Aplikasi Area Process Berbasis C# menggunakan Visual Studio

Yudi Ahmad Hambali
yudihambali@yahoo.com

Pendahuluan

Menurut Setiawan Hadi **[1]**, Pengolahan Citra Digital adalah representasi citra digital dari fungsi kontinu menjadi nilai-nilai diskrit yang difokuskan kepada pengolahan gambar-gambar, ditujukan untuk meningkatkan kualitas citra untuk keperluan persepsi visual manusia maupun interprestasi oleh komputer.
Dalam artikel ini, penulis akan menjelaskan pengolahan citra digital yaitu area process dengan menggunakan bahasa pemrograman C# menggunakan aplikasi Visual Studio.

Area Process

Area Process yaitu algoritma yang digunakan untuk memanipulasi citra pada pixel utama dengan melibatkan pixel tetangga. Yang termasuk pada area process yaitu:

1. Konvolusi, yang terdiri dari : Embossing, Blurring, Sharpening, Edge Detection.
2. Filtering, yang terdiri dari : Median, Average, Maximum, Minimum.

Konvolusi (convolution) adalah sebuah proses dimana citra dimanipulasi dengan menggunakan eksternal mask / subwindows untuk menghasilkan citra yang baru. Sedangkan Filtering tanpa menggunakan ekternal mask tetapi hanya menggunakan pixel tetangga untuk mendapatkan pixel yang baru.
Rumus yang digunakan untuk melakukan proses konvolusi adalah:

P0	P1	P2
P3	P4	P5
P6	P7	P8

M0	M1	M2
M3	M4	M5
M6	M7	M8

Ket: P=Pixel, M=Externel Mask

Pixel baru diperoleh dari ((P0 x M0)+ (P1 x M1)+ (P2 x M2)+ (P3 x M3)+ (P4 x M4)+ (P5 x M5)+ (P6 x M6)+ (P7 x M7)+ (P8 x M8) / (M0+M1+M2+M3+M4+M5+M6+M7+M8)
Jika jumlah dari external mask sama dengan 0 maka diganti menjadi 127.

Konvolusi sangat banyak dipergunakan dalam pengolahan citra untuk memperhalus (smoothing), menajamkan (crispening), mendeteksi tepi (edge detection), serta efek lainnya.

1. **Embossing**
   Embossing menurut Agus Arif **[2]** yaitu membuat citra seolah diukir pada permukaaan selembar nikel. Koefisien jendela konvolusi memiliki bobot tengah bernilai 0 & jumlah seluruh bobot = 0.
   Matriks konvolusi yang digunakan:

-1	0	0
0	0	0
0	0	-1

   Setelah konvolusi, seluruh pixel disesuaikan dengan menambahkan 128.
   Berikut ini penggalan dari algoritma embossing:
   (untuk lebih lengkapnya dapat dilihat pada aplikasinya)

```
private void embossingToolStripMenuItem_Click(object sender, EventArgs e)
{
    int i, j;
    if (img != null)
    {
        MessageBox.Show("Matrik = -1, 0, -1, 0, 4, 0, -1, 0, -1");
        img2 = new Bitmap(img);
        W[0] = -1;
        W[1] = 0;
        W[2] = -1;
        W[3] = 0;
        W[4] = 4;
        W[5] = 0;
        W[6] = -1;
        W[7] = 0;
        W[8] = -1;
        d_nel = W[0] + W[1] + W[2] + W[3] + W[4] + W[5] + W[6] + W[7] + W[8];
        if (d_nel == 0)
        {
            d_nel = 1;
        }
        of_seat = 127;
        for (i = 1; i <= img.Width - 2; i++)
        {
            for (j = 1; j <= img.Height - 2; j++)
            {
                Color pixelcolor = img.GetPixel(i, j);

                pixelcolor = img.GetPixel(i - 1, j - 1);
                dot_gamM[0] = pixelcolor.R;
                dot_gamH[0] = pixelcolor.G;
```

Berikut ini hasil operasi pengolahan citra Embossing:

Gambar 1. Citra Awal

Gambar 2. Citra Hasil Embossing

2. Blurring

Blurring (Pengaburan) yaitu filter spasial low-pass yang melenyapkan detil halus dari suatu citra. Pengaburan dicapai melalui konvolusi dari seluruh koefisien mask bernilai sama. Blurring ini perataan nilai pixel-pixel tetangga, makin besar ukuran mask maka makin besar efek pengaburan.

Matriks konvolusi yang digunakan:

0.0625	0.125	0.0625
0.125	0.25	0.125
0.625	0.125	0.625

Berikut ini penggalan adalah algoritma Blurring:

Algoritma ini hampir sama dengan embossing hanya berbeda pada input mask konvolusi dan seleksi kondisinya.

```
                    if (jimer < 0)
                    {
                        jimer = 0;
                    }
                    if (jihij < 0)
                    {
                        jihij = 0;
                    }
                    if (jibir < 0)
                    {
                        jibir = 0;
                    }
                    if (jimer > 255)
                    {
                        jimer = 255;
                    }
                    if (jihij > 255)
                    {
                        jihij = 255;
                    }
                    if (jibir > 255)
                    {
                        jibir = 255;
                    }
                    Color newpixelcolor = Color.FromArgb(jimer, jihij, jibir);
                    img2.SetPixel(i, j, newpixelcolor);
                }
            }
            pictureBox2.Image = img2;
        }
    }
```

Hasil operasi pengolahan citra Blurring:

Gambar 3. Citra Awal

Gambar 4. Citra Hasil Blurring

3. Sharpening

Sharpening (Penajaman) yaitu memperjelas detil suatu citra(menambah kontras) dengan penjumlahan atas citra tepi dengan citra aslinya maka bagian tepi objek akan terlihat berbeda dengan latarnya, sehingga citra terkesan lebih tajam.

Matriks konvolusi yang digunakan:

0	-1	0
-1	5	-1
0	-1	0

Berikut ini penggalan Algoritma Sharpening:

Algoritma ini hampir sama dengan embossing hanya berbeda pada input mask konvolusi dan seleksi kondisinya.

```
                if (jimer > 255)
                {
                    jimer = 255;
                }
                if (jimer < 0)
                {
                    jimer = 0;
                }
                if (jihij > 255)
                {
                    jihij = 255;
                }
                if (jihij < 0)
                {
                    jihij = 0;
                }
                if (jibir > 255)
                {
                    jibir = 255;
                }
                if (jibir < 0)
                {
                    jibir = 0;
                }

                Color newpixelcolor = Color.FromArgb(jimer, jihij, jibir);
                img2.SetPixel(i, j, newpixelcolor);
            }
        }
        pictureBox2.Image = img2;
    }
}
```

Hasil operasi pengolahan citra Sharpening:

Gambar 5. Citra Asal

Gambar 6. Citra Hasil Sharpening

4. Edge Detection

Deteksi tepi yaitu proses menentukan lokasi titik-titik yang merupakan tepi objek.
Matriks yang digunakan:

Robert

0	0	-1
0	1	0
0	0	0

Prewitt

1	0	-1
1	0	-2
1	0	-1

Sobel

1	0	-1
2	0	-2
1	0	-1

Prei-Chan

1	0	-1
√2	0	-√2
1	0	-1

Laplacian

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

Berikut ini penggalan algoritma Edge Detection:
Algoritma ini hampir sama dengan embossing hanya berbeda pada input mask konvolusi dan seleksi kondisinya.

```
                if (jimer > 255)
                {
                    jimer = 255;
                }
                if (jimer < 0)
                {
                    jimer = 0;
                }
                if (jihij > 255)
                {
                    jihij = 255;
                }
                if (jihij < 0)
                {
                    jihij = 0;
                }
                if (jibir > 255)
                {
                    jibir = 255;
                }
                if (jibir < 0)
                {
                    jibir = 0;
                }

                Color newpixelcolor = Color.FromArgb(jimer, jihij, jibir);
                img2.SetPixel(i, j, newpixelcolor);
            }
        }
        pictureBox2.Image = img2;
    }
}
```

Hasil operasi pengolahan citra Edge Detection:

Gambar 7. Citra Awal

Gambar 8. Citra Hasil Edge Detection

Filtering

Menurut R. Fisher, S. Perkins, A.Walker and E. Wolfart, Mean Filtering **[3]** itu sederhana, intuitif dan mudah untuk menerapkan metode gambar smoothing, yaitu mengurangi jumlah variasi intensitas antara satu pixel dan berikutnya. Hal ini sering digunakan untuk mengurangi noise pada gambar.

1. **Median**

1	3	5
5	7	4
3	8	9

Urutkan nilai pixel : 1,3,…,8,9 , maka diperoleh nilai pixel baru = 5.

Berikut ini penggalan algoritma Median:
(untuk lebih lengkapnya dapat dilihat pada aplikasinya).

```
private void medianToolStripMenuItem_Click(object sender, EventArgs e)
{
    int i, j;
    if (img != null)
    {
        img2 = new Bitmap(img);
        for (i = 1; i <= img.Width - 2; i++)
        {
            for (j = 1; j <= img.Height - 2; j++)
            {
                Color pixelcolor = img.GetPixel(i, j);

                pixelcolor = img.GetPixel(i - 1, j - 1);
                dot_gamM[0] = pixelcolor.R;
                dot_gamH[0] = pixelcolor.G;
                dot_gamB[0] = pixelcolor.B;

                pixelcolor = img.GetPixel(i, j - 1);
                dot_gamM[1] = pixelcolor.R;
                dot_gamH[1] = pixelcolor.G;
                dot_gamB[1] = pixelcolor.B;

                pixelcolor = img.GetPixel(i + 1, j - 1);
                dot_gamM[2] = pixelcolor.R;
                dot_gamH[2] = pixelcolor.G;
                dot_gamB[2] = pixelcolor.B;

                pixelcolor = img.GetPixel(i - 1, j);
                dot_gamM[3] = pixelcolor.R;
                dot_gamH[3] = pixelcolor.G;
                dot_gamB[3] = pixelcolor.B;
```

Hasil operasi pengolahan citra Median:

Gambar 9. Citra Awal

Gambar 10. Citra Hasil Median

2. **Average**

1	3	5
5	7	4
3	8	9

Pixel baru dapat diperoleh dari penjumlahan pixel lalu dibagi oleh jumlah pixel tsb. Nilai pixel baru adalah (1+3+5+5+7+4+3+8+9)/8 = 5.

Berikut penggalan algortima Average:

```
                pixelcolor = img.GetPixel(i - 1, j + 1);
                dot_gamM[6] = pixelcolor.R;
                dot_gamH[6] = pixelcolor.G;
                dot_gamB[6] = pixelcolor.B;

                pixelcolor = img.GetPixel(i, j + 1);
                dot_gamM[7] = pixelcolor.R;
                dot_gamH[7] = pixelcolor.G;
                dot_gamB[7] = pixelcolor.B;

                pixelcolor = img.GetPixel(i + 1, j + 1);
                dot_gamM[8] = pixelcolor.R;
                dot_gamH[8] = pixelcolor.G;
                dot_gamB[8] = pixelcolor.B;

                jimer = Convert.ToInt32((dot_gamM[0] + dot_gamM[1] + dot_gamM[2] +
                dot_gamM[3] + dot_gamM[4] + dot_gamM[5] + dot_gamM[6] + dot_gamM[7] +
                dot_gamM[8]) / 9);
                jihij = Convert.ToInt32((dot_gamH[0] + dot_gamH[1] + dot_gamH[2] +
                dot_gamH[3] + dot_gamH[4] + dot_gamH[5] + dot_gamH[6] + dot_gamH[7] +
                dot_gamH[8]) / 9);
                jibir = Convert.ToInt32((dot_gamB[0] + dot_gamB[1] + dot_gamB[2] +
                dot_gamB[3] + dot_gamB[4] + dot_gamB[5] + dot_gamB[6] + dot_gamB[7] +
                dot_gamB[8]) / 9);

                Color newPixelColor = Color.FromArgb(jimer, jihij, jibir);
                img2.SetPixel(i, j, newPixelColor);
            }
        }
        pictureBox2.Image = img2;
    }
}
```

Hasil operasi pengolahan citra Average:

Gambar 11. Citra Awal

Gambar 12. Citra Hasil Average

3. **Maximum**

1	3	5
5	7	4
3	8	9

Pixel dapat diperoleh dari nilai tertinggi untuk maximum filter yaitu 9.

Berikut penggalan algoritma Maximum:
Algoritma ini hampir sama dengan median hanya berbeda pada seleksi kondisinya.

```
for (int x = 0; x <= 8; x++)
{
    for (int y = x; y <= 8; y++)
    {
        if (dot_gamM[y] > dot_gamM[x])
        {
            temp = dot_gamM[x];
            dot_gamM[x] = dot_gamM[y];
            dot_gamM[y] = dot_gamM[x];
        }
        if (dot_gamH[y] > dot_gamH[x])
        {
            temp = dot_gamH[x];
            dot_gamH[x] = dot_gamH[y];
            dot_gamH[y] = dot_gamH[x];
        }
        if (dot_gamB[y] > dot_gamB[x])
        {
            temp = dot_gamB[x];
            dot_gamB[x] = dot_gamB[y];
            dot_gamB[y] = dot_gamB[x];
        }
    }
}

jimer = dot_gamM[8];

jihij = dot_gamH[8];
jibir = dot_gamB[8];

Color newPixelColor = Color.FromArgb(jimer, jihij, jibir);
img2.SetPixel(i, j, newPixelColor);
```

Hasil operasi pengolahan citra Maximum:

Gambar 13. Citra Awal

Gambar 14. Citra Hasil Maximum

4. **Minimum**

1	3	5
5	7	4
3	8	9

Pixel dapat diperoleh dari nilai terendah untuk minimum filter yaitu 1.

Berikut penggalan algoritma Minimum:
Algoritma ini hampir sama dengan median hanya berbeda pada seleksi kondisinya.

```
for (int x = 0; x <= 8; x++)
{
    for (int y = x; y <= 8; y++)
    {
        if (dot_gamM[y] > dot_gamM[x])
        {
            temp = dot_gamM[x];
            dot_gamM[x] = dot_gamM[y];
            dot_gamM[y] = dot_gamM[x];
        }
        if (dot_gamH[y] > dot_gamH[x])
        {
            temp = dot_gamH[x];
            dot_gamH[x] = dot_gamH[y];
            dot_gamH[y] = dot_gamH[x];
        }
        if (dot_gamB[y] > dot_gamB[x])
        {
            temp = dot_gamB[x];
            dot_gamB[x] = dot_gamB[y];
            dot_gamB[y] = dot_gamB[x];
        }
    }
}

jimer = dot_gamM[0];
jihij = dot_gamH[0];
jibir = dot_gamB[0];

Color newPixelColor = Color.FromArgb(jimer, jihij, jibir);
img2.SetPixel(i, j, newPixelColor);
}
```

Hasil operasi pengolahan citra Minimum:

Gambar 15. Citra Awal

Gambar 16. Citra Hasil Minimum

Kesimpulan

Area Process yaitu algoritma yang digunakan untuk memanipulasi citra pada pixel utama dengan melibatkan pixel tetangga. Yang termasuk pada area process yaitu

1. Konvolusi yang terdiri dari: Embossing, Blurring, Sharpening, Edge Detection.
2. Filtering, yang terdiri dari: Median, Average, Maximum, Minimum.

Konvolusi sangat banyak dipergunakan dalam pengolahan citra untuk memperhalus (smoothing), menajamkan (crispening), mendeteksi tepi (edge detection), serta efek lainnya. Sedang Mean Filtering itu sederhana, intuitif dan mudah untuk menerapkan metode gambar smoothing, yaitu mengurangi jumlah variasi intensitas antara satu pixel dan berikutnya. Hal ini sering digunakan untuk mengurangi noise pada gambar.

Referensi

[1] Setiawan Hadi, "pcd00-pendahuluan", pptx, 2010
[2] Agus Arif, "konvolusi", pdf, 2007
[3] R. Fisher, S. Perkins, A.Walker and E. Wolfart, "Mean Filter", http://homepages.inf.ed.ac.uk/rbf/HIPR2/mean.htm, (Diakses 15 Maret 2010)

Biografi Penulis

Yudi Ahmad Hambali , lahir di Purwakarta, 30 Mei 1990. Telah menyelesaikan sekolah menengah atas di SMA N 1 Purwakarta. Saat ini, saya sedang menjalani kuliah semester akhir di Universitas Padjadjaran, program studi Teknik Informatika dan sedang menyusun tugas akhir.