

Course DICis: The DICOM Medical Image Format

Chapter C3: The DICOM Image Project

Copyright © by V. Miszalok, last update: 09-03-2004

- ✚ [Projekt image1](#)
- ✚ [Dump and Tag Search](#)
- ✚ [Image and Histogram](#)
- ✚ [Weitere Aufgaben](#)

Diese Übung liest beim Programmstart automatisch eine Textdatei `C:\temp\dicom.dic` mit dem vollständigen und aktuellen DICOM Dictionary 2001 (bis supplement 59). Jede Zeile der Datei enthält einen Eintrag mit 5 Feldern: `Tag`, `VR`, `Name`, `VM`, `Version`. Die letzten 2 Felder werden von `tags1` ignoriert und abgeschnitten.

Man kann nunmehr unbekannte Dateien jeder Art einlesen. Das Programm listet maximal 16x16 Bytes in 3 parallelen Typkonvertierungen: Bytes hexadezimal, 16-Bit-Integer und Character.

Falls es sich um Dicom-Bilder handelt, listet es sämtliche Felder des DICOM Headers und interpretiert deren Bedeutung an Hand des DICOM Dictionary.

Weiterhin liest es das Rasterbild, wandelt es (falls unkomprimiert) um in ein 32-Bit Bitmap-Objekt und stellt dieses unterhalb der Ausgabe des Headers dar. Falls mehrere Bilder im File sind (z.B. AVI-Filmsequenz) sehen Sie allerdings nur das erste. Unter dem Bild erscheint noch das Histogramm (nur bis 256 Grauwerte) in logarithmischem Maßstab.

Wichtig: DICOM Dictionary ASCII-Textdatei hier laden: [dicom.dic 86 kB](#) und nach `C:\temp\dicom.dic` kopieren !

Vorschlag: Kopieren Sie folgende Bilder in eine neue Directory `C:\temp\Images`.

Beispielbild: [Ultrasound 303 kB](#)

Beispielbild: [Ultrasound 303 kB](#)

Beispielbild: [NMR 137 kB](#)

Beispielbild: [CT 514 kB](#)

Beispielbild: [Uro 570 kB](#)

Projekt image1

Microsoft Visual Studio.NET starten

File - New - Project - Project Types: Visual C# Projects, Templates: Windows Application

Name: image1

Location: `C:\temp`

Button OK unten Mitte klicken.

Klicken Sie mit der **rechten** Maustaste auf des Innere von Form1.

Es öffnet sich ein kleines Kontextmenü. Klicken Sie auf View Code.

Sie sehen jetzt den vorprogrammierten Code von VisualStudio.NET. Löschen Sie den gesamten Code vollständig.

Dump and Tag Search

Schreiben Sie in das leere Fenster folgenden Code, der 99 % identisch ist (bis auf drei neue Zeilen im Kopf von `Form1` und drei neue Zeilen im Konstruktor von `Form1`) mit dem von Chapter 2: The DICOM Tag Project.

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

```

public class Form1 : Form
{
    byte[] mybuffer;//space for Dicom file
    String dic; //space for dictionary filie
    String DicText; //Dicom dictionary additional text;
    StringBuilder s = new StringBuilder();
    TextBox TB = new TextBox();
    Bitmap myImage;
    int[] Histogram;
    int ImageHeight=0, ImageWidth=0, ImageBitsStored=0;
    static void Main() { Application.Run( new Form1() ) ; }

    public Form1()
    {
        Text = "DicomHeader";
        MenuItem miOpen = new MenuItem("&Open", new EventHandler( MenuFileOpenOnClick ) );
        MenuItem miExit = new MenuItem("&Exit", new EventHandler( MenuFileExitOnClick ) );
        MenuItem miFile = new MenuItem("&File", new MenuItem[] { miOpen, miExit } );
        Menu = new MainMenu( new MenuItem[] { miFile } );
        ClientSize = new Size( 800, 800 );
        AutoScroll = true;
        AutoScrollMinSize = new Size( 1200, 1400 );
        TB.Size = new Size( ClientSize.Width, ClientSize.Height / 2 );
        TB.Font = new Font( "Courier New", 8 );
        TB.Multiline = true;
        TB.WordWrap = false;
        TB.ScrollBars = ScrollBars.Both;
        Controls.Add( TB );
        try { StreamReader sr = new StreamReader( "C:\\temp\\dicom.dic" );
            dic = sr.ReadToEnd();
            sr.Close(); }
        catch { MessageBox.Show( "Cannot find C:\\temp\\dicom.dic." ); }
    }

    void MenuFileOpenOnClick( object obj, EventArgs ea )
    {
        OpenFileDialog dlg = new OpenFileDialog();
        if ( dlg.ShowDialog() == DialogResult.OK )
        {
            FileStream fs = new FileStream( dlg.FileName, FileMode.Open, FileAccess.Read );
            mybuffer = new Byte[fs.Length];
            fs.Read( mybuffer, 0, (int)fs.Length );
            DumpIt();
        }
    }

    void MenuFileExitOnClick( object obj, EventArgs ea )
    {
        Close(); }

    private void DumpIt()
    {
        s.Length = 0;
        int x, y;
        for ( y=0; y < 64; y++ ) // print 64 lines
        {
            for ( x=0; x < 16; x++ ) // 16 bytes as hexadecimal in each line
                s.Append( String.Format( "{0:X2} ", mybuffer[y*16+x] ) );
            s.Append( " " );
            for ( x=0; x < 16; x+=2 ) // 16 bytes as 8 Int16 in each line
            {
                int figure = mybuffer[y*16+x] + 16*mybuffer[y*16+x+1];
                s.Append( String.Format( "{0:D4} ", figure ) );
            }
            s.Append( " " );
            for ( x=0; x < 16; x++ ) // 16 bytes as 16 characters in each line
            {
                char c = Convert.ToChar( mybuffer[y*16+x] );
                if ( Char.IsControl(c) ) s.Append( "." ); else s.Append( c.ToString() );
            }
            s.Append( "\r\n" );
        }
        s.Append( "\r\n" );
    }
}

```

```

int i, bytecount = 0;
do
{
    int tlen1 = (int)mybuffer[bytecount++];
    int tlen2 = (int)mybuffer[bytecount++];
    int tlen3 = (int)mybuffer[bytecount++];
    int tlen4 = (int)mybuffer[bytecount++];
    String tag = String.Format ("({0:X4},{1:X4})", tlen1 + 256*tlen2, tlen3 + 256*tlen4 );
    int halftag = tlen1 + 256*tlen2;
    tlen1 = (int)mybuffer[bytecount++];
    tlen2 = (int)mybuffer[bytecount++];
    tlen3 = (int)mybuffer[bytecount++];
    tlen4 = (int)mybuffer[bytecount++];
    int tlen = tlen1 + 256*tlen2 + 256*256*tlen3 + 256*256*256*tlen4;
    s.Append( tag + ' ' + String.Format("{0:D8} ", tlen ) + " " );
    //output the first 48 bytes of any arbitrary content
    for ( i=0; i < Math.Min( tlen, 48 ); i++ )
    {
        char c = Convert.ToChar( mybuffer[bytecount + i] );
        if ( Char.IsControl( c ) ) s.Append( '.' ); else s.Append( c.ToString() );
    }
    for ( ; i < 48; i++ ) s.Append( ' ' );
    String VR = FindTagInDictionary( tag ); //function: look into the dictionary
    String Value = GetValue( VR, bytecount ); //function: get US, SS, UL, SL data types
    s.Append( ' ' + VR + Value + "\r\n" );
    bytecount += tlen;
} while ( bytecount < mybuffer.Length );
TB.Text = s.ToString();
Invalidate();
}

private String FindTagInDictionary( String tag )
{
    //uneven tags are never listed in the dictionary
    if ( Convert.ToInt16( tag[4] ) % 2 != 0 ) return "?? PrivateTag";
    int n1, n2, n3, n4;
    n1 = dic.IndexOf( tag ); //find the even tag in the dictionary
    if ( n1 < 0 ) return String.Empty; //not found
    int i = 11; // jump over the tag in order to search for the following tag
    do
    {
        n2 = dic.IndexOf( '(', n1 + i++ ); //leading clause ?
        while ( dic[n2+5] != ',' || dic[n2+10] != ')' ); //middle comma and end clause ?
        if ( n2 < 0 ) n2 = n1 + 13; //there is no following tag, suppress the rest
        DicText = dic.Substring( n1, n2-n1 ); //cut out one line from the dictionary
        DicText = DicText.Substring( 12, DicText.Length-12 ); //cut off anything in front of VR
        String VR = DicText.Substring( 0, 2 ); //2 bytes of the value representation
        n3 = DicText.IndexOf( '\t', 0 ); //1. tab inside the dictionary line
        n4 = DicText.IndexOf( '\t', n3+1 ); //2. tab
        String Description = DicText.Substring( n3+1, n4-n3 ); //Cut out between tabs
        return VR + ' ' + Description;
    }
}

private String GetValue( String VR, int bytecount )
//This function extracts values from Dicom header tags carrying decimal data of type:
//unsigned short, signed short, unsigned long, signed long.
{
    if ( VR.Length < 2 || bytecount <= 0 ) return String.Empty; //bad parameter
    if ( VR[0] == '?' ) return String.Empty; //private tag
    if ( VR[0] == 'U' && VR[1] == 'S' || VR[0] == 'S' && VR[1] == 'S' ) //16-bit integers ?
    {
        int number = (int)mybuffer[bytecount ] + 256*(int)mybuffer[bytecount+1];
        return( " = " + String.Format("{0}", number ) );
    }
    if ( VR[0] == 'U' && VR[1] == 'L' || VR[0] == 'S' && VR[1] == 'L' ) //32-bit integers ?
    {
        int number = (int)mybuffer[bytecount ] + 256*(int)mybuffer[bytecount+1] +
            256*256*(int)mybuffer[bytecount+2] + 256*256*256*(int)mybuffer[bytecount+3];
        return( " = " + String.Format("{0}", number ) );
    }
    return String.Empty;
}
}
}

```

Übersetzen, linken und starten Sie mit Start Without Debugging Ctrl F5.

Falls eine Fehlermeldung erscheint, steht C:\temp\dicom.dic nicht an der richtigen Stelle. Holen Sie das nach und erproben Sie tags1 erneut.

Image and Histogram

Gegen Ende der Funktion `private void DumpIt()` fügen Sie **unter** der Zeile `s.Append(' ' + VR + Value + "\r\n");`, aber noch **vor** `bytecount += tlen;` folgende zwei Funktionsaufrufe ein:

```
GetImage_H_W_Bits ( tag, bytecount ); //function: find and get image descriptors
GetImage_and_Histo( tag, bytecount ); //function: find pixel data and compose image
```

Hinter die Funktion `private String GetValue(String VR, int bytecount)`, aber noch vor der letzten Klammer, die das Programm abschließt, fügen Sie die beiden neuen Funktionen ein:

```
private void GetImage_H_W_Bits( String tag, int bytecount )
{ int number = (int)mybuffer[bytecount ] + 256*(int)mybuffer[bytecount+1];
  if ( tag == "(0028,0010)" ) { ImageHeight      = number; return; }
  if ( tag == "(0028,0011)" ) { ImageWidth       = number; return; }
  if ( tag == "(0028,0101)" )   ImageBitsStored = number; return;
}

private void GetImage_and_Histo( String tag, int bytecount )
{ if ( tag != "(7FE0,0010)" ) return; //VR = ox = pixel data ?
  myImage = new Bitmap( ImageWidth, ImageHeight,
    System.Drawing.Imaging.PixelFormat.Format32bppRgb );
  Histogram = new int[256];
  int gray = 0;
  //read all pixels and put them into the histogram and into bitmap object
  if ( ImageBitsStored <= 8 )
  { for ( int y=0; y < ImageHeight; y++ )
    for ( int x=0; x < ImageWidth; x++ )
    { gray = mybuffer[bytecount++];
      Histogram[gray]++;
      Color mycolor = Color.FromArgb( gray, gray, gray );
      myImage.SetPixel( x, y, mycolor );
    }
  }
  else //if more than one byte per pixel
  { for ( int y=0; y < ImageHeight; y++ )
    for ( int x=0; x < ImageWidth; x++ )
    { gray = mybuffer[bytecount]; //1st byte
      if ( mybuffer[bytecount+1] == 0 ) Histogram[gray]++; //nothing in the 2nd byte
      else Histogram[ 255]++; //something in the 2nd byte
      gray += 256 * mybuffer[bytecount+1]; //1st and 2nd byte together
      switch ( ImageBitsStored ) //press everything in one byte
      { case 10: gray /= 4; break;
        case 12: gray /= 16; break;
        case 16: gray /= 256; break;
      }
      bytecount += 2;
      Color mycolor = Color.FromArgb( gray, gray, gray );
      myImage.SetPixel( x, y, mycolor );
    }
  }
  // map the histogram values to 10 * logarithms in base 2
  for ( int i=0; i < 256; i++ )
    if ( Histogram[i] <= 1 ) Histogram[i] = 0;
    else Histogram[i] = Convert.ToInt32( 10.0 * Math.Log( (double)Histogram[i], 2 ) );
  return;
}
```

Nun müssen Sie noch die `OnPaint`-Funktion programmieren, damit Bild und Histogramm angezeigt werden. Schreiben Sie `OnPaint` am besten hinter den Eventhandler `void MenuFileExitOnClick(object obj, EventArgs ea)`, aber noch vor die Funktion `private void DumpIt()`.

```
protected override void OnPaint( PaintEventArgs pea )
{
    TB.Text = s.ToString();
    Graphics g = pea.Graphics;
    Point pt = AutoScrollPosition;
    int verticalPosition = TB.Height + 10;
    try { g.DrawImage( myImage, pt.X, pt.Y + verticalPosition ); } catch { return; }
    verticalPosition += ImageHeight + 10;
    g.DrawRectangle( new Pen( Color.Red, 3 ), pt.X, pt.Y + verticalPosition, 256, 200 );
    for ( int x=0; x < Histogram.Length; x++ )
    {
        int y = Histogram[x];
        g.DrawLine( new Pen( Color.Black ), pt.X + x, pt.Y + verticalPosition + 200,
                    pt.X + x, pt.Y + verticalPosition + 200 - y );
    }
}
```

Übersetzen, linken und starten Sie mit Start Without Debugging Ctrl F5.

Wenn kein Bild erscheint, dann kann das mehrere Ursachen haben:

- (1) Es ist kein DICOM-File.
- (2) Das DICOM-File enthält kein Bild.
- (3) Das Bild ist komprimiert (GIF, JPEG, user compress etc.).
- (4) Die Pixel haben weder 8 noch 16 Bit - Format (32 Bit, user defined etc.).
- (5) Das Bild ist leer (Sequenzen fangen manchmal mit Leerbildern an.).

Wenn das Bild nur 2 Grauwerte hat oder sonst wie ganz verfremdet aussieht, dann kann das folgende Ursachen haben:

- (1) Es gibt eine Palette, aber `image1` ignoriert Paletten.
- (2) Es handelt sich um Rohbilder der physikalischen Messwerte, die (noch) nicht sinnvoll quantisiert sind.

Bedenken Sie, dass die vertikale Häufigkeits-Achse des Histogramms logarithmischen Maßstab ($\text{Id} = \log_2$) hat. Doppelte Höhe H bedeutet folglich nicht $2 \cdot H$ sondern $H \cdot H$.

Weitere Aufgaben

(1) Machen Sie das Histogramm linear durch die Änderung der Zeile

```
else Histogramm[i] = Convert.ToInt32( 10.0 * Math.Log( (double)Histogramm[i], 2 ) );
in
```

`else Histogramm[i] = Convert.ToInt32(0.01 * Histogramm[i]);` und beobachten Sie die Änderung der Ausgabe.

(2) Vergrößern Sie automatisch die Ausgabe kleiner Bilder.

(3) Laden und speichern Sie Beispieldaten von:

<http://www.xray.hmc.psu.edu/physresources/dicom/sampleimg.htm>.

(4) Klicken sie auf `help` in der Menüleiste von VS.NET und suchen sie Information zu den ihnen unbekanntem Sprachelementen.

(5) Erfinden und erproben sie neue Varianten des Programms (in Form von neuen Projekten `image2`, `image3` usw. nach obigem Muster).