Code Samples: The Simple Averaging Filter in C#

This code has been developed with Visual C#2.0.
It contains two versions of the same algorithm:
a) `average_no_border()`
b) `average_with_border()`.
Both are very similar and work with 4 nested `for`-loops. They accept any rectangular filter sizes `M≠N` or `M==N` where `M, N` must be positive odd integers `M < b0.Width, N < b0.Height`.
Both work with any image format that can be read and written with the `GetPixel(..), SetPixel(..)` methods of the `Bitmap`-class.
a) is probably the most simple of all possible average filters.
Time depends on ImageSize*FilterSize = `b0.Width*b0.Height*M*N`.
b) just contains 3 more statements than a) to process the image borders. It is hard to believe how these 3 statements slow it down.

Create a new Windows-project "`simple_average_filter`". 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 M =  21, Mh = M/2;          //M must be an odd integer < b0.Width
  const Int32 N =  21, Nh = N/2, MN=M*N;  //N must be an odd integer < b0.Height
  Byte [,] R0, G0, B0;
  String s;

  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 = "Simple Average Filter";
    SetStyle( ControlStyles.ResizeRedraw, true );
    try { b0 = new Bitmap( typeof( Form1 ), "simple_average_filter.Butterfly.jpg" );
          byte_arrays_and_b1_image();
          average_no_border(); // change to average_with_border();
        } catch {}
    Width  = 800;
    Height = 600;
  }

  void MenuFileRead( object obj, EventArgs ea )
  { OpenFileDialog dlg = new OpenFileDialog();
    if ( dlg.ShowDialog() != DialogResult.OK ) return;
    b0 = (Bitmap)Image.FromFile( dlg.FileName );
    byte_arrays_and_b1_image();
    average_no_border(); // change to average_with_border();
    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 average_no_border()
{ Int32 sumR, sumG, sumB, x, xx, xxx, y, yy, yyy;
  Int64 t0, t1;
  float Rf, Gf, Bf;
  Cursor.Current = Cursors.WaitCursor;
  t0 = DateTime.Now.Ticks;
  for ( y=Nh; y < b0.Height-Nh; y++ ) //==================
  { for ( x=Mh; x < b0.Width-Mh; x++ ) //===============
    { sumR=sumG=sumB=0;
      for ( yy=-Nh; yy <= Nh; yy++ ) //=============
      { yyy = y + yy;
        for ( xx=-Mh; xx <= Mh; xx++ )//========
        { xxx = x + xx;
          sumR+=R0[yyy,xxx];
          sumG+=G0[yyy,xxx];
          sumB+=B0[yyy,xxx];
        } //====== end for (int xx... ================
      } //======== end for (int yy... ==================
      Rf = sumR/MN; Gf = sumG/MN; Bf = sumB/MN;
      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 average filter without border handling\r\n" +
      "Image:  " + b0.Width.ToString() + " x " + b0.Height.ToString() + "\r\n" +
      "Filter: " + M.ToString() + " x " + N.ToString() + "\r\n" +
      "Filter Time: " + String.Format( "{0:F1}", (t1 - t0)/1000000f ) + " MegaTicks";
  Cursor.Current = Cursors.Arrow;
}
```

```
  private void average_with_border()
  { Int32 nS, sumR, sumG, sumB, x, xx, xxx, y, yy, yyy;
    Int64 t0, t1;
    float Rf, Gf, Bf;
    t0 = DateTime.Now.Ticks;
    Cursor.Current = Cursors.WaitCursor;
    for ( y=0; y < b0.Height; y++ ) //==================
    { for ( x=0; x < b0.Width; x++ ) //===============
      { nS=sumR=sumG=sumB=0;
        for ( yy=-Nh; yy <= Nh; yy++ ) //=============
        { yyy = y + yy;
          if ( yyy >= 0 && yyy < b0.Height )
            for ( xx=-Mh; xx <= Mh; xx++ )//========
            { xxx = x + xx;
              if ( xxx >= 0 && xxx < b0.Width )
              { nS++;
                sumR+=R0[yyy,xxx];
                sumG+=G0[yyy,xxx];
                sumB+=B0[yyy,xxx];
              }
            } //====== end for (int xx... ================
          } //======== end for (int yy... ==================
        Rf = sumR/nS; Gf = sumG/nS; Bf = sumB/nS;
        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 average filter with border handling\r\n" +
        "Image:  " + b0.Width.ToString() + " x " + b0.Height.ToString() + "\r\n" +
        "Filter: " + M.ToString() + " x " + N.ToString() + "\r\n" +
        "Filter Time: " + String.Format( "{0:F1}", (t1 - t0)/1000000f ) + " MegaTicks";
    Cursor.Current = Cursors.Arrow;
  }
}
```

Recommended experiments:
1) In line 9 of the code change `const Int32 M = 21` to any odd integer `1 ≤ M < image.Width`.
2) In line 10 of the code change `const Int32 N = 21` to any odd integer `1 ≤ N < image.Height`.
3) Load different images via the `File` menu.
4) Replace the calls `average_no_border();` in lines 23 and 35 by `average_with_border();` and notice the enormous computing costs of border handlig.

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 `simple_average_filter` and (in case of VS 2005) delete `Form1.Designer.cs` and `Program.cs`.
1) In the `Solution Explorer` window right click on `simple_average_filter`. 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 21 of the program below is: `b0 = new Bitmap( typeof( Form1 ), "simple_average_filter.Butterfly.jpg" );`
Replace `Butterfly.jpg` by the name of your image.