

Course CVCis: Computer Vision with C#

Chapter C6: The Complete Code of the Fourier Transform Project

Copyright © by V. Miszalok, last update: 13-01-2008

Main Menu after start of VS 2008: File -> New Project... ->
 Visual Studio installed templates: Windows Forms Application
 Name: fourier_transform1 -> Location: C:\temp -> Create directory for solution:
 switch off -> OK

Delete two superfluous files: Form1.Designer.cs and Program.cs.

Replace the pre-programmed code of Form1.cs by:

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Collections;
using System.Text;
using System.IO;

public class Form1 : Form
{
    static void Main() { Application.Run( new Form1() ); }
    const Int32 xSize = 16;
    const Int32 ySize = 16;
    Byte[,] i0      = new Byte[ySize, xSize]; // raster matrix
    Byte threshold  = 1;
    Brush[] brush   = new Brush[10]; //for 10 different gray values
    Brush redbrush  = new SolidBrush( Color.Red   );
    Brush bluebrush = new SolidBrush( Color.Blue  );
    Brush graybrush = new SolidBrush( Color.Gray  );
    Brush greenbrush = new SolidBrush( Color.Green );
    Pen redpen1     = new Pen( Color.Red,   1 );
    Pen redpen5     = new Pen( Color.Red,   5 );
    Pen greenpen    = new Pen( Color.Green, 5 );
    Pen bluepen     = new Pen( Color.Blue,  5 );
    Pen yellowpen   = new Pen( Color.Yellow,5 );
    Pen whitepen;
    Font arial10    = new Font( "Arial", 10 );
    Font arial20    = new Font( "Arial", 20 );

    Button[] button = new Button[ySize];
    Int32 i, x, y, dx, dy, backtransform;
    Point p0, p1;
    struct crackcode { public Point start; public String cracks; }
    struct FPoint    { public float x;    public float y; }
    crackcode cc;
    Graphics g;
    ArrayList pointArray = new ArrayList();
    ArrayList phil;
    float[] phi2, phi3, fr, fi;

    public Form1()
    {
        BackColor = Color.White;
        Text      = "Fourier Transform";
        SetStyle( ControlStyles.ResizeRedraw, true );
        Width    = 800;
        Height   = 600;
        for ( i=0; i < 10; i++ ) // 10 brushes with 10 different gray values
            brush[i] = new SolidBrush( Color.FromArgb( i*25, i*25, i*25 ) );
        for ( y=0; y < ySize; y++ ) //a column of buttons
        {
            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 = "FourierTransf";
button[1].Name = "Back";
button[2].Name = button[2].Text = "Clear";
button[3].Name = "Threshold";
button[4].Name = button[4].Text = "Noise";
button[1].Text = "Back=0";
button[3].Text = "Threshold=1";
cc.cracks = "";
} // end of constructor

protected override void OnPaint( PaintEventArgs e )
{
    g = e.Graphics;
    String s;
    Rectangle r = ClientRectangle;
    dx = r.Width / (xSize+2);
    dy = (r.Height - 2 * FontHeight) / ySize;
    for ( y=0; y < ySize; y++ )
    {
        button[y].Top = y*dy+1;
        button[y].Left = xSize*dx+1;
        button[y].Width = r.Width - button[y].Left - 2;
        button[y].Height = dy-2;
    }
    button[1].Text = "Back=" + backtransform.ToString();
    button[3].Text = "Threshold=" + threshold.ToString();
    //draw the raster matrix
    for ( y=0; y < ySize; y++ )
        for ( x=0; x < xSize; x++ )
            g.FillRectangle( brush[i0[y,x]], x*dx, y*dy, dx, dy );
    //draw the initial invitation
    if ( cc.cracks.Length == 0 )
    {
        s = "Draw something -> FourierTransf -> BackTransf";
        g.DrawString( s, arial10, redbrush, (xSize*dx)/4, 0 );
        return;
    }
    //draw the crack code
    x = cc.start.X;
    y = cc.start.Y;
    for ( i = 0; i < cc.cracks.Length; i++ )
        switch ( cc.cracks[i] )
        {
            case 'e': g.DrawLine( redpen5, x*dx, y*dy, (x+1)*dx, y*dy ); x++; break;
            case 's': g.DrawLine( redpen5, x*dx, y*dy, x*dx, (y+1)*dy ); y++; break;
            case 'w': g.DrawLine( redpen5, x*dx, y*dy, (x-1)*dx, y*dy ); x--; break;
            case 'n': g.DrawLine( redpen5, x*dx, y*dy, x*dx, (y-1)*dy ); y--; break;
        }
    g.FillRectangle( bluebrush, x*dx-5, y*dy-5, 11, 11 );
    //draw some info into the white space below the image
    try
    {
        s = " No. of cracks = " + cc.cracks.Length.ToString() + " ";
        s += " No. of Fourier steps = " + fr.Length.ToString();
        g.DrawString( s, arial10, redbrush, 0, ySize*dy );
    } catch { return; }
    //draw both Fourier coefficient curves
    g.DrawString( "fr", arial20, greenbrush, 10, 10 );
    g.DrawString( "fi", arial20, bluebrush, 10, 40 );
    float x0, y0, x1, y1, y0fr, y1fr, y0fi, y1fi, y_zoom = 3;
    for ( i=0; i < fr.Length; i++ )
    {
        x0 = i * xSize*dx / fr.Length;
        x1 = (i+1) * xSize*dx / fr.Length;
        y0fr = y_zoom * fr[i] + r.Height/2;
        y0fi = y_zoom * fi[i] + r.Height/2;
        //fr[i+1], fi[i+1] can be beyond the arrays. If so, set them to 0.
        try { y1fr = y_zoom * fr[i+1] + r.Height/2;
            y1fi = y_zoom * fi[i+1] + r.Height/2; }
        catch { y1fr = y1fi = r.Height/2; }
        g.DrawLine( greenpen, x0, y0fr, x1, y1fr );
        g.DrawLine( bluepen, x0, y0fi, x1, y1fi );
    }
    //draw the no.s of coefficients on the x-axis
    Int32 step = fr.Length < (ySize*dy / 10) ? 1 : 2;
    for ( i=0; i <= fr.Length; i+=step )
    {
        x0 = i * xSize*dx / fr.Length;
        if ( i <= backtransform ) g.FillRectangle( graybrush, x0-4, r.Height/2-5, 17, 17 );
        g.DrawString( i.ToString(), arial10, redbrush, x0-4, r.Height/2-5 );
    }
}

```

```

//create a polygon (with float x, y) from the back-transformed phi3
const double angle_to_arcus = 2 * Math.PI / 360;
double circular_angle_per_crack = 360.0f / cc.cracks.Length;
double angle = 0;
FPoint[] polygon_f = new FPoint[phi3.Length];
float xmin, ymin, xmax, ymax;
xmin = xmax = polygon_f[0].x = cc.start.X;
ymin = ymax = polygon_f[0].y = cc.start.Y;
for ( i=0; i < phi3.Length-1; i++ )
{ angle = angle + angle_to_arcus * (phi3[i] - circular_angle_per_crack);
  x1 = polygon_f[i+1].x = polygon_f[i].x - (float)Math.Cos( angle );
  y1 = polygon_f[i+1].y = polygon_f[i].y - (float)Math.Sin( angle );
  if ( x1 > xmax ) xmax = x1;
  if ( y1 > ymax ) ymax = y1;
  if ( x1 < xmin ) xmin = x1;
  if ( y1 < ymin ) ymin = y1;
}
//The following 6 lines keep the complete polygon inside the window
float w = xmax - xmin + cc.start.X;
float h = ymax - ymin + cc.start.Y;
if (xmin < 0 ) for ( i=0; i < phi3.Length; i++ ) polygon_f[i].x -= xmin;
if (ymin < 0 ) for ( i=0; i < phi3.Length; i++ ) polygon_f[i].y -= ymin;
if (w > xSize) for ( i=0; i < phi3.Length; i++ ) polygon_f[i].x *= xSize/w;
if (h > ySize) for ( i=0; i < phi3.Length; i++ ) polygon_f[i].y *= ySize/h;
//draw the polygon
for ( i=0; i < phi3.Length-1; i++ )
{ x0 = polygon_f[i].x; x1 = polygon_f[i+1].x;
  y0 = polygon_f[i].y; y1 = polygon_f[i+1].y;
  g.DrawLine( yellowpen, x0*dx, y0*dy, x1*dx, y1*dy );
}
//close the polygon
x0 = polygon_f[phi3.Length-1].x; x1 = polygon_f[0].x;
y0 = polygon_f[phi3.Length-1].y; y1 = polygon_f[0].y;
g.DrawLine( yellowpen, x0*dx, y0*dy, x1*dx, y1*dy );
} // end of OnPaint(...)

protected override void OnMouseDown( MouseEventArgs e )
{ p0.X = e.X;
  p0.Y = e.Y;
  pointArray.Add( p0 );
  whitepen = new Pen( Color.White, dx < dy ? dx : dy );
  whitepen.StartCap = whitepen.EndCap = System.Drawing.Drawing2D.LineCap.Round;
}

protected override void OnMouseMove( MouseEventArgs e )
{ if ( e.Button == MouseButton.None ) return;
  p1.X = e.X;
  p1.Y = e.Y;
  //Insert an additional point if the distance is too far
  if ( Math.Abs( p1.X - p0.X ) >= dx || Math.Abs( p1.Y - p0.Y ) >= dy )
    pointArray.Add( new Point( (p1.X + p0.X)/2, (p1.Y + p0.Y)/2 ) );
  pointArray.Add( p1 );
  g = this.CreateGraphics();
  g.DrawLine( whitepen, p0, p1 );
  p0 = p1;
}

private void do_it( object sender, System.EventArgs e )
{ switch( ((Button)sender).Name)
  { case "Threshold"://*****
    if ( ++threshold > 9 ) threshold = 1;
    button[3].Text = "Threshold=" + threshold.ToString();
    cc.cracks = ""; break;
  case "Noise"://*****
    Random random = new Random();
    for ( y=0; y < ySize; y++ )
      for ( x=0; x < xSize; x++ )
        { Int32 noise = random.Next() % 3 - 1;//gives -1 or 0 or +1
          noise += i0[y,x];//add former gray value
          if (noise < 0) i0[y,x] = 0;
          else if (noise > 9) i0[y,x] = 9;
          else i0[y,x] = (Byte)noise;
        }
    cc.cracks = ""; break;
  }
}

```

```

case "Clear"://*****
for ( y=0; y < ySize; y++ )
    for ( x=0; x < xSize; x++ ) i0[y,x] = 0;
backtransform = threshold = 1;
cc.cracks = ""; pointArray.Clear(); break;
case "FourierTransf"://*****
//Digitize*****
for ( i = 0; i < pointArray.Count; i++ )
{ Point vertex = (Point)pointArray[i];
  x = vertex.X / dx;
  y = vertex.Y / dy;
  try { if ( i0[y,x] < 9 ) i0[y,x]++; } catch{};
}
//Crack Code 8*****
//Search for a vertical start crack
Byte leftpix;
for ( y=0; y < ySize; y++ )
    for ( x=0; x < xSize; x++ )
        { if ( x > 0 ) leftpix = i0[y,x-1]; else leftpix = 0;
          if ( leftpix < threshold && i0[y,x] >= threshold )
              { cc.start.X = x; cc.start.Y = y; goto start_crack_found; }
        }
return; //nothing to start with
start_crack_found:
Int32 xmin = x, xmax = x, ymin = y, ymax = ++y;
phil = new ArrayList(); phil.Add(-90);
System.Text.StringBuilder cracks = new System.Text.StringBuilder();
cracks.Append('s'); Char last_crack = 's';
do
{ switch ( last_crack )
  { case 'e': if ( x == xSize ) { phil.Add(-90); goto n; }
            if ( y < ySize && i0[y ,x ] >= threshold) { phil.Add(+90); goto s; }
            if ( i0[y-1,x ] >= threshold) { phil.Add( 0); goto e; }
            { phil.Add(-90); goto n; }

            case 's': if ( y == ySize ) { phil.Add(-90); goto e; }
            if ( x > 0 && i0[y ,x-1] >= threshold) { phil.Add(+90); goto w; }
            if ( i0[y ,x ] >= threshold) { phil.Add( 0); goto s; }
            { phil.Add(-90); goto e; }

            case 'w': if ( x == 0 ) { phil.Add(-90); goto s; }
            if ( y > 0 && i0[y-1,x-1] >= threshold) { phil.Add(+90); goto n; }
            if ( i0[y ,x-1] >= threshold) { phil.Add( 0); goto w; }
            { phil.Add(-90); goto s; }

            case 'n': if ( y == 0 ) { phil.Add(-90); goto w; }
            if ( x < xSize && i0[y-1,x ] >= threshold) { phil.Add(+90); goto e; }
            if ( i0[y-1,x-1] >= threshold) { phil.Add( 0); goto n; }
            { phil.Add(-90); goto w; }

            }
e: last_crack = 'e'; cracks.Append( 'e' ); x++; continue;
s: last_crack = 's'; cracks.Append( 's' ); y++; continue;
w: last_crack = 'w'; cracks.Append( 'w' ); x--; continue;
n: last_crack = 'n'; cracks.Append( 'n' ); y--; continue;
} while ( x != cc.start.X || y != cc.start.Y ); //end of do
cc.cracks = cracks.ToString(); Int32 N = cc.cracks.Length;
//Bad property of function phil: All phil[i] sum up to -360 degrees
phi2 = new float[N]; //Fourier transform needs float values
phi3 = new float[N]; //Back transform creates float values
float circular_angle_per_crack = 360.0f / N;
for ( i=0; i < N; i++ )//Construction of a phi2 with a sum of 0
    phi2[i] = (Int32)phil[i] + circular_angle_per_crack;
//Fourier Transform*****
fr = new float[N/2+1]; fi = new float[N/2+1];
FourierTransform( phi2, fr, fi ); //output = fr, fi
//just for fun: immediate back transform
backtransform = fr.Length;
FourierBackTransform( fr, fi, phi3, backtransform ); //output = phi3
break;
case "Back"://*****
if ( cc.cracks.Length == 0 ) return;
if ( ++backtransform > fr.Length ) backtransform = 0;
FourierBackTransform( fr, fi, phi3, backtransform ); //output = phi3
break;
} // end of switch, end of all cases
Invalidate();
} // end of protected void do_it( object sender, EventArgs e )

```

```

private void FourierTransform( float[] a, float[] fr, float[] fi )
{ int Na = a.Length, Nf = fr.Length;
  float sum_r, sum_i; fr[0] = fi[0] = 0;
  for ( i=0; i < Na; i++ ) fr[0] += a[i]; fr[0] /= Na;//fr[0] is the average = 0.
  double dpi_div_N, x1_dpi_div_N, x2_x1_dpi_div_N;
  dpi_div_N = 2 * Math.PI / Na;
  for ( int x1 = 1; x1 < Nf; x1++ )
  { sum_r = sum_i = 0;
    x1_dpi_div_N = (double)x1 * dpi_div_N;
    for ( int x2 = 0; x2 < Na; x2++ )
    { x2_x1_dpi_div_N = (double)x2 * x1_dpi_div_N;
      sum_r += a[x2] * (float)Math.Cos( -x2_x1_dpi_div_N );
      sum_i += a[x2] * (float)Math.Sin( -x2_x1_dpi_div_N );
    }
    fr[x1] = 2 * sum_r / (float)Na;
    fi[x1] = 2 * sum_i / (float)Na;
  }
  fr[Nf-1] /= 2; fi[Nf-1] /= 2;
} // end of private void FourierTransform(...)

private void FourierBackTransform( float[] fr, float[] fi, float[] a, int Nf )
{ if ( Nf > fr.Length ) return;
  int Na = a.Length;
  Array.Clear( a, 0, a.Length );
  double dpi_div_N, x1_dpi_div_N, x2_x1_dpi_div_N;
  dpi_div_N = 2 * Math.PI / Na;
  for ( int x1 = 0; x1 < Na; x1++ )
  { x1_dpi_div_N = (double)x1 * dpi_div_N;
    for ( int x2 = 0; x2 < Nf; x2++ )
    { x2_x1_dpi_div_N = (double)x2 * x1_dpi_div_N;
      a[x1] += fr[x2] * (float)Math.Cos( x2_x1_dpi_div_N ) -
        fi[x2] * (float)Math.Sin( x2_x1_dpi_div_N );
    }
  }
} // end of private void FourierBackTransform(...)
}

```