

Code Samples: The Median Filter in C#

Copyright © by W. Kovalevski and V. Miszalok, last update: 14-02-2006

This code has been developed with Visual C#2.0.

The algorithm realizes the well known median filter = sorting the pixels inside the filter window from dark to bright just for choosing the value in the mid of the range. The median filter is still popular because of its noise reduction without blurring properties. Nevertheless it should be replaced by the sigma filter which is better, faster and produces less artifacts.

The filtering itself has the classical form with 4 nested `for`-loops. The usual division is replaced by a quick sort algorithm.

The algorithm accepts any quadratic filter sizes $N \times N$ 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 "median". 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;

    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 = "Median Filter";
        SetStyle( ControlStyles.ResizeRedraw, true );
        b0 = new Bitmap( 300, 300 );
        Graphics gb0 = Graphics.FromImage( b0 );
        gb0.Clear( Color.Black );
        Pen pen1 = new Pen( Color.White );
        Pen pen2 = new Pen( Color.White, 3 );
        SolidBrush wbrush = new SolidBrush( Color.White );
        SolidBrush rbrush = new SolidBrush( Color.Red );
        for ( int i=0; i < 10; i++ ) gb0.DrawLine( pen1, 20+2*i, 10*i, 20+2*i, b0.Height-10*i );
        for ( int i=0; i < 10; i++ ) gb0.DrawLine( pen1, 10*i, 20+2*i, b0.Width-10*i, 20+2*i );
        gb0.DrawRectangle( pen2, 120, 120, 100, 100 );
        gb0.FillRectangle( wbrush, 100, 100, 100, 100 );
        gb0.FillRectangle( rbrush, 120, 120, 60, 60 );
        byte_arrays_and_b1_image();
        median_filter();
        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;
        if ( b0 != null ) b0.Dispose();
        b0 = (Bitmap)Image.FromFile( dlg.FileName );
        byte_arrays_and_b1_image();
        median_filter();
        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 median_filter()
{ Int32 x, xx, xxx, y, yy, yyy, i, n = N*N/2;
  Byte[] r = new Byte[N*N], g = new Byte[N*N], b = new Byte[N*N];
  Int64 t0, t1;
  Cursor.Current = Cursors.WaitCursor;
  t0 = DateTime.Now.Ticks;
  for ( y=Nh; y < b0.Height-Nh; y++ ) //=====
  { for ( x=Nh; x < b0.Width-Nh; x++ ) //=====
    { i = 0;
      for ( yy=-Nh; yy <= Nh; yy++ ) //=====
      { yyy = y + yy;
        for ( xx=-Nh; xx <= Nh; xx++ ) //=====
        { xxx = x + xx;
          r[i] = R0[yyy,xxx];
          g[i] = G0[yyy,xxx];
          b[i] = B0[yyy,xxx]; i++;
        } //===== end for (int xx... =====
      } //===== end for (int yy... =====
      Array.Sort(r); Array.Sort(g); Array.Sort(b);
      b1.SetPixel( x, y, Color.FromArgb( r[n], g[n], b[n] ) );
    } //===== end for (int x... =====
  } //===== end for (int y... =====
  t1 = DateTime.Now.Ticks;
  s = "Median 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;
}
}

```

Recommended experiments:

- 1) In line 9 of the code change `const Int32 N = 7` to any odd integer $1 \leq N < \text{image.Width} \ \&\& \ 1 \leq N < \text{image.Height}$. Try out extreme values $N = 1$ and $N = 249$ (caution: very time consuming > 250 MegaTicks).
- 2) Load different images by using the File menu.