Course IPCis: Image Processing with C#
Chapter C5: The Convolution Project

Projekt convolution1

Main Menu nach dem Start von VS 2008: `File -> New Project... -> Visual Studio installed templates: Windows Forms Application Name: convolution1 -> Location: C:\temp -> Create directory for solution:` ausschalten `-> OK`
Sie müssen zwei überflüssige Files löschen: `Form1.Designer.cs` und `Program.cs`.
Außerdem löschen Sie den gesamten Code von `Form1.cs`.

Seitenaufbau, Bildaufbau

Schreiben in das leere Codefenster `Form1.cs` folgenden Code:

```
using System;
using System.Drawing;
using System.Windows.Forms;
public class Form1 : Form
{ static void Main() { Application.Run( new Form1() ); }
  const Int32 xSize = 11;
  const Int32 ySize = 12;
  Byte[,]  i0    = new Byte[ySize,xSize];
  Int32[,] kernel;
  Brush[]  brush  = new Brush[10];
  Button[] button = new Button[ySize];
  Int32 i, x, y, dx, dy, n;

  public Form1()
  { BackColor  = Color.White;
    Text       = "Convolution1";
    SetStyle( ControlStyles.ResizeRedraw, true );
    Width  = 800;
    Height = 600;
    for ( i=0; i < 10; i++ )
       brush[i] = new SolidBrush(Color.FromArgb( i*25, i*25, i*25 ) );
    for ( y=0; y < ySize; y++ )
    { button[y] = new Button();
      Controls.Add( button[y] );
      button[y].BackColor = Color.Gray;
      button[y].Text = "nothing";
      button[y].Name = y.ToString();
      button[y].Click += new EventHandler( do_it );
    }
    button[0].Name = button[0].Text = "Homunculus";
    button[1].Name = button[1].Text = "Convolution 3x3";
    button[2].Name = button[2].Text = "Convolution 5x5";
    button[3].Name = button[3].Text = "Ranking Filter 3";
    button[4].Name = button[4].Text = "Noise";
    button[5].Name = button[5].Text = "Clear";
  }
```

```
  protected override void OnPaint(PaintEventArgs e)
  { Graphics g = e.Graphics;
    Rectangle r = ClientRectangle;
    dx = r.Width / (xSize+2);
    dy = r.Height / ySize;
    for ( y=0; y < ySize; y++ )
    { button[y].Top = y*dy+1;
      button[y].Left = xSize*dx+1;
      button[y].Width = 2*dx-2;
      button[y].Height = dy-2;
    }
    for ( y=0; y < ySize; y++ )
      for ( x=0; x < xSize; x++ )
        g.FillRectangle( brush[i0[y,x]], x*dx, y*dy, dx, dy );
  }
  protected void do_it( object sender, System.EventArgs e )
  { switch( ((Button)sender).Name)
    { case "Homunculus"://************************************
        i0 =  new Byte[,]{ {9,0,0,0,0,0,0,0,0,0,9},
                           {0,0,0,0,9,9,9,0,0,0,0},
                           {0,0,0,0,9,5,9,0,0,0,0},
                           {0,0,0,0,9,9,9,0,0,0,0},
                           {0,0,0,0,0,9,0,0,0,0,0},
                           {0,9,9,9,9,9,9,9,9,9,0},
                           {0,0,0,0,9,9,9,0,0,0,0},
                           {0,0,0,0,9,9,9,0,0,0,0},
                           {0,0,0,0,9,0,9,0,0,0,0},
                           {0,0,0,0,9,0,9,0,0,0,0},
                           {0,0,0,0,9,0,9,0,0,0,0},
                           {9,0,0,0,0,0,0,0,0,0,9} };
        Invalidate();
        break;
    }
  }
}
```

Klicken Sie `Debug` -> `Start Without Debugging Ctrl F5`. Erproben Sie das Programm. Nur der erste Button "`Homunculus`" tut etwas, alle anderen sind tot.

Unterprogramm convolution

Version2: Beenden Sie Ihr Programm `convolution1`.
Schreiben Sie eine neue Funktion unterhalb der Klammer, die `protected void do_it(...)` abschließt, aber noch vor die Klammer, die das Hauptprogramm `public class Form1 : System.Windows.Forms.Form` abschließt:

```
  private void convolution( Int32 Z )
  { if ( Z != 3 & Z != 5 ) return;
    Z = (Z - 1) / 2;
    Single[,] help = new Single[ySize,xSize];
    Single gray, factor, min = Single.MaxValue, max = Single.MinValue;
    Int32 xx, yy, xxx, yyy, weight, sum, divisor;
    for ( y=0; y < ySize; y++ )
      for ( x=0; x < xSize; x++ )
      { sum = divisor = 0;
        for ( yy = 0; yy <= 2*Z; yy++ )
        { yyy = y+yy-Z;
          if (yyy < 0 | yyy >= ySize) continue;
          for (xx = 0; xx <= 2*Z; xx++ )
          { xxx = x+xx-Z;
            if (xxx < 0 | xxx >= xSize) continue;
            weight = kernel[yy,xx];
            sum += i0[yyy,xxx] * weight;
            divisor += weight;
          }
```

```
        }
        if ( divisor <= 1 ) gray = (Single)sum;
        else gray = (Single)sum / (Single)divisor;
        if ( gray > max ) max = gray;
        if ( gray < min ) min = gray;
        help[y,x] = gray;
      }
    factor = 9.0f / (max - min);
    for ( y=0; y < ySize; y++ )
      for ( x=0; x < xSize; x++ )
        i0[y,x] = (Byte)( factor * ( help[y,x] - min ) );
  }
```

Schreiben Sie folgende neuen Cases unterhalb des letzten `break;` von `case "Homunculus":` im Event Handler `protected void do_it(...)`:

```
      case "Convolution 3x3"://*********************************
        kernel = new Int32[,] { {0,1,0},
                                {1,4,1},
                                {0,1,0} };
        convolution( 3 );
        Invalidate();
        break;
      case "Convolution 5x5"://*********************************
        kernel = new Int32[,] { {0, 0, 0, 0, 0},
                                {0,-1,-1,-1, 0},
                                {0,-1,10,-1, 0},
                                {0,-1,-1,-1, 0},
                                {0, 0, 0, 0, 0} };
        convolution( 5 );
        Invalidate();
        break;
```

Klicken Sie `Debug` -> `Start Without Debugging Ctrl F5`. Erproben Sie die beiden Filter.

Rangordnungsfilter

Version3: Beenden Sie Ihr Programm `convolution1`.
Schreiben Sie folgende neuen Cases unterhalb des letzten `break;` von `case "Convolution 5x5":` im Event Handler `protected void do_it(...)`:

```
      case "Ranking Filter 3"://*********************************
        Byte[,] help = new Byte[ySize,xSize];
        Byte[] a = new Byte[9];
        for ( y = 0; y < ySize; y++ )
           for ( x = 0; x < xSize; x++ )
           { n = 0;
             Array.Clear(a,0,9);
             try { a[n++] = i0[y-1,x-1]; } catch {}
             try { a[n++] = i0[y-1,x  ]; } catch {}
             try { a[n++] = i0[y-1,x+1]; } catch {}
             try { a[n++] = i0[y  ,x-1]; } catch {}
             try { a[n++] = i0[y  ,x  ]; } catch {}
             try { a[n++] = i0[y  ,x+1]; } catch {}
             try { a[n++] = i0[y+1,x-1]; } catch {}
             try { a[n++] = i0[y+1,x  ]; } catch {}
             try { a[n++] = i0[y+1,x+1]; } catch {}
             Array.Sort(a,0,n);
             help[y,x] = a[n/2];
           }
        for ( y=0; y < ySize; y++ )
          for ( x=0; x < xSize; x++ )
            i0[y,x] = help[y,x];
        Invalidate();
        break;
```

Klicken Sie `Debug` -> `Start Without Debugging Ctrl F5`. Erproben Sie den Rangordnungsfilter.

Noise und Clear

Version3: Beenden Sie Ihr Programm `convolution1`.
Schreiben Sie folgende neuen Cases unterhalb des letzten `break;` von `case "Ranking Filter 3":` im Event Handler `protected void do_it(...)`:

```
      case "Noise"://*****************************************
        Random random = new Random();
        for ( y=0; y < ySize; y++ )
          for ( x=0; x < xSize; x++ )
          { Int32 noise = random.Next() % 3 - 1;
            noise += i0[y,x];
            if (noise < 0) i0[y,x] = 0;
            else if (noise > 9) i0[y,x] = 9;
            else                i0[y,x] = (Byte)noise;
          }
        Invalidate();
        break;
      case "Clear"://*****************************************
        for ( y=0; y < ySize; y++ )
          for ( x=0; x < xSize; x++ ) i0[y,x] = 0;
        Invalidate();
        break;
```

Klicken Sie `Debug` -> `Start Without Debugging Ctrl F5`. Erproben Sie `Noise` (auch mehrfach drücken) und `Clear`.

Experimente

(1) Erproben Sie andere 3x3 Filterkerne:

```
-2, 1, 1                          1, 1,-2
-2, 1, 1 = Filter für linke Kanten  1, 1,-2 = Filter für rechte Kanten
-2, 1, 1                          1, 1,-2

-2,-2,-2                          1, 1, 1
 1, 1, 1 = Filter für obere Kanten  1, 1, 1 = Filter für untere Kanten
 1, 1, 1                         -2,-2,-2

-1,-1,-1                          1, 1, 1
-1, 8,-1 = Filter für einsame Pixel  1,-8, 1 = Filter für einsame Löcher
-1,-1,-1                          1, 1, 1
```

(2) Entwerfen Sie weitere 3x3 Filterkerne speziell für die linke Hand, für die rechte Hand und 5x5 Filterkerne für beide Füße und für den Hals des Homunculus.
Bauen Sie diese in die Buttonleiste ein.
(3) Variieren Sie den Homunculus und beobachten Sie die Ergebnisse Ihrer Filterkerne.
(4) Verschieben Sie im Rangordnungsfilter das ausgewählte Pixel (Befehl `help[y,x] = a[n/2];`) nach höheren oder nach niederen Rangordnungen. Beobachten Sie die Änderungen.

Weitere Aufgaben

Klicken Sie auf `Help` in der Menüleiste von Visual Studio. Klicken Sie auf das Untermenü `Index`.
Gehen Sie in das Feld `Filtered by:` und wählen Sie dort `.NET Framework`.
Dann geben Sie im Feld `Look for:` folgende Schlüsselwörter ein und lesen Sie die Texte:
`Array class, all members`, Suchen sie die Methoden `Clear` und `Sort.`
Erarbeiten Sie sich die Logik der 4 verschachtelten `for`-Schleifen im Unterprogramm `convolution`.
Programmieren Sie eine vereinfachte `myconvolution`-Funktion der festen Größe 3x3.