

Course IPCis: Image Processing with C#

Chapter C2: The Complete Code of the Histogram Project

Copyright © by V. Miszalok, last update: 09-01-2008

Copy all this code into an empty Form1.cs of a new Windows Application C#-project histo1 and clear Form1.Designer.cs and Program.cs.

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

public class Form1 : Form
{
    static void Main() { Application.Run( new Form1() ); }
    Brush bbrush = SystemBrushes.ControlText;
    Brush wbrush = new SolidBrush( Color.White );
    Pen bpen = SystemPens.ControlText;
    Pen rpen = new Pen( Color.Red );
    Bitmap bmp, bmp_binary, bmp_histo;
    BitmapData binaryData; //for Versions 2 and 3
    Byte[,] grayarray; //2D-Byte-Array
    Int32[] Histogram = new Int32[256];
    Rectangle histo_r = new Rectangle( 0,0,257,101 );
    Graphics g, g_histo;
    Byte[] ORmask = { 128, 64, 32, 16, 8, 4, 2, 1 };// 1 bit each //for Version 3
    Byte[] ANDmask = { 127, 191, 223, 239, 247, 251, 253, 254 };// 7 bits each //for Version 3

    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 = "Histo1";
        SetStyle( ControlStyles.ResizeRedraw, true );
        Width = 800;
        Height = 600;
    }

    void MenuFileRead( object obj, EventArgs ea )
    {
        OpenFileDialog dlg = new OpenFileDialog();
        if ( dlg.ShowDialog() != DialogResult.OK ) return;
        try
        {
            Cursor.Current = Cursors.WaitCursor;
            bmp = (Bitmap)Image.FromFile( dlg.FileName );
            GenerateTheHistogram();
            Cursor.Current = Cursors.Arrow;
            Invalidate();
        } catch {}
    }

    void GenerateTheHistogram()
    {
        if ( bmp == null ) return;
        //bmp_binary = new Bitmap( bmp.Width, bmp.Height, PixelFormat.Format32bppRgb ); //Version 1
        //bmp_binary = new Bitmap( bmp.Width, bmp.Height, PixelFormat.Format32bppRgb ); //Version 2
        bmp_binary = new Bitmap( bmp.Width, bmp.Height, PixelFormat.Format1bppIndexed ); //Version 3
        grayarray = new Byte[bmp.Height, bmp.Width];
        Color color;
        for ( Int32 y=0; y < bmp.Height; y++ )
            for ( Int32 x=0; x < bmp.Width; x++ )
            {
                color = bmp.GetPixel( x, y );
                Int32 gray = ( color.R + color.G + color.B ) / 3;
                grayarray[y, x] = (Byte)gray;
                Histogram[gray]++;
            }
        Int32 hmax = 0;
        for ( Int32 i=0; i < 256; i++ )
            if ( Histogram[i] > hmax ) hmax = Histogram[i];
        for ( Int32 i=0; i < 256; i++ )
            Histogram[i] = (100*Histogram[i]) / hmax;
        bmp_histo = new Bitmap( histo_r.Width, histo_r.Height, PixelFormat.Format32bppRgb );
        g_histo = Graphics.FromImage( bmp_histo );
        g_histo.FillRectangle( wbrush, 0,0,256,100 );
        g_histo.DrawString( "click here and move!", Font, bbrush, 1, 1 );
        for ( Int32 i=0; i < 256; i++ ) g_histo.DrawLine( bpen, i, 100, i, 100 - Histogram[i] );
        g_histo.DrawRectangle( rpen, 0,0,256,100 );
    }
}
```

```

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

protected override void OnMouseMove( MouseEventArgs e )
{ if ( e.Button == MouseButtons.None ) return;
  if ( !histo_r.Contains( e.X, e.Y ) ) return;
  if ( bmp == null ) return;
  Byte threshold = (Byte)(e.X - histo_r.X);

  /*
  //Version 1 (no pointers but slow)*****
  for ( Int32 i=0; i < bmp.Height; y++ )
    for ( Int32 x=0; x < bmp.Width; x++ )
      { if ( grayarray[y ,x] > threshold )
        bmp_binary.SetPixel( x, y, Color.White );
        else bmp_binary.SetPixel( x, y, Color.Black );
      }
  //End of Version 1 *****
  */
  /*
  //Version 2 (fast pointers creating a memory wasting 32-bit binary image)*
  unsafe
  { //lock bmp_binary from being shifted in memory by the garbage collector
    binaryData = bmp_binary.LockBits( new Rectangle( 0,0,bmp.Width,bmp.Height ),
      ImageLockMode.WriteOnly, PixelFormat.Format32bppRgb );
    //Byte* plfix, plrun = fixed + running pointers to grayarray = input image
    UInt32* p2fix, p2run; //fixed + running pointers to binaryData = output image
    fixed ( Byte* plfix = grayarray ) // lock grayarray in memory
    { Byte* plrun = plfix; // running pointer to grayarray
      p2fix = (UInt32*)binaryData.Scan0; // pointer to output image
      for ( int y=0; y < bmp.Height; y++ )
        { p2run = p2fix + y * bmp.Width; //p2run points to first byte in row y
          for ( int x=0; x < bmp.Width; x++ )
            { if ( *plrun++ > threshold ) *p2run++ = 0xFFFFF; // white
              else *p2run++ = 0; // black
            } // end of for x
          } // end of for y
        } // end of fixed, end of plfix, unlock grayarray
    bmp_binary.UnlockBits( binaryData ); //end of p2fix, unlock bmp_binary
  } // end of unsafe
  //End of Version 2 *****
  */

  //Version 3 (fast pointers creating a 1-bit binary image)*****
  unsafe
  { //lock bmp_binary from being shifted in memory by the garbage collector
    binaryData = bmp_binary.LockBits( new Rectangle( 0,0,bmp.Width,bmp.Height ),
      ImageLockMode.WriteOnly, PixelFormat.Format1bppIndexed );
    Byte* p2fix, p2row, p2run; // pointers to binaryData = output image
    fixed ( Byte* plfix = grayarray ) // lock grayarray = input image in memory
    { Byte* plrun = plfix; // running pointer to grayarray
      p2fix = (byte*)binaryData.Scan0; // pointer to output image
      for ( int y=0; y < bmp.Height; y++ )
        { p2row = p2fix + y * binaryData.Stride; //p2row points to first byte in row y
          for ( int x=0; x < bmp.Width; x++ )
            { p2run = p2row + x / 8;
              if ( *plrun++ > threshold ) *p2run |= ORmask[ x % 8 ]; //set 1 bit
              else *p2run &= ANDmask[ x % 8 ]; //remove 1 bit
            } // end of for x
          } // end of for y
        } // end of fixed, end of plfix, unlock grayarray
    bmp_binary.UnlockBits( binaryData ); // end of p2fix, unlock bmp_binary
  } // end of unsafe
  //End of Version 3 *****

  g = CreateGraphics();
  g.DrawImage( bmp_binary, ClientRectangle );
  g.DrawImage( bmp_histo, histo_r );
  g.DrawLine( rpen, histo_r.X+threshold, histo_r.Y,
    histo_r.X+threshold, histo_r.Y+histo_r.Height-1 );
}

protected override void OnMouseUp(MouseEventArgs e)
{ Invalidate(); }

protected override void OnPaint( PaintEventArgs e )
{ if ( bmp == null )
  { e.Graphics.DrawString( "Open an Image File !", Font, bbrush, 0, 0 ); return; }
  e.Graphics.DrawImage( bmp, ClientRectangle );
  histo_r.X = ClientRectangle.Width - histo_r.Width - 10;
  histo_r.Y = ClientRectangle.Height - histo_r.Height - 10;
  e.Graphics.DrawImage( bmp_histo, histo_r );
}
}

```