

# Course CVCis: Computer Vision with C#

## Chapter C2: The ChainCode Project

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

Theorie zu dieser Übung siehe:

[www.miszalok.de/Lectures/L08\\_ComputerVision/Computer\\_Vision\\_deutsch.htm](http://www.miszalok.de/Lectures/L08_ComputerVision/Computer_Vision_deutsch.htm)

und: [www.miszalok.de/Samples/CV/ChainCode/chaincode\\_kovalev\\_e.htm](http://www.miszalok.de/Samples/CV/ChainCode/chaincode_kovalev_e.htm)

### Projekt chaincode1

- 1) Main Menu nach dem Start von VS 2008: File -> New Project... -> Visual Studio installed templates: Windows Forms Application  
Name: chaincode1 -> Location: C:\temp -> Create directory for solution: ausschalten -> OK  
Es meldet sich Form1.cs[Design].
- 2) Sie müssen zwei überflüssige Files löschen: Form1.Designer.cs und Program.cs.  
Sie erreichen diese Files über das Solution Explorer - chaincode1-Window: Klicken Sie das Pluszeichen vor chaincode1 und dann das Pluszeichen vor Form1.cs.  
Klicken Sie mit der **rechten** Maustaste auf den Ast Program.cs. Es öffnet sich ein Kontextmenu. Sie Klicken auf Delete. Eine Message Box erscheint: 'Program.cs' will be deleted permanently. Sie quittieren mit OK.  
Klicken Sie mit der **rechten** Maustaste auf den Ast Form1.Designer.cs und löschen auch dieses File.
- 3) Klicken Sie mit der **rechten** Maustaste auf das graue Fenster Form1. Es öffnet sich ein kleines Kontextmenü. Klicken Sie auf View Code.  
Sie sehen jetzt den vorprogrammierten Code von Form1.cs. Löschen Sie den gesamten Code vollständig.

### Seitenaufbau, Bildaufbau

Schreiben in das leere Codefenster Form1.cs folgenden Code:

```
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 = 11;
    const Int32 ySize = 12;
    Byte[,] i0 = new Byte[ySize ,xSize ];
    Brush[] brush = new Brush[10];
    Brush blackbrush = SystemBrushes.ControlText;
    Brush bluebrush = new SolidBrush( Color.Blue );
    Pen redpen = new Pen( Color.Red, 5 );
    Font arial10 = new Font( "Arial", 10 );
    Int32 i, x, y, dx, dy;
    Byte threshold = 1;
    Button[] button = new Button[ySize];
    Boolean CC4, CC8;
    Point start = new Point();
    chaincode cc = new chaincode();
    class chaincode
    {
        public Point start;
        public String cracks;
        public Int32 perimeter, area;
    }

    public Form1()
    {
        BackColor = Color.White;
        Text = "Chain Code";
        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 = "Homunculus";
button[1].Name = "Threshold";
button[2].Name = button[2].Text = "Noise";
button[3].Name = button[3].Text = "Clear";
button[4].Name = button[4].Text = "Chain Code 4";
button[5].Name = button[5].Text = "Chain Code 8";
button[1].Text = String.Format( "Threshold={0:#}", threshold );
}
protected override void OnPaint( PaintEventArgs e )
{ Graphics g = e.Graphics;
  Rectangle r = ClientRectangle;
  dx = r.Width / (xSize+2);
  dy = (r.Height - 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;
  }
  for ( y=0; y < ySize; y++ )
    for ( x=0; x < xSize; x++ )
      g.FillRectangle( brush[i0[y,x]], x*dx, y*dy, dx, dy );
  if ( cc.perimeter == 0 ) return;
  x = cc.start.X;
  y = cc.start.Y;
  for ( Int32 ii = 0; ii < cc.perimeter; ii++ )
    switch ( cc.cracks[ii] )
    { case 'e': g.DrawLine(redpen, x*dx, y*dy, (x+1)*dx, y*dy ); x++; break;
      case 's': g.DrawLine(redpen, x*dx, y*dy, x*dx, (y+1)*dy ); y++; break;
      case 'w': g.DrawLine(redpen, x*dx, y*dy, (x-1)*dx, y*dy ); x--; break;
      case 'n': g.DrawLine(redpen, x*dx, y*dy, x*dx, (y-1)*dy ); y--; break;
    }
  g.FillRectangle( bluebrush, x*dx-5, y*dy-5, 11, 11 );
  String s = "(" + cc.start.X.ToString() + "/" +
    cc.start.Y.ToString() + ")" +
    cc.cracks + " P=" +
    cc.perimeter.ToString() + " A=" +
    cc.area.ToString();
  g.DrawString(s, arial10, blackbrush, 0, button[ySize-1].Top + button[ySize-1].Height );
}
protected void do_it( object sender, System.EventArgs e )
{ switch( ((Button)sender).Name)
  { case "Homunculus"://*****
    i0 = new Byte[,]{ {0,0,0,0,0,0,0,0,0,0},
                     {0,0,0,0,9,9,9,0,0,0},
                     {0,0,0,0,9,0,9,0,0,0},
                     {0,0,0,0,9,8,9,0,0,0},
                     {1,0,0,0,0,7,0,0,0,1},
                     {0,2,6,6,6,6,6,6,2,0},
                     {1,0,0,0,5,5,5,0,0,1},
                     {0,0,0,0,4,4,4,0,0,0},
                     {0,0,0,0,3,0,3,0,0,0},
                     {0,0,0,0,2,0,2,0,0,0},
                     {0,0,0,0,1,0,1,0,0,0},
                     {0,0,0,0,1,0,1,0,0,0} };
    break;
  }
  Invalidate();
}
}

```

Klicken Sie Debug -> Start Without Debugging Ctrl F5. Erproben Sie das Programm. Nur der erste Button "Homunculus" tut etwas, alle anderen sind tot.

## Threshold, Noise und Clear

Version2: Beenden Sie Ihr Programm chaincode1.

Schreiben Sie folgende neuen Cases unterhalb des letzten break; von case "Homunculus": im Event Handler protected void do\_it(...):

```

case "Threshold"://*****
    if ( ++threshold > 9 ) threshold = 1;
    button[1].Text = "Threshold=" + threshold.ToString();
    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;
            }
    break;
case "Clear"://*****
    for ( y=0; y < ySize; y++ )
        for ( x=0; x < xSize; x++ ) i0[y,x] = 0;
    threshold = 1; button[1].Text = "Threshold=1";
    cc.cracks = ""; cc.perimeter = 0;
    break;

```

Klicken Sie Debug -> Start Without Debugging Ctrl F5. Erproben Sie Threshold (es ändert sich nur die Buttonbeschriftung), Noise (auch mehrfach drücken) und Clear.

## One Chain Code without Border Handling

Version3: Beenden Sie Ihr Programm chaincode1.

Schreiben Sie folgende neuen Cases unterhalb des letzten break; von case "Clear": im Event Handler protected void do\_it(...):

```
case "Chain Code 4": CC4 = true; CC8 = false; chaincode_finder(); break;
case "Chain Code 8": CC4 = false; CC8 = true; chaincode_finder(); break;
```

Schreiben Sie folgende Funktion unter Invalidate(); und unter die letzte Klammer, die protected void do\_it( object sender, System.EventArgs e) abschließt, aber noch vor der allerletzten Klammer:

```
private void chaincode_finder()
{ for ( x=0; x < xSize; x++ ) i0[0,x] = i0[ySize-1,x] = 0;//clear 1. & last row
  for ( y=0; y < ySize; y++ ) i0[y,0] = i0[y,xSize-1] = 0;//clear 1. & last column
  for ( y=1; y < ySize; y++ )//search for a start crack
    for ( x=1; x < xSize; x++ )
      if ( i0[y,x-1] < threshold && i0[y,x] >= threshold )
        { start.X = x; start.Y = y; goto found; }
return; //no start crack found
found:
System.Text.StringBuilder cracks = new System.Text.StringBuilder();
cc.start = start; cracks.Append( 's' ); cc.area = 0;
x = start.X;
y = start.Y + 1;
Char last_crack = 's';
do
{ if( CC4 ) // 4-connectivity
  switch ( last_crack )
  { case 'e': if ( i0[y-1,x ] < threshold) goto n;
    if ( i0[y ,x ] < threshold) goto e; goto s;
    case 's': if ( i0[y ,x ] < threshold) goto e;
    if ( i0[y ,x-1] < threshold) goto s; goto w;
    case 'w': if ( i0[y ,x-1] < threshold) goto s;
    if ( i0[y-1,x-1] < threshold) goto w; goto n;
    case 'n': if ( i0[y-1,x-1] < threshold) goto w;
    if ( i0[y-1,x ] < threshold) goto n; goto e;
  }
else if ( CC8 ) // 8-connectivity
  switch ( last_crack )
  { case 'e': if ( i0[y ,x ] >= threshold) goto s;
    if ( i0[y-1,x ] >= threshold) goto e; goto n;
    case 's': if ( i0[y ,x-1] >= threshold) goto w;
    if ( i0[y ,x ] >= threshold) goto s; goto e;
    case 'w': if ( i0[y-1,x-1] >= threshold) goto n;
    if ( i0[y ,x-1] >= threshold) goto w; goto s;
    case 'n': if ( i0[y-1,x ] >= threshold) goto e;
    if ( i0[y-1,x-1] >= threshold) goto n; goto w;
  }
  e: last_crack = 'e'; cracks.Append( 'e' ); x++; cc.area += y; continue;
  s: last_crack = 's'; cracks.Append( 's' ); y++; continue;
  w: last_crack = 'w'; cracks.Append( 'w' ); x--; cc.area -= y; continue;
  n: last_crack = 'n'; cracks.Append( 'n' ); y--; continue;
} while ( x != start.X || y != start.Y ); //end of do
cc.cracks = cracks.ToString();
cc.perimeter = cc.cracks.Length;
}
```

Klicken Sie Debug -> Start Without Debugging Ctrl F5. Erproben Sie chaincode1 ohne und mit ausgiebiger Benutzung von Noise.

Verändern Sie den Homunculus so, dass bei 4er und 8er Nachbarschaft verschiedene Ergebnisse herauskommen.

Beachten sie, wie die Bildränder immer schwarz werden.

Erarbeiten Sie sich die Logik der Konturverfolgung zunächst bei der 4er und dann bei der 8er Nachbarschaft.

## One Chain Code with Border Handling

Version4: Beenden Sie Ihr Programm chaincode1.

Ändern sie die Funktion private void chaincode\_finder() bis sie so aussieht:

```
private void chaincode_finder()
{ Byte left;
  for ( y=0; y < ySize; y++ )//search for a start crack
    for ( x=0; x < xSize; x++ )
      { if ( x > 0 ) left = i0[y,x-1]; else left = 0;
        if ( left < threshold && i0[y,x] >= threshold )
          { start.X = x; start.Y = y; goto found; }
      }
  return; //no start crack found
found:
System.Text.StringBuilder cracks = new System.Text.StringBuilder();
cc.start = start; cracks.Append( 's' ); cc.area = 0;
x = start.X;
y = start.Y + 1;
Char last_crack = 's';
do
{ if ( CC4 ) // 4-connectivity
  switch ( last_crack )
    { case 'e': if ( x == xSize || i0[y-1,x ] < threshold) goto n;
              if ( y == ySize || i0[y ,x ] < threshold) goto e; goto s;
          case 's': if ( y == ySize || i0[y ,x ] < threshold) goto e;
                   if ( x == 0 || i0[y ,x-1] < threshold) goto s; goto w;
          case 'w': if ( x == 0 || i0[y ,x-1] < threshold) goto s;
                   if ( y == 0 || i0[y-1,x-1] < threshold) goto w; goto n;
          case 'n': if ( y == 0 || i0[y-1,x-1] < threshold) goto w;
                   if ( x == xSize || i0[y-1,x ] < threshold) goto n; goto e;
        }
  else if ( CC8 ) // 8-connectivity
    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'; cracks.Append( 'e' ); x++; cc.area += y; continue;
    s: last_crack = 's'; cracks.Append( 's' ); y++; continue;
    w: last_crack = 'w'; cracks.Append( 'w' ); x--; cc.area -= y; continue;
    n: last_crack = 'n'; cracks.Append( 'n' ); y--; continue;
  } while ( x != start.X || y != start.Y ); //end of do
cc.cracks = cracks.ToString();
cc.perimeter = cc.cracks.Length;
}
```

Klicken Sie Debug -> Start Without Debugging Ctrl F5. Erproben Sie chaincode1 ohne und mit ausgiebiger Benutzung von Noise.

Erarbeiten Sie sich die Logik der Behandlung der Bildränder zunächst bei der 4er und dann bei der 8er Nachbarschaft. Versuchen Sie, die area-Berechnung zu verstehen.

## All Chain Codes with Border Handling

Version5: Beenden Sie Ihr Programm chaincode1.

Fügen Sie folgende Deklarationen im Kopf von `public class Form1 : System.Windows.Forms.Form` zusätzlich ein:

```
Byte[,] clv      = new Byte[ySize  ,xSize+1];
ArrayList all_chaincodes = new ArrayList();
```

Ändern Sie die Funktion `protected override void OnPaint(PaintEventArgs e)` bis sie so aussieht:

```
protected override void OnPaint(PaintEventArgs e)
{ Graphics g = e.Graphics;
  Rectangle r = ClientRectangle;
  dx = r.Width / (xSize+2);
  dy = (r.Height - all_chaincodes.Count * 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;
  }
  Int32 textfieldY = button[ySize-1].Top + button[ySize-1].Height + 2;

  for ( y=0; y < ySize; y++ )
    for ( x=0; x < xSize; x++ )
      g.FillRectangle( brush[i0[y,x]], x*dx, y*dy, dx, dy );
  if ( all_chaincodes.Count == 0 ) return;
  for ( i=0; i < all_chaincodes.Count; i++ )
  { chaincode c = (chaincode)all_chaincodes[i];
    x = c.start.X;
    y = c.start.Y;
    for ( Int32 ii = 0; ii < c.perimeter; ii++ )
      switch ( c.cracks[ii] )
      { case 'e': g.DrawLine(redpen, x*dx, y*dy, (x+1)*dx, y*dy ); x++; break;
        case 's': g.DrawLine(redpen, x*dx, y*dy, x*dx, (y+1)*dy ); y++; break;
        case 'w': g.DrawLine(redpen, x*dx, y*dy, (x-1)*dx, y*dy ); x--; break;
        case 'n': g.DrawLine(redpen, x*dx, y*dy, x*dx, (y-1)*dy ); y--; break;
      }
    g.FillRectangle( bluebrush, x*dx-5, y*dy-5, 11, 11 );
    String s = "(" + c.start.X.ToString() + "/" +
              c.start.Y.ToString() + ")" +
              c.cracks + " P=" +
              c.perimeter.ToString() + " A=" +
              c.area.ToString();
    g.DrawString(s, arial10, blackbrush, 0, textfieldY );
    textfieldY += FontHeight;
  }
}
```

Ändern sie die Funktion `private void chaincode_finder()` bis sie so aussieht:

```
private void chaincode_finder()
{ all_chaincodes.Clear();
  for ( y=0; y < ySize; y++ )
    for ( x=0; x <= xSize; x++ ) clv[y,x] = 0;
  start.X = start.Y = 0;
  for ( Byte cc_no = 1; start_crack_search(); cc_no++ )
  { chaincode cc = new chaincode();
    System.Text.StringBuilder cracks = new System.Text.StringBuilder();
    cc.start = start; cracks.Append( 's' ); cc.area = 0;
    x = start.X;
    y = start.Y + 1;
    Char last_crack = 's';
    do
    { if ( CC4 ) // 4-connectivity
      switch ( last_crack )
      { case 'e': if ( x == xSize || i0[y-1,x ] < threshold) goto n;
              if ( y == ySize || i0[y ,x ] < threshold) goto e; goto s;
        case 's': if ( y == ySize || i0[y ,x ] < threshold) goto e;
              if ( x == 0 || i0[y ,x-1] < threshold) goto s; goto w;
        case 'w': if ( x == 0 || i0[y ,x-1] < threshold) goto s;
              if ( y == 0 || i0[y-1,x-1] < threshold) goto w; goto n;
        case 'n': if ( y == 0 || i0[y-1,x-1] < threshold) goto w;
              if ( x == xSize || i0[y-1,x ] < threshold) goto n; goto e;
      }
    } else if ( CC8 ) // 8-connectivity
      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'; cracks.Append( 'e' ); x++; cc.area += y; continue;
    s: last_crack = 's'; cracks.Append( 's' ); y++; clv[y-1,x] = cc_no; continue;
    w: last_crack = 'w'; cracks.Append( 'w' ); x--; cc.area -= y; continue;
    n: last_crack = 'n'; cracks.Append( 'n' ); y--; clv[y ,x] = cc_no; continue;
  } while ( x != start.X || y != start.Y ); //end of do
  cc.cracks = cracks.ToString();
  cc.perimeter = cc.cracks.Length;
  all_chaincodes.Add( cc );
} // end of for
}
```

Schreiben Sie eine zusätzliche Funktion `private Boolean start_crack_search()` unter die letzte Klammer von `private void chaincode_finder()`, aber noch vor die allerletzte Klammer, die das Programm abschließt:

```
private Boolean start_crack_search()
{ Byte left;
  for ( y=start.Y; y < ySize; y++ )
    for ( x=0; x < xSize; x++ )
    { if ( x > 0 ) left = i0[y,x-1]; else left = 0;
      if ( left < threshold && i0[y,x] >= threshold && clv[y,x] == 0 )
        { start.X = x; start.Y = y; clv[y,x] = 1; return true; }
    }
  return false;
}
```

Fügen Sie in `protected void do_it(...)` im case "Homunculus", im case "Threshold", im case "Noise" und im case "Clear" (insgesamt 4 mal) jeweils vor den abschließenden `break`; noch folgenden Befehl ein:

```
all_chaincodes.Clear();
```

Klicken Sie Debug -> Start Without Debugging Ctrl F5. Erproben Sie `chaincode1`. Lernen Sie, an Hand der Lage des Startpunktes und des Vorzeichens von Area Außenkonturen von Innenkonturen (Löchern) zu unterscheiden. Ordnen Sie die ausgegebenen Textzeilen den zugehörigen Gebieten zu. Versuchen Sie zu verstehen, warum die durchlaufenen vertikalen Cracks in die `clv`-Matrix gespeichert werden und warum `start_crack_search()` solche Cracks ausschließt.

## Chain Code Animation

Version6: Beenden Sie Ihr Programm chaincode1.

Fügen Sie folgende Deklarationen im Kopf von `public class Form1 : System.Windows.Forms.Form` zusätzlich ein:

```
Pen pinkpen = new Pen( Color.FromArgb(255,128,128), 3 );
```

Fügen Sie die folgenden 2 Befehle am Ende des Konstruktors `public Form1()` an:

```
redpen.EndCap = System.Drawing.Drawing2D.LineCap.ArrowAnchor;
pinkpen.EndCap = System.Drawing.Drawing2D.LineCap.ArrowAnchor;
```

Suchen Sie in der Funktion `private void chaincode_finder()` folgende 3 Befehle:

```
...
Char last_crack = 's';
do
{ if ( CC4 ) // 4-connectivity
...

```

Ersetzen Sie diese 3 Befehle durch:

```
Char last_crack = 's';
Graphics g = this.CreateGraphics();
do
{ g.DrawLine( pinkpen, x*dx, y*dy, x*dx+dx/4, y*dy );
  g.DrawLine( pinkpen, x*dx, y*dy, x*dx, y*dy+dy/4 );
  g.DrawLine( pinkpen, x*dx, y*dy, x*dx-dx/4, y*dy );
  g.DrawLine( pinkpen, x*dx, y*dy, x*dx, y*dy-dy/4 );
  switch ( last_crack )
  { case 'e': g.DrawLine( redpen, (x-1)*dx, y*dy, x*dx, y*dy ); break;
    case 's': g.DrawLine( redpen, x*dx, (y-1)*dy, x*dx, y*dy ); break;
    case 'w': g.DrawLine( redpen, (x+1)*dx, y*dy, x*dx, y*dy ); break;
    case 'n': g.DrawLine( redpen, x*dx, (y+1)*dy, x*dx, y*dy ); break;
  }
  Int64 ticks = DateTime.Now.Ticks + 2000000;
  do {} while ( ticks > DateTime.Now.Ticks );
  if ( CC4 ) // 4-connectivity

```

Malen Sie noch den letzten Crack für jedes Gebiet, indem Sie am Ende der Funktion `private void chaincode_finder()` direkt hinter das Ende des `do`-Loops, also unter der Zeile

```
} while ( x != start.X || y != start.Y ); //end of do
```

noch folgendes zusätzliche `switch`-Statement anfügen:

```
switch ( last_crack )
{ case 'e': g.DrawLine( redpen, (x-1)*dx, y*dy, x*dx, y*dy ); break;
  case 'w': g.DrawLine( redpen, (x+1)*dx, y*dy, x*dx, y*dy ); break;
}

```

Klicken Sie Debug -> Start Without Debugging Ctrl F5. Erproben Sie chaincode1.

Verlangsamen Sie die Animation, indem Sie die Anzahl der abzuwartenden Uhr-Ticks von 2 Mio auf 4, 8, 16 Mio anheben (1 Tick = 100 Nanosekunden).

## Weitere Aufgaben

Erzeugen Sie Varianten.

Beispiele:

- (1) Ersetzen Sie die 4 Richtungen durch Zahlen des Datentyps `Byte`: `e = 0, s = 1, w = 2, n = 3`.
- (2) Zeichnen Sie die Chain Codes von Version 5 in verschiedene Farben, die sich zyklisch wiederholen, falls es mehr Gebiete als Farben gibt.
- (3) Füllen Sie die Gebiete mit diesen Farben unter Verwendung der `c1v`-Matrix.
- (4) Ersetzen Sie die Bildrandbehandlungen von Version 4/5 durch `try - catch` Exception Handling.

Sie finden eine wesentlich kompaktere Version des Chain Code Algorithmus (allerdings nicht so leicht zu verstehen und mit etwas längerer Laufzeit) unter:

[www.miszalok.de/Samples/CV/ChainCode/chain\\_code.htm](http://www.miszalok.de/Samples/CV/ChainCode/chain_code.htm)