Course CVCis: Computer Vision with C#
Chapter C3: The Complete Code of the Polygon Project

Main Menu after start of VS 2008: `File -> New Project... ->`
`Visual Studio installed templates: Windows Forms Application`
`Name: polygon1 -> Location: C:\temp -> Create directory for solution`: switch off `-> OK`
Delete two superfluous files: `Form1.Designer.cs` and `Program.cs`.
Replace the preprogrammed code of `Form1.cs` by:

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

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];
  Brush[] brush    = new Brush[10];
  Brush blackbrush = SystemBrushes.ControlText;
  Brush redbrush   = new SolidBrush( Color.Red  );
  Brush bluebrush  = new SolidBrush( Color.Blue );
  Pen redpen1      = new Pen( Color.Red, 1 );
  Pen redpen5      = new Pen( Color.Red, 5 );
  Pen whitepen;
  Font arial10     = new Font("Arial",10);
  Int32 i, j, x, y, dx, dy;
  Byte threshold = 1;
  Button[] button = new Button[ySize];
  TextBox textboxEpsilon;
  Point p0, p1, p2;
  struct chaincode { public Point start; public String cracks;  }
  chaincode cc;
  Graphics g;
  ArrayList pointArray = new ArrayList();
  ArrayList poly1      = new ArrayList();
  ArrayList poly2      = new ArrayList();

  public Form1()
  { BackColor  = Color.White;
    Text       = "Polygon";
    SetStyle( ControlStyles.ResizeRedraw, true );
    Width  = 800;
    Height = 600;
    for ( i=0; i < 10; i++ )
      brush[i] = new SolidBrush( Color.FromArgb( i*25, i*25, i*25 ) );
    for ( y=0; y < ySize; y++ )
    { 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 = "Polygon";
    button[1].Name = button[1].Text = "eps=0";
    button[2].Name = button[2].Text = "eps=sqrt(2)/2";
    button[3].Name = button[3].Text = "eps=1";
    button[4].Name = button[4].Text = "eps=sqrt(2)";
    button[5].Name = button[5].Text = "eps>=2";
    button[6].Name = button[6].Text = "Clear";
    button[7].Name =                  "Threshold";
    button[8].Name = button[8].Text = "Noise";
    button[7].Text = String.Format("Threshold={0:#}",threshold);
    textboxEpsilon = new TextBox(); Controls.Add(textboxEpsilon);
    textboxEpsilon.Text = "2";
    cc.cracks = "";
  }
```

```
protected override void OnPaint( PaintEventArgs e )
{ g = e.Graphics;
  String s;
  Rectangle r = ClientRectangle;
  dx = r.Width / ( xSize + 2 );
  dy = ( r.Height - FontHeight - 1 ) / ySize;
  for ( y=0; y < ySize; y++ )
  { button[y].Top = y*dy+1;
    button[y].Left = xSize*dx+1;
    if ( y != 5 ) button[y].Width = r.Width - button[y].Left -  2;
    else          button[y].Width = r.Width - button[y].Left - 25;
    button[y].Height = dy - 2;
  }
  textboxEpsilon.Top = button[5].Top + ( button[5].Height - textboxEpsilon.Height ) / 2;
  textboxEpsilon.Left = button[1].Left + button[5].Width + 1;
  textboxEpsilon.Width = button[0].Width - button[5].Width - 2;
  textboxEpsilon.TextAlign = HorizontalAlignment.Center;

  for ( y=0; y < ySize; y++ )
    for ( x=0; x < xSize; x++ )
      g.FillRectangle( brush[i0[y,x]], x*dx, y*dy, dx, dy );
  for ( x=0; x < xSize; x++ ) g.DrawLine( redpen1, x*dx, 0, x*dx, ySize*dy );
  for ( y=0; y < ySize; y++ ) g.DrawLine( redpen1, 0, y*dy, xSize*dx, y*dy );
  if ( poly1.Count == 0 )
  { s = "Draw something -> Polygon -> eps -> Clear -> Next";
    g.DrawString( s, arial10, redbrush, (xSize*dx)/4, 0 );
    return;
  }
  for ( i = 0; i < poly2.Count-1; i++ )
  { p0 = (Point)poly2[i];
    p1 = (Point)poly2[i+1];
    g.DrawLine(redpen5, p0.X*dx, p0.Y*dy, p1.X*dx, p1.Y*dy );
    g.FillRectangle( bluebrush, p0.X*dx-5, p0.Y*dy-5, 11, 11 );
  }
  s = "No of vertices of the closed polygon = ";
  if      ( poly2.Count > 0 ) s = s + poly2.Count.ToString();
  else if ( poly1.Count > 0 ) s = s + poly1.Count.ToString();
  else s = "";
  g.DrawString( s, arial10, redbrush, 0, ySize*dy );
}

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 == MouseButtons.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;
}

protected void do_it( object sender, System.EventArgs e )
{ switch( ( (Button)sender).Name )
  { case "Threshold"://**********************************
      if ( ++threshold > 9 ) threshold = 1;
      button[7].Text = "Threshold=" + threshold.ToString();
      cc.cracks =""; poly2.Clear(); 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 =""; poly2.Clear(); break;
case "Clear"://*************************************
  for ( y=0; y < ySize; y++ )
    for ( x=0; x < xSize; x++ ) i0[y,x] = 0;
  threshold = 1; button[7].Text = "Threshold=1";
  cc.cracks =""; pointArray.Clear(); poly1.Clear(); poly2.Clear(); break;
case "Polygon"://***********************************
  //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{};
  }
  //Chain 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: poly1.Clear();
  p0 = cc.start; poly1.Add(p0);
  y++; p0.Y++;   poly1.Add(p0);
  System.Text.StringBuilder cracks = new System.Text.StringBuilder();
  Char last_crack = 's';
  do
  { switch ( last_crack )
    { case 'e': if ( x == xSize )                            goto n;
                if ( y <  ySize && i0[y  ,x  ] >= threshold) goto s;
                if (               i0[y-1,x  ] >= threshold) goto e; goto n;
      case 's': if ( y == ySize )                            goto e;
                if ( x >  0     && i0[y  ,x-1] >= threshold) goto w;
                if (               i0[y  ,x  ] >= threshold) goto s; goto e;
      case 'w': if ( x == 0 )                                goto s;
                if ( y >  0     && i0[y-1,x-1] >= threshold) goto n;
                if (               i0[y  ,x-1] >= threshold) goto w; goto s;
      case 'n': if ( y == 0 )                                goto w;
                if ( x <  xSize && i0[y-1,x  ] >= threshold) goto e;
                if (               i0[y-1,x-1] >= threshold) goto n; goto w;
    }
    e: last_crack = 'e'; x++; p0.X++; poly1.Add( p0 ); continue;
    s: last_crack = 's'; y++; p0.Y++; poly1.Add( p0 ); continue;
    w: last_crack = 'w'; x--; p0.X--; poly1.Add( p0 ); continue;
    n: last_crack = 'n'; y--; p0.Y--; poly1.Add( p0 ); continue;
  } while ( x != cc.start.X || y != cc.start.Y ); //end of do
  poly2 = (ArrayList)poly1.Clone();
  break;
case "eps=0"://*********************************
  polygon_remove_colinear_vertices( poly1, poly2 );
  break;
case "eps=sqrt(2)/2"://*************************
  polygon_approximation( poly1, poly2, 0.5*Math.Sqrt( 2.0 ) );
  break;
case "eps=1"://*********************************
  polygon_approximation( poly1, poly2, 1.0f );
  break;
case "eps=sqrt(2)"://***************************
  polygon_approximation( poly1, poly2, Math.Sqrt( 2.0 ) );
  break;
```

```
      case "eps>=2"://**********************************
        double epsilon;
        try { epsilon = Convert.ToSingle( textboxEpsilon.Text ); } catch { break; }
        polygon_approximation( poly1, poly2, epsilon );
        break;
    }
    Invalidate();
  }

  void polygon_approximation( ArrayList poly_in, ArrayList poly_out, double epsilon )
  { Int32 F, start=0, stop, dx, dy;
    double length_line_p0p1, distance_of_p2_to_line_p0p1;
    ArrayList poly_help = new ArrayList();
    poly_help.Add( (Point)poly_in[0] );
    for ( stop = start + 2; stop < poly_in.Count; stop++ )
    { p0 = (Point)poly_in[start];
      p1 = (Point)poly_in[stop ];
      dx = p1.X - p0.X;
      dy = p1.Y - p0.Y;
      length_line_p0p1 = Math.Sqrt( dx*dx + dy*dy );
      for ( i = start + 1; i < stop; i++ )
      { p2 = (Point)poly_in[i];
        F  = ( p1.X - p0.X ) * ( p1.Y + p0.Y );//2 * area of trapezoid with p0-p1
        F += ( p2.X - p1.X ) * ( p2.Y + p1.Y );//2 * area of trapezoid with p1-p2
        F += ( p0.X - p2.X ) * ( p0.Y + p2.Y );//2 * area of trapezoid with p2-p0
        //F is now 2 * the area of the triangle, can be negative
        F = Math.Abs( F ); //area of a rectangle with length p0-p1
        //next line computes the height of the rectangle
        //this height is also the height of the triangle
        distance_of_p2_to_line_p0p1 = (double)F / length_line_p0p1;
        if ( distance_of_p2_to_line_p0p1 > epsilon )
        { start = stop - 1;
          poly_help.Add( (Point)poly_in[start] );
          break;
        }
      }
    }
    poly_help.Add( (Point)poly_in[0] );
    if ( poly_help.Count > 3 ) polygon_remove_colinear_vertices( poly_help, poly_out );
    else                       polygon_reduce_to_a_line( poly_in, poly_out );
  }

  void polygon_remove_colinear_vertices( ArrayList poly_in, ArrayList poly_out )
  { poly_out.Clear();
    poly_out.Add((Point)poly_in[0]);
    Int32 dx01, dy01, dx12, dy12, dx02, dy02;
    double d01, d12, d02;
    for ( i=0; i < poly_in.Count-2; i++ )
    { p0 = (Point)poly_in[i  ];
      p1 = (Point)poly_in[i+1];
      p2 = (Point)poly_in[i+2];
      dx01 = p1.X - p0.X; dy01 = p1.Y - p0.Y; d01 = Math.Sqrt( dx01*dx01 + dy01*dy01 );
      dx12 = p2.X - p1.X; dy12 = p2.Y - p1.Y; d12 = Math.Sqrt( dx12*dx12 + dy12*dy12 );
      dx02 = p2.X - p0.X; dy02 = p2.Y - p0.Y; d02 = Math.Sqrt( dx02*dx02 + dy02*dy02 );
      if ( d01 + d12 > d02 ) poly_out.Add(p1);
    }
    poly_out.Add((Point)poly_in[0]);
  }

  void polygon_reduce_to_a_line( ArrayList poly_in, ArrayList poly_out )
  { int imax=1, distance, maxdistance=0;
    p0 = (Point)poly_in[0];
    for ( i=1; i < poly_in.Count; i++ )
    { p1 = (Point)poly_in[i];
      dx = p1.X - p0.X; dy = p1.Y - p0.Y;
      distance = dx*dx + dy*dy;
      if ( distance > maxdistance ) { maxdistance = distance; imax = i; }
    }
    poly_out.Clear();
    poly_out.Add( (Point)poly_in[   0] );
    poly_out.Add( (Point)poly_in[imax] );
    poly_out.Add( (Point)poly_in[   0] );
  }
}
```