Code Samples:
The Simple Gauss Filter in C#

Copyright © by W. Kovalevski and V. Miszalok, last update: 13-02-2006

This code has been developed with Visual C#2.0.
The algorithm makes its own quadratic filter kernel with circularily symmetric values of a Gaussian bell-shaped curve.
The bell-shaped curve automatically adjusts (at any kernel size) its central value to 1.0 and its corner values to 0.01.
Only exception: The 3x3-kernel has 0.14 at its corners.
The filtering itself has the classical form of a convolution with 4 nested `for`-loops and a final division by the sum of the kernel.
The algorithm accepts any quadratic filter sizes `NxN` where `N` must be a positive odd integer `N < b0.Width, N < b0.Height`.
The algorithm works with any image format that can be read and written with the `GetPixel(..), SetPixel(..)` methods of the `Bitmap`-class.
Time depends on ImageSize*FilterSize = `b0.Width*b0.Height*N*N`.

Create a new Windows-project "`gauss_simple`". Delete the files `Form1.Designer.cs` and `Program.cs` from the project. Clear any prefabricated content from `Form1.cs` and replace it by the following code:

```
using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.Windows.Forms;

public class Form1 : Form
{ static void Main() { Application.Run( new Form1() );  }
  Bitmap b0, b1;
  const Int32 N = 7, Nh = N/2;  //N must be an odd integer < b0.Width && < b0.Height
  Byte [,] R0, G0, B0;
  String s;
  float[,] kernel = new float[N,N]; //quadratic Gauss kernel
  float sum_kernel = 0.0f;

  public Form1()
  { MenuItem miRead = new MenuItem( "&Read", new EventHandler( MenuFileRead ) );
    MenuItem miExit = new MenuItem( "&Exit", new EventHandler( MenuFileExit ) );
    MenuItem miFile = new MenuItem( "&File", new MenuItem[] { miRead, miExit } );
    Menu = new System.Windows.Forms.MainMenu( new MenuItem[] { miFile } );
    Text = "Gauss Filter";
    SetStyle( ControlStyles.ResizeRedraw, true );
    try { b0 = new Bitmap( typeof( Form1 ), "gauss_simple.Butterfly.jpg" );
          byte_arrays_and_b1_image();
          convolution_with_a_gauss_filter();
          border_painting();
        } catch {}
    Width  = 800;
    Height = 600;
  }

  void MenuFileRead( object obj, EventArgs ea )
  { OpenFileDialog dlg = new OpenFileDialog();
    dlg.Filter = "bmp files (*.bmp)|*.bmp|All files (*.*)|*.*"  ;
    if ( dlg.ShowDialog() != DialogResult.OK ) return;
    b0 = (Bitmap)Image.FromFile( dlg.FileName );
    byte_arrays_and_b1_image();
    convolution_with_a_gauss_filter();
    border_painting();
    Invalidate();
  }

  void MenuFileExit( object obj, EventArgs ea )
  { Application.Exit(); }
```

```
protected override void OnPaint( PaintEventArgs e )
{ Graphics g = e.Graphics;
  g.Clear( BackColor );
  try
  { g.DrawImage( b0, 0, 0, b0.Width, b0.Height );
    g.DrawImage( b1, b0.Width+10, 0 );
    g.DrawString( s, new Font( "Arial", 16 ), new SolidBrush( Color.Red ),
                  0, ClientRectangle.Height-120 );
  } catch{}
}

private void byte_arrays_and_b1_image()
{ R0 = new Byte [b0.Height, b0.Width];
  G0 = new Byte [b0.Height, b0.Width];
  B0 = new Byte [b0.Height, b0.Width];
  if ( b1 != null ) b1.Dispose();
  b1 = new Bitmap( b0 );
  for ( int y=0; y < b0.Height; y++ )
    for ( int x=0; x < b0.Width; x++ )
    { Color c = b0.GetPixel( x, y );
      R0[y,x] = (Byte)c.R;
      G0[y,x] = (Byte)c.G;
      B0[y,x] = (Byte)c.B;
      b1.SetPixel( x, y, Color.Black );
    }
}

private void convolution_with_a_gauss_filter()
{ Int32 x, xx, xxx, y, yy, yyy;
  Int64 t0, t1;
  float sumR, sumG, sumB, weight, Rf, Gf, Bf;
  Cursor.Current = Cursors.WaitCursor;
  t0 = DateTime.Now.Ticks;
  //Construction of a suitable 2D-Gaussian bell-shape kernel:
  //The corner elements of the kernel are Nh*sqrt(2) away from its center
  //and therefore obtain the lowest values.
  //We adjust the parameter a of the e-function y = exp(-(x*x/a) so,
  //that always (at any kernel size except 3x3) these corners obtain at least 1% weight.
  double a = 1.0f; // default value for N <= 3
  if ( N > 3 ) a = - 2 * Nh * Nh / Math.Log( 0.01 );
  //fill the kernel with elements depending on their distance and on a.
  for ( y=0; y < N; y++ )
    for ( x=0; x < N; x++ )
    { double dist = Math.Sqrt( (x-Nh)*(x-Nh) + (y-Nh)*(y-Nh) );
      sum_kernel += kernel[y,x] = (float)( Math.Exp( - dist*dist / a ) );
    }
  //Convolution
  for ( y=Nh; y < b0.Height-Nh; y++ ) //==================
  { for ( x=Nh; x < b0.Width-Nh; x++ ) //===============
    { sumR = sumG = sumB = 0.0f;
      for ( yy=-Nh; yy <= Nh; yy++ ) //=============
      { yyy = y + yy;
        for ( xx=-Nh; xx <= Nh; xx++ )//========
        { weight = kernel[yy+Nh,xx+Nh];
          xxx = x + xx;
          sumR += weight * R0[yyy,xxx];
          sumG += weight * G0[yyy,xxx];
          sumB += weight * B0[yyy,xxx];
        } //====== end for (int xx... ================
      } //======== end for (int yy... ==================
      Rf = sumR/sum_kernel; Gf = sumG/sum_kernel; Bf = sumB/sum_kernel;
      b1.SetPixel( x, y, Color.FromArgb( Convert.ToInt32(Rf),
                                         Convert.ToInt32(Gf),
                                         Convert.ToInt32(Bf) ) );
    } //============ end for (int x... =====================
  } //============== end for (int y... =======================
  t1 = DateTime.Now.Ticks;
  s = "Simple quadratic Gauss filter\r\n" +
      "Image:  " + b0.Width.ToString() + " x " + b0.Height.ToString() + "\r\n" +
      "Filter: " + N.ToString() + " x " + N.ToString() + "\r\n" +
      "Filter Time: " + String.Format( "{0:F1}", (t1 - t0)/1000000f ) + " MegaTicks";
  Cursor.Current = Cursors.Arrow;
}
```

```
  private void border_painting()
  { Graphics gb1 = Graphics.FromImage( b1 );
    Pen pen = new Pen( Color.Black );
    SolidBrush brush = new SolidBrush( Color.Black );
    //left and right border
    for ( int y=Nh; y < b0.Height-Nh; y++ )
    { pen.Color = b1.GetPixel( Nh, y );
      gb1.DrawLine( pen, 0, y, Nh-1, y );
      pen.Color = b1.GetPixel( b0.Width-Nh-1, y );
      gb1.DrawLine( pen, b0.Width-Nh, y, b0.Width-1, y );
    }
    //upper and lower border
    for ( int x=Nh; x < b0.Width-Nh; x++ )
    { pen.Color = b1.GetPixel( x, Nh );
      gb1.DrawLine( pen, x, 0, x, Nh-1 );
      pen.Color = b1.GetPixel( x, b0.Height-Nh-1 );
      gb1.DrawLine( pen, x, b0.Height-Nh, x, b0.Height-1 );
    }
    //corners
    brush.Color = b1.GetPixel( Nh, Nh ); //left upper
    gb1.FillRectangle( brush, 0, 0, Nh, Nh );
    brush.Color = b1.GetPixel( b0.Width-Nh-1, Nh ); //right upper
    gb1.FillRectangle( brush, b0.Width-Nh, 0, Nh, Nh );
    brush.Color = b1.GetPixel( Nh, b0.Height-Nh-1 ); //left lower
    gb1.FillRectangle( brush, 0, b0.Height-Nh, Nh, Nh );
    brush.Color = b1.GetPixel( b0.Width-Nh-1, b0.Height-Nh-1 ); //right lower
    gb1.FillRectangle( brush, b0.Width-Nh, b0.Height-Nh, Nh, Nh );
  }
}
```

Recommended experiments:
1) In line 9 of the code change `const Int32 N = 7` to
any odd integer `1 ≤ N < image.Width && 1 ≤ N < image.Height`.
Try out extreme values `N = 1` and `N = 249`.
2) Load different images via the `File` menu.
3) Investigate the `kernel`-matrix with the debugger.
4) Comment out the `border_painting();`-functions in lines 25 and 38.

How to embed an arbitrary sample image into the code:
Copy all the code into an empty `Form1.cs` of a new `Windows Application C#`-project `gauss_simple` and delete `Form1.Designer.cs` and `Program.cs`.
1) In the `Solution Explorer` window right click on `gauss_simple`. A context menu opens.
2) Click `Add`. Click `Existing Item`. A file dialog box opens.
3) At the bottom of the box is a drop-down menu: `Files of type:`
4) Select `Image Files (*.gif;*.jpg;...)`
5) Choose an arbitrary image from your computer and leave the file dialog box with button `Add`.
6) The file name should now appear in the `Solution Explorer` tree. Right click this name.
7) A context menu opens. Click `Properties`.
A `Properties`-window opens below the `Solution Explorer` window.
8) Click its first property: `Build Action` and set it to `Embedded Resource`.
9) Line 22 of the program below is:
`b0 = new Bitmap( typeof( Form1 ), "gauss_simple.Butterfly.jpg" );`
Replace `Butterfly.jpg` by the name of your image.