Multimedia Technologies

Project Report

1. Members

序号	学号	专业班级	姓名	性别	分工
1	3062211102	软件工程 0604	刘颖	男	Design, UI, Differential Coding, Supporting Classes
2	3062211117		堵斌	男	DCT, quantization
3	3062211122		赵武	男	Huffman Coding

2. Project Introduction

2.1 Overall Information

Da Vinci Code

Image compression application

Features

- Implemented Algorithms: Differential, DCT & quantization, Huffman...
- **Compression Ratio Display**
- Uncompressed / Compressed / **Difference** Mode Switch
- **Easy examination of Distortion** of Source/Compressed Image
- Basic classes for image manipulation, pixel format conversion, codecs
- **HSV-RGB-YCbCr** Color Model for display
- **Channel Selection** (any mix of 1~3 color channels)
- **Easy registration of codec**; **auto detection** of coded stream
- **MVC** Architecture (or Frame - View - Document)

Development Environment

Visual Studio 2005, .NET Framework 2.0, Window XP SP2

System Requirements

.NET Framework 2.0, 24-bit True Color Display (or higher)

2.2 Design

1. Model (Document) + View + Controller (Frame):

Document: holds image data (Compressed / Uncompressed / Difference).
View: renders image according to supplied options.
Frame: Accepts user commands, controls the behavior of View, loads and saves the current Document.

2. FileSplitter - IImageCodec

IImageCodec defines codec operations: encode/decode, get content type id.
FileSplitter provides image file load/save capabilities, runtime registration / unregistration of codecs, content type auto-detection using registered codecs.

3. DVCImage, PixelType, PixelTypeConverter

DVCImage: The Internal Image Data Format, multiple of 16x16 block.
PixelType: A combination of bit allocation method and color model.
PixelTypeConverter: RGB<->YCbCr Conversions are implemented.

4. File format: DVC header (4 bytes) + content type id (8 bytes) + code stream.

3. Technical Details

3.1 Theoretical Basis

1. Discrete Cosine Transform

The Discrete Cosine Transform maps a discrete signal from the temporal / spatial domain to the frequency domain. Mathematically stated, it transforms a vector from one linear space to another. Consecutively performing DCT and IDCT results in the original signal and no loss is introduced.

Like any other transform coding, DCT transforms vectors and aims at reducing the correlation of the components of each vector. We can drop the high frequency components and still retain acceptable quality.

In this project, 2D-DCT with 8*8 block size is used, which is defined by:

$$\boxed{F(u,v)=\frac{C(u)C(v)}{4}\sum_{i=0}^{7}\cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}f(i,j)}$$

And the corresponding IDCT is:

$$\boxed{\tilde{f}(i,j)=\sum_{u=0}^{7}\sum_{v=0}^{7}\frac{C(u)C(v)}{4}\cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}F(u,v)}$$

2. Differential Coding

Differential coding takes advantage of the fact that many images look “continuous”, that is, the color values of neighboring pixels are close. Coded images usually have a histogram with a very sharp peak at nearby zero. This allows entropy coding works to produce much higher compression ratios.

In this project, this formula is simply used:

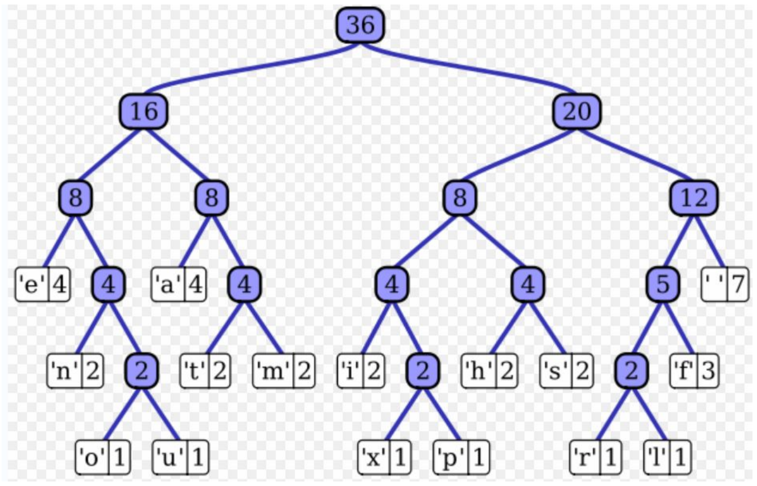
$$\boxed{\mathrm{d}I(x, y) = I(x, y) - I(x - 1, y)}$$

3. Huffman Coding

Huffman coding encodes source symbols with variable-length code words. Code table is based on the estimated probability of the symbols. Frequently used symbols are assigned shorter code words, resulting in shorter average code length. It has been proven that:

$\boxed{\eta \le l < \eta + 1}$, where l denotes the average code length.

A Huffman tree is needed to generate the code table, where symbols are always placed at leaf nodes. Nodes in higher layer always have higher probabilities. The path from root to a leaf is the code of that symbol.



3.2 Algorithms

1. 2D-DCT & 2D-IDCT

Let M_{ij} = C(i) cos (2j +1)iπ/4. Let $\mathbf{Q_C}$, $\mathbf{Q_Y}$ denotes the quantization matrix.

Note $\mathbf{M}^{-1} = \mathbf{M}^t$ since **M** is orthogonal.

Encode:

$\mathbf{Y = M\ y\ M^t}$, $\mathbf{Cb = M\ cb\ M^t}$, $\mathbf{Y = M\ cr\ M^t}$

$Y'_{ij} = Y_{ij} / Q_{ij}$, $Cb'_{ij} = Cb_{ij} / Q_{ij}$, $Cb'_{ij} = Y_{ij} / Q_{ij}$ (Quantization)

Decode:

$Y_{ij} = Y'_{ij} \cdot Q_{ij}$, $Cb_{ij} = Cb'_{ij} \cdot Q_{ij}$, $Cr_{ij} = Cr'_{ij} \cdot Q_{ij}$ (Unquantization)

$\mathbf{y = M^t\ Y\ M}$, $\mathbf{cb = M^t\ Cb\ M}$, $\mathbf{cr = M^t\ Cr\ M}$

2. Differential Coding

Encode

```
for (y = 0; y < height; y++) {
    dI(0, y) = I(0, y);
    for (x = 1; x < width; x++) {
        dI(x, y) = I(x, y) – I(x - 1, y);
    }
}
```

Decode

```
for (y = 0; y < height; y++) {
    dI(0, y) = I(0, y);
    for (x = 1; x < width; x++) {
        dI(x, y) = I(x, y) – I(x - 1, y);
    }
}
```

3. Huffman Coding

Encode (Pseudo Code)

```
//  class Node {frequency, symbol, left, right }
//  class Code {symbol, codeBits}
Node[] nodes = countSymbols(source).sortByFrequency();
queue1 = emptyqueue, queue2 = emptyqueue;
queue1.enqueue(nodes);

//  Build Huffman tree.
while (count(queue1) + count(queue2) >= 2) {
    new_node = combine(dequeLeastTwoFromTwoQueues(queue1, queue2));
    queue2.enqueue(new_node);
}
rootNode = dequeueTheOnlyOneFromTwoQueue(queue1, queue2)
Code[] codeTable = traverse(rootNode);

write(sourceLength, codeTable);
write(translate(source, codeTable));
```

```
Decode (Pseudo Code)

read(sourceLength, codeTable);
rootNode = rebuildTreeFromTable();
for (i = 0; i < sourceLength; i++) {
    node = rootNode;
    do {
        node = readBit() == 0 ? node.left : node.right;
    } while (node is not leaf);
    write(symbol);
}
```

3.3 Important Routines

Bitmap.LockBits (System.Drawing)

It temporarily locks a bitmap in memory. This allows pointer access in unsafe context and speeds up rendering. Otherwise, Bitmap.SetPixel method must be called once per pixel, which is really heavy overhead.

Marshal.Copy (System.InteropServices)

Copies data between managed and unmanaged memory.

MemoryStream (System.IO)

Allows memory-based variable-length streams. It is used to hold compressed data. Images can be compressed and then decompressed in memory for observing distortion, instead of file saves and reloads.

View.updateView (this project)

Selects render routine according to current view options. It is this routine that enables easy switch among different color models, different channels, color / monochrome, compressed / uncompressed / difference.

FileSplitter.load (System.IO)

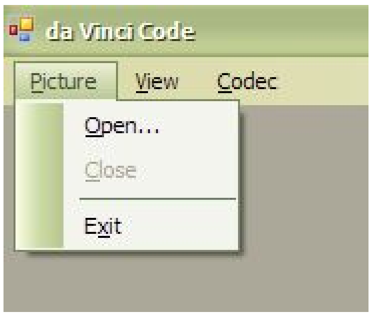
FileSplitter allows any codec to register to it at runtime. Any content type supported by a registered codec will be identified and sent to that codec for decoding. This routine accomplishes content/codec auto-detection.

4. Experiment Results

4.1 User Interface

1. Load an image file.

Click menu item [Picture->Open].



Choose an image.

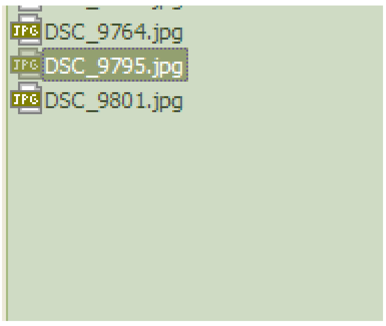
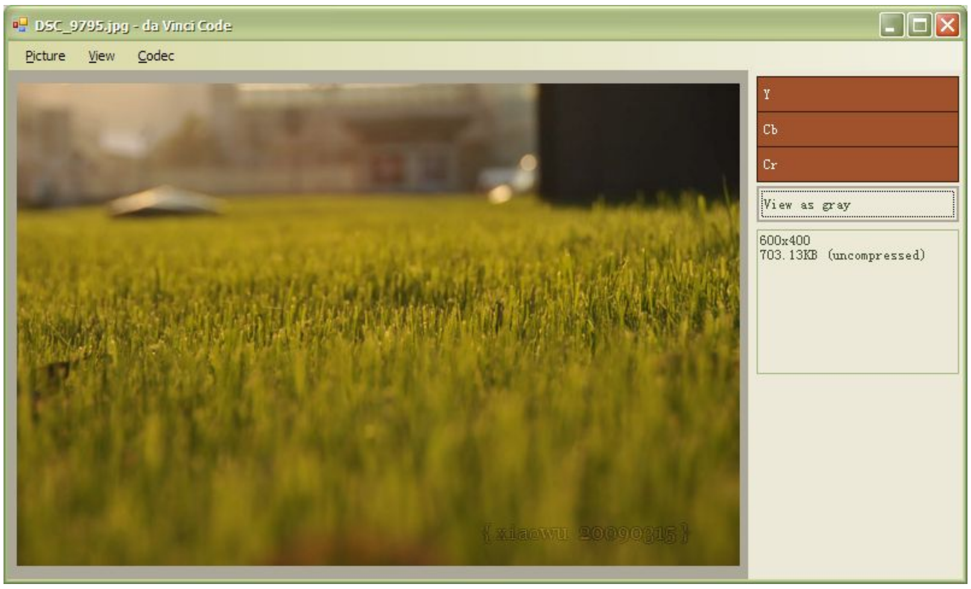


Image is loaded now, and the uncompressed size is displayed.

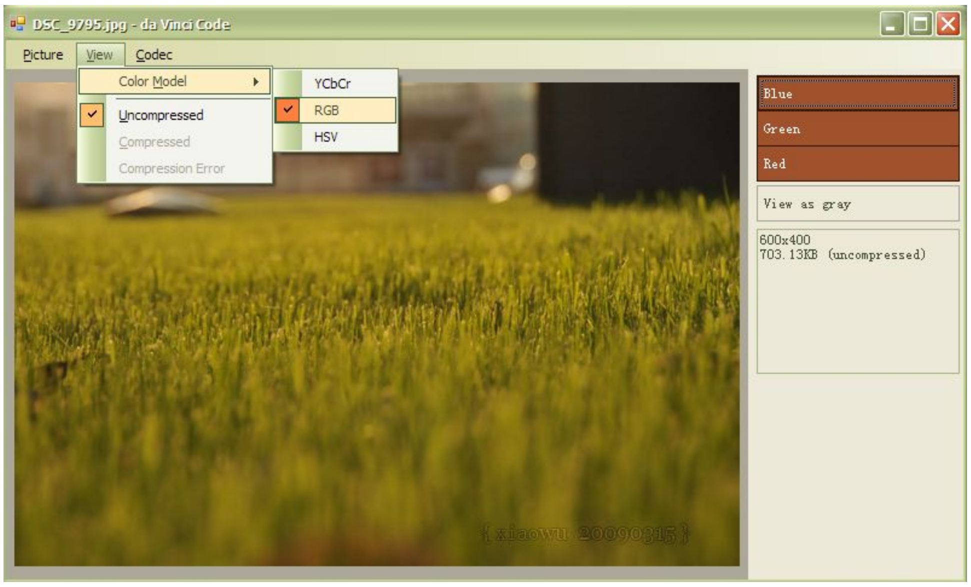


600x400
703.13KB (uncompressed)

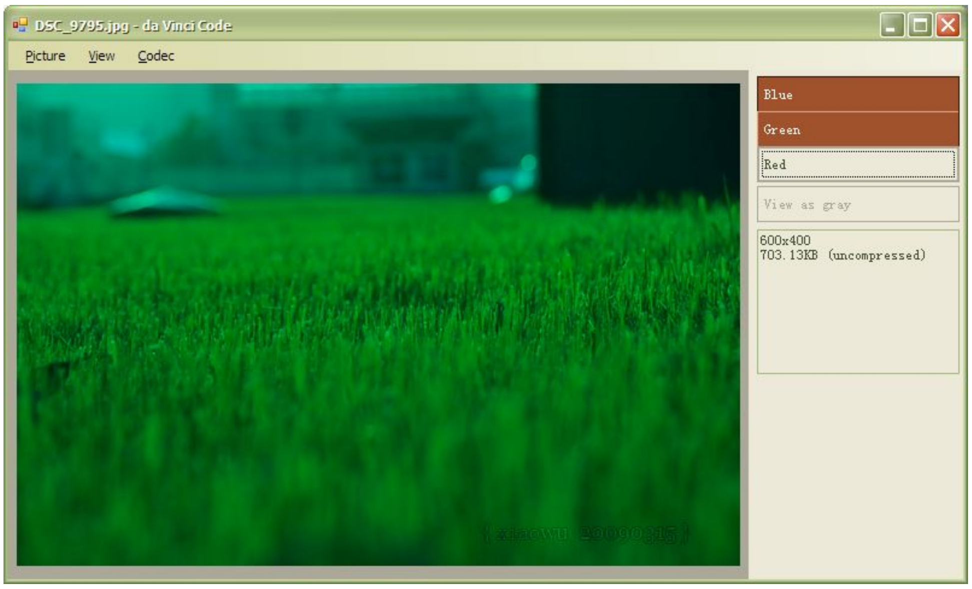
DSC_9795.jpg (600x400, Uncompressed 703.13KB)

2. Change view options.

Change the color model used for rendering via [View->Color Model].

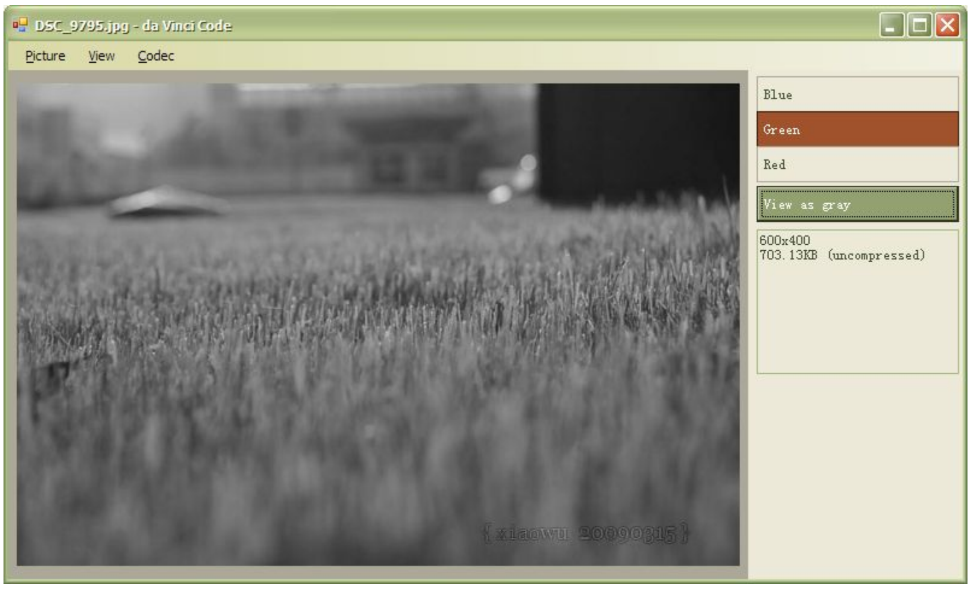


Click a button on the right panel to "turn on/off" a color channel.



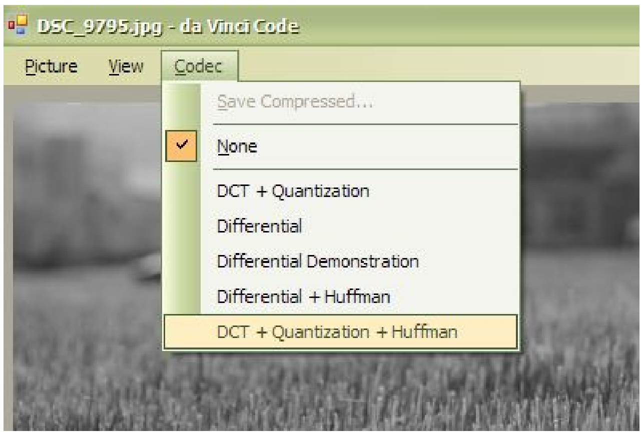
Here, the Red color channel is turned off.

Click “View as gray” to switch between color / monochrome view.



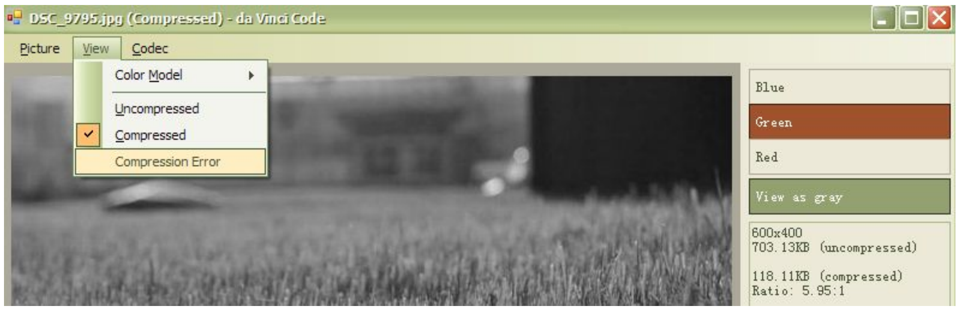
600x400
703.13KB (uncompressed)

Use [Codec] menu to try a codec.



When a codec is activated, three view modes can be chosen:

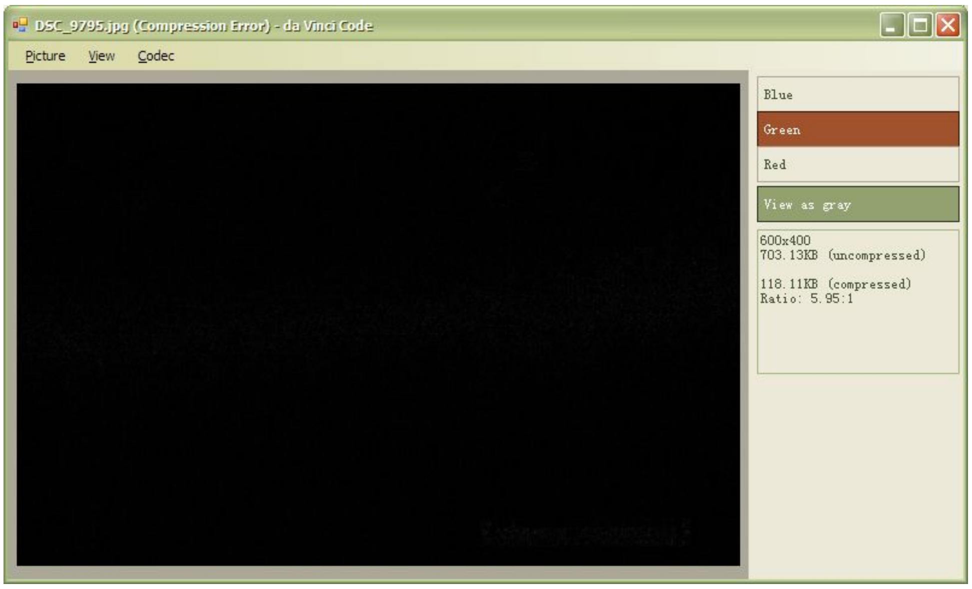
--Uncompressed, Compressed and Compression Error.



600x400
703.13KB (uncompressed)

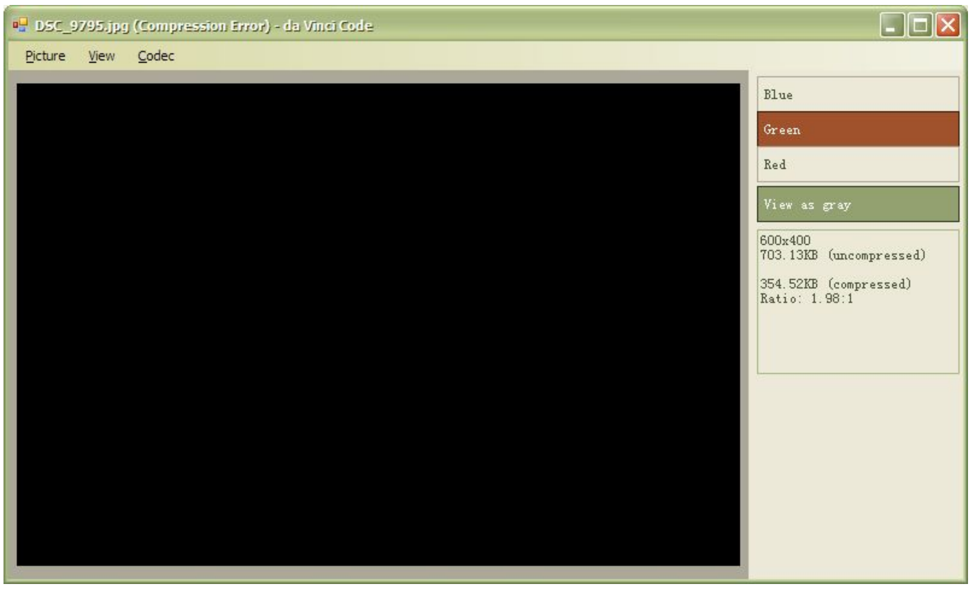
118.11KB (compressed)
Ratio: 5.95:1

Distortion can be observed if a lossy codec is used, though slight.



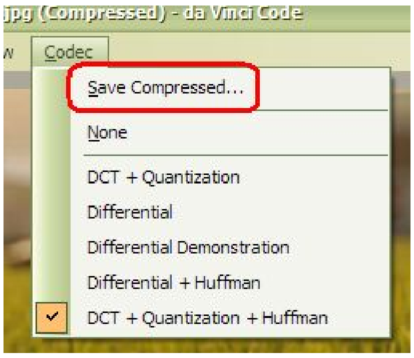
600x400
703.13KB (uncompressed)
118.11KB (compressed)
Ratio: 5.95:1

On the contrary, the Differential-Huffman Codec is lossless.

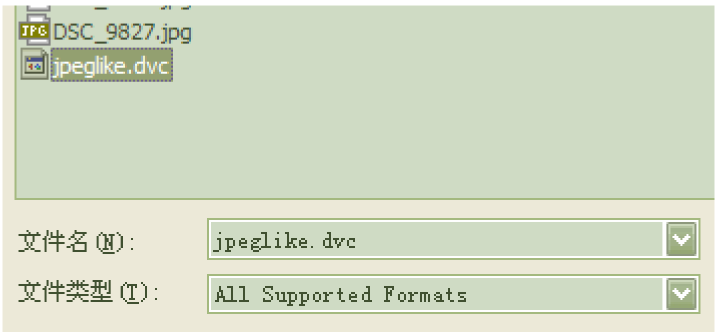


600x400
703.13KB (uncompressed)
354.52KB (compressed)
Ratio: 1.98:1

Try to save a compressed image.


.jpg (Compressed) - da Vinci Code
w
Codec
Save Compressed...
None
DCT + Quantization
Differential
Differential Demonstration
Differential + Huffman
DCT + Quantization + Huffman

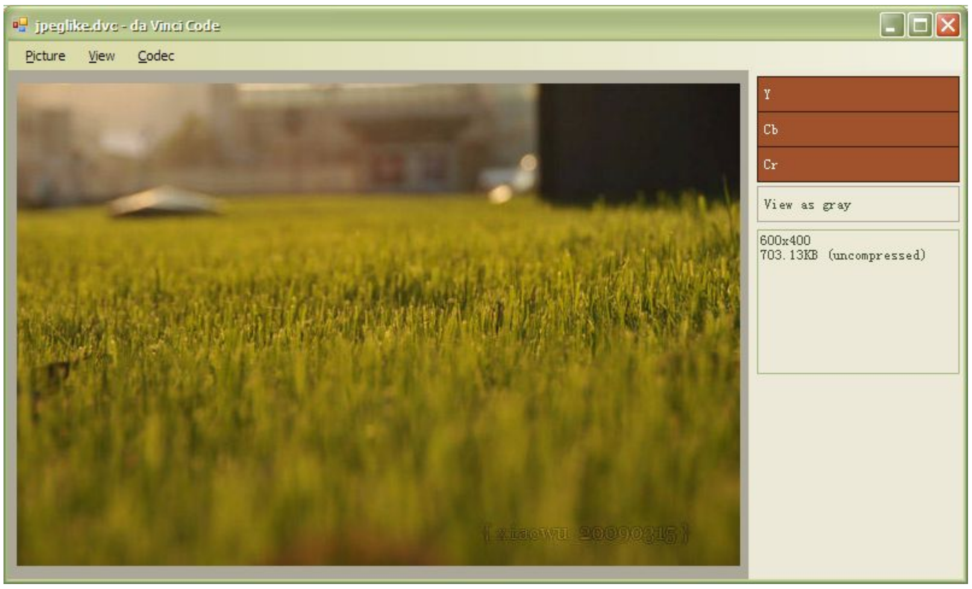
Close the current file and open the compressed one.



文件名(N): jpeglike.dvc

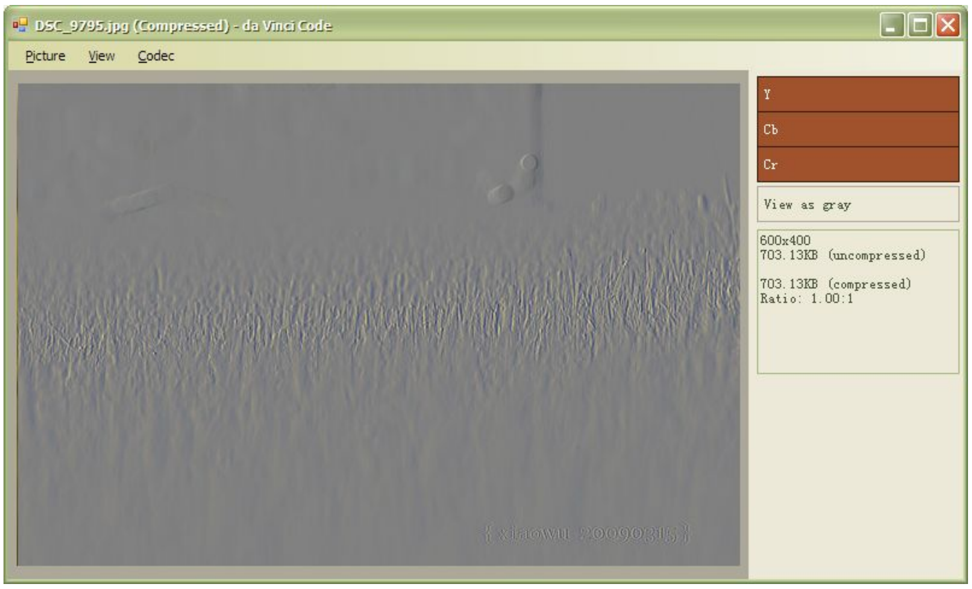
文件类型(T): All Supported Formats

Everything is fine.



600x400
703.13KB (uncompressed)

Try another codec? This illustrates how a differential metod works.

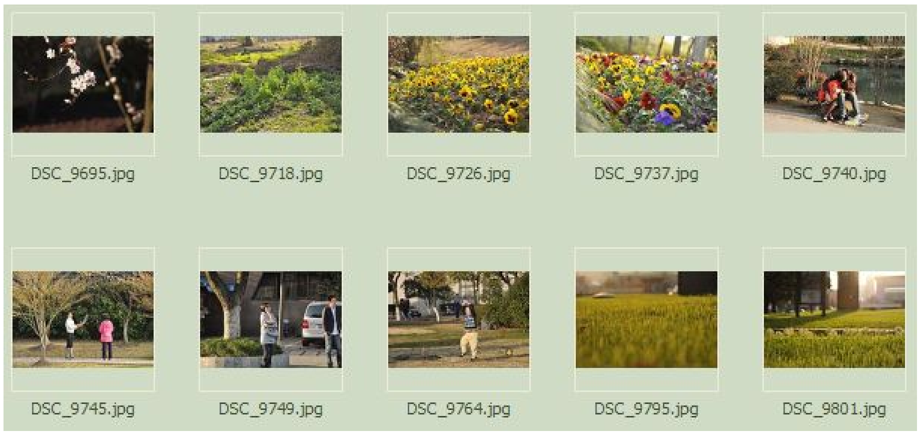


600x400
703.13KB (uncompressed)

703.13KB (compressed)
Ratio: 1.00:1

4.2 Compression Ratios

Let’s try some images.



All the 10 files are 600x400, 24 bpp, JPEG files (Q=75). Uncompressed size of each file is 703.13K.

Now, we are to compare the compression ratio of three different codecs: 1 - JPEG (Q75), 2 - DCT+Quantization+Huffman; 3 - Differential+Huffman.

Source	1		2		3	
DSC_9695	71	9.9:1	111.11	6.3:1	298.03	2.4:1
DSC_9718	180	3.9:1	158.12	4.4:1	555.27	1.3:1
DSC_9726	161	4.4:1	153.14	4.6:1	512.48	1.4:1
DSC_9737	153	4.6:1	148.26	4.7:1	511.14	1.4:1
DSC_9740	167	4.2:1	153.76	4.6:1	575.43	1.2:1
DSC_9745	178	4:1	157.7	4.5:1	572.54	1.2:1
DSC_9749	118	6:1	128.93	5.5:1	474.87	1.5:1
DSC_9764	133	5.3:1	139.28	5:1	498.82	1.4:1
DSC_9765	80	8.8:1	118.11	6:1	354.52	2:1
DSC_9801	106	6.6:1	129.29	5.4:1	427.71	1.6:1
Total	1347	5.2:1	1397.7	5:1	4780.81	1.5:1

Without zigzag scan and RLE on AC coefficients, and differential coding on DC coefficients, a DCT+quantization+Huffman coding scheme can achieve nearly the same compression ratio as JPEG. The lossless differential-Huffman coding, however, only reach a low degree of compression.

[References]

Fundamentals of Multimedia,Ze-Nian Li, Mark S. Drew

Wikipedia：
- DCT,Y’CbCr Color Space,JPEG,HSV HSL Color Model,
- Huffman Coding, Orthogonal matrix
http://en.wikipedia.org/wiki/

JPEG Color Space
http://www.impulseadventure.com/photo/jpeg-color-space.html

MSDN
- MemoryStream (System.IO),BitmapData (System.Drawing.Imaging)
- Marshal members (System.InteropServices)

JPEG
http://en.wikipedia.org/wiki/JPEG

JPEG File Interchange Format Version 1.02