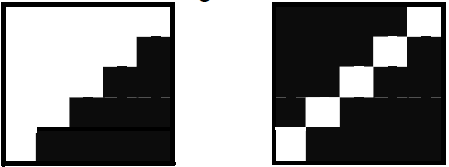
Z(5,5) = f(5,5).h(1,1) + f(5,6).h(1,2)

= 0 + 0

= 0

Z(x,y) =

Dari fungsi citra awal dan fungsi citra hasil dapat direpresenrasikan citra seperti pada Gambar 7. 1



Gambar 7. 1 Representasi Hasil Konvolusi

Terdapat banyak metode yang dapat digunakan untuk memperoleh tepi objek pada citra (deteksi tepi), antara lain :

1. Metode Robert
2. Metode Prewitt
3. Metode Sobel
4. Metode Laplacian
5. Metode Kirsch
   1. **Metode Robert**

Metode Robert merupakan metode deteksi tepi dengan teknik deferensial arah horizontal dan deferensial arah vertical. Metode ini menggunakan matrik filter ukuran 2 x 1 atau 1 x 2.

H = H =

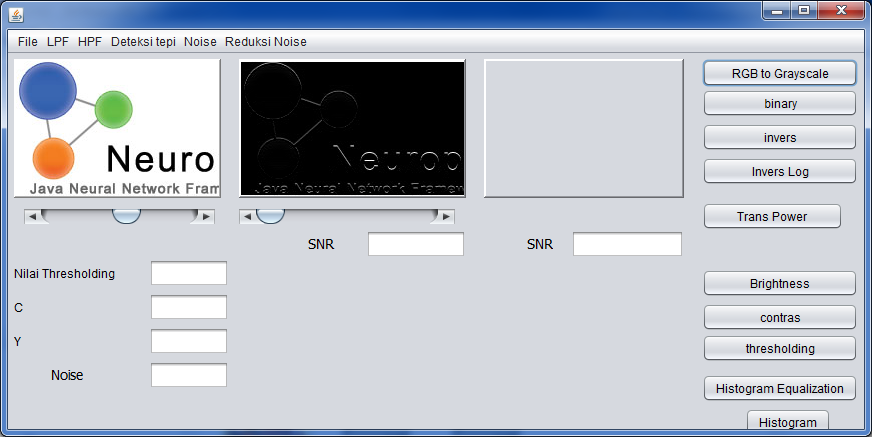
Kode program berikut ini merupakan potongan kode program implemetasi deteksi tepi dengan metode Robert. Hasil eksekusi kode program seperti pada Gambar 7. 2.

float[] edgeKernel = {-1,1};

BufferedImageOp edgeop = new ConvolveOp(new Kernel(1,2, edgeKernel), ConvolveOp.EDGE\_NO\_OP, null);

BufferedImage image3 = edgeop.filter(bi, null);

jLabel2.setIcon(new ImageIcon(image3));



Gambar 7. 2. Hasil Deteksi Tepi Menggunakan Metode Robert

* 1. **Metode Prewitt**

Metode Prewitt merupakan pengembangan dari metode Robert. Metode ini menggunakan matrik filter 3 x 3. Metode ini menghasilkan garis tepi objek yang lebh tebal dan tajam serta memberikan efek timbul bila dibandingkan dengan dengan hasil menggunakan metode Robert.

H = H =

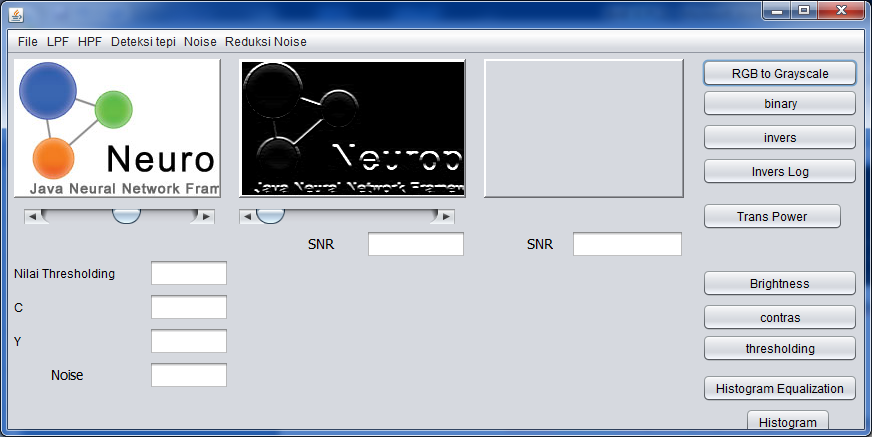
Kode program berikut ini adala kode program implementasi operasi deteksi tepi menggunakan metode prewitt. Gambar 7. 3 menujukan hasil operasi deteksi tepi menggunakan metode ini.

float[] edgeKernel = {-1,0,1,-1,0,1,-1,0,1};

BufferedImageOp edgeop = new ConvolveOp(new Kernel(3, 3, edgeKernel),ConvolveOp.EDGE\_NO\_OP, null);

BufferedImage image3 = edgeop.filter(bi, null);

jLabel2.setIcon(new ImageIcon(image3));



Gambar 7. 3 Hasil Operasi Deteksi tepi dengan metode Prewitt

* 1. **Metode Sobel**

Metode ini menggunakan prinsip fungsi Laplacian dan fungsi Gaussian sehingga metode ini mempunyai kelebihan bila dibandingkan dengan metode Robert dan prewitt yaitu ada tahapan pengurangan noise sebelum operasi deteksi tepi. Metode sobel menggunakan kernel filter ukuran 3 x 3 dengan elemen sebagai berikut:

H = H =

Hasil operasi deteksi tepi menggunakan metode sobel seperti terlihat pada Gambar 7. 4. Kode program untuk operasi deteksi tepi dengan metode sobel adalah sebagai berikut:

float[] edgeKernel = {-1,-2,-1,0,0,0,1,2,1};

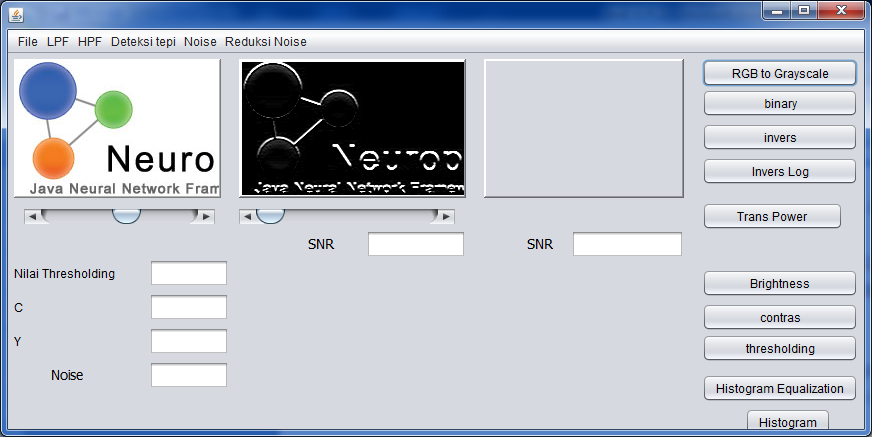
BufferedImageOp edgeop = new ConvolveOp(new Kernel(3, 3, edgeKernel), ConvolveOp.EDGE\_NO\_OP, null);

BufferedImage image3 = edgeop.filter(bi, null);

jLabel2.setIcon(new ImageIcon(image3));

} catch (Exception e) {

System.out.println(e);



Gambar 7. 5. Hasil Deteksi tepi menggunakan metode Sobel

* 1. **Metode Laplacian**

Metode deteksi tepi Laplacian menggunakan matrik filter dengan ukuran 3 x 3 dengan tiga varian elemen. Matrik filter untuk deteksi tepi metode Laplacian adalah sebagai berikut:

H4 = H8 = H12 =

Index pada matrik filter (4,8,12) menunjukan elemen pusat H(2,2). Kode program untuk penerapan operasi deteksi tepi menggunakan metode laplacian, dengan memanfaatkan method convolOp dari Java 2d adalah sebagai berikut :

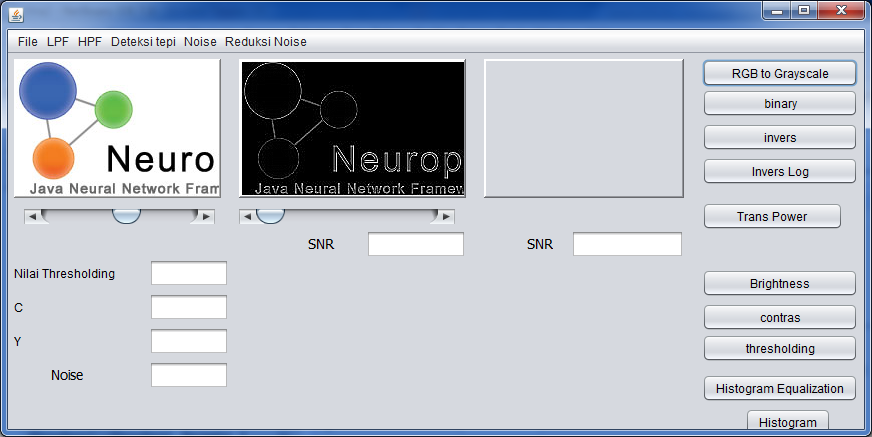
float[] edgeKernel = {0,1,0,1,-4,1,0,1,0};

BufferedImageOp edgeop = new ConvolveOp(new Kernel(3, 3, edgeKernel),ConvolveOp.EDGE\_NO\_OP, null);

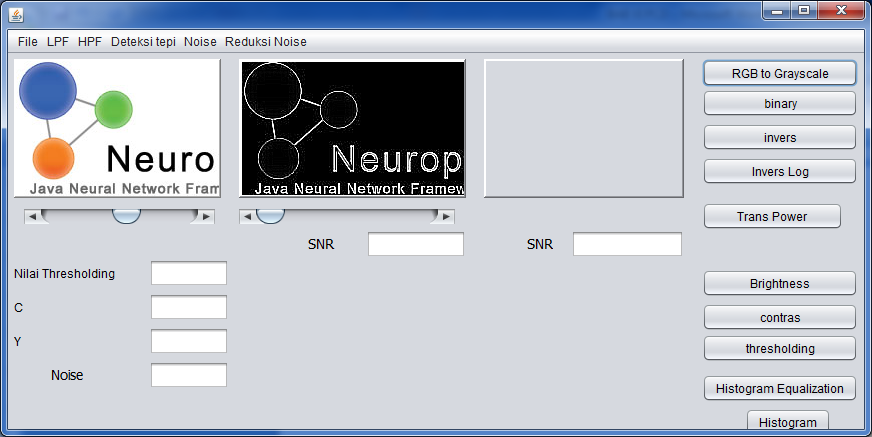
BufferedImage image3 = edgeop.filter(bi, null);

jLabel2.setIcon(new ImageIcon(image3));

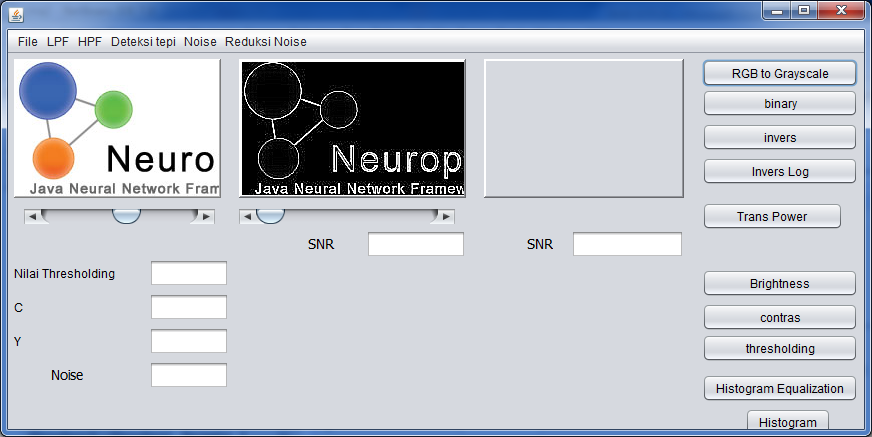
Dengan menyesuaikan elemen pada matrik filter maka kode program tersebut dapat digunakan untuk matrik filter H8 dan H12. Hasil eksekusi program untuk operasi deteksi tepi menggunakan metode Laplacian berturut – turut seperti terlihat pada Gambar 7.6.



(a)



(b)



(c)

Gambar 7. 6 Hasil Operasi Deteksi tepi Laplacian (a) H4,(b)H8, (c)H12

Dari hasil operasi deteksi tepi dengan metode Laplacian terlihat citra hasil H12, lebih tajam bila dibandingkan dengan hasil H8, begitu pula dengan hasil H8 lebih tajam H12.

* 1. **Metode Kirsch**

Deteksi tepi dengan metode Kirsch merupakan pengembangan dari metode sobel. Meotde ini menggunakan matrik filter dengan ukuran 3 x 3. Matrik filter yang digunakan bervariasi antara lain sebagai berikut:

H0 = H1 =

H2 = H3 =

Kode program berikut merupakan kode program implementasi dari deteksi tepi menggunakan metode Kirsch. Hasil eksekusi dari metode ini seperti pada Gambar 7. 7. Pada gambar tersebut terlihat tepi objek pada citra lebih tebal hampir sama dengan hasil dari metode sobel.

float[] edgeKernel = {3,-5,-5,3,0,-5,3,3,3};

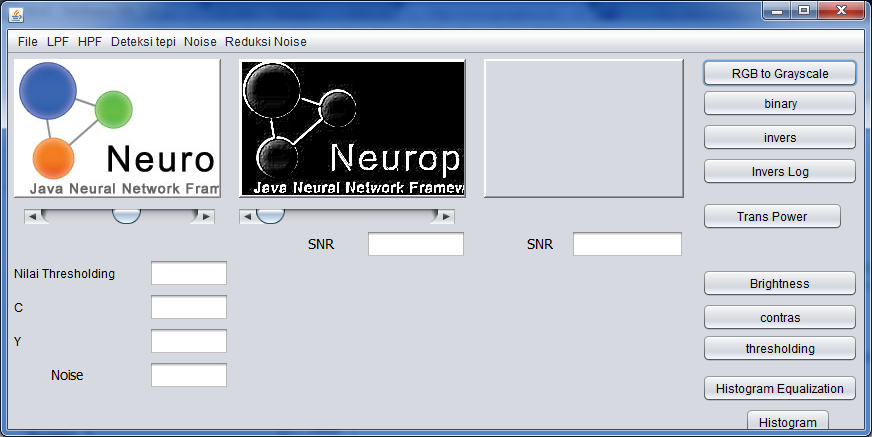
BufferedImageOp edgeop = new ConvolveOp(new Kernel(3, 3, edgeKernel), ConvolveOp.EDGE\_NO\_OP, null);

BufferedImage image3 = edgeop.filter(bi, null);

jLabel2.setIcon(new ImageIcon(image3));



(a)



(b)



(c)



(d)

Gambar 7. 6 Hasil Operasi Deteksi tepi Kirsch (a) H0,(b)H1, (c)H2, (d)H3

* 1. **Rangkuman**

1. Operasi deteksi tepi dilakukan dengan tujuan mendapatkan garis tepian obyek pada citra.
2. Operasi deteksi tepi dapat dilakukan dengan filtering citra menggunakan low pass filter.
3. Pada Java 2D telah disediakan method untuk operasi filtering sehingga didapatkan hasil yang optimal dengan kode program yang lebih singkat.
   1. **Soal Latihan**
4. Sebutkan macam – macam metode pada operasi deteksi tepi.
5. Jelaskan proses filtering untuk mendapatkan garis tepian obyek pada citra.
6. Jelaskan ciri – ciri kernel filter untuk operasi deteksi tepi
7. Sebutkan metode deteksi tepi yang menghasilkan tepi obyek yang paling tipis.
8. Buatlah program untuk deteksi tepi metode laplacian denga kernel filter H8 dan H12