Description of the FFV1 Video Codec

by Michael Niedermayer <michaelni@gmx.at>

19th March 2004

Contents

1 Introduction

The FFV1 video codec is a simple lossless intra only codec. It is still under development and the bitstream might change between versions.

The latest version of this document is available at http://www.mplayerhq.hu/~michael/ffv1.{lyx,txt,html,ps}

This document assumes familiarity with mathematical and coding concepts such as Arithmetic coding and YCbCr colorspaces.

2 Terms and Definitions

CABAC Context Adaptive Binary Arithmetic Coder[H264]

RCT Reversible component transform

3 High-level Description

Each frame is split in 3 planes (Y, Cb, Cr). In the case of the normal YCbCr colorspace the Y plane is coded first followed by the Cb and Cr planes, in the case of the JPEG2000-RCT colorspace the lines are interleaved to reduce cache trashing as most likely the RCT will be immedeatly converted to RGB during decoding, the order of the lines in the interleaving is again Y,Cb,Cr.

Samples within a plane are coded in raster scan order (left->right, top->bottom), each sample is predicted by the median predictor from samples in the same plane and the difference is stored

3.1 Border

Samples above the coded picture are assumed to be 0, right of the coded picture are identical to the closest left sample and left of the coded picture are identical to the top right one

0	0	0	0	0
0	a	b	c	c
a	d		e	e
d	f	g	h	h

Note, this is identical to [JPEGLS]

3.2 Median predictor

median(left, top, left + top - diag)

left, top, diag are the left, top and lefttop samples

Note, this is also used in [JPEGLS, HUFFYUV]

3.3 Context

		T	
	tl	t	tr
L	l	X	

The quantized sample differences L-l, l-tl, tl-t, t-T, t-tr are used as context

$context = Q_0[l - tl] + |Q_0| (Q_1[tl - t] + |Q_1| (Q_2[t - tr] + |Q_2| (Q_3[L - l] + |Q_3| Q_4[T - t])))$

If context < 0 then -context is used and the difference between the sample and its predicted value is encoded with a flipped sign

3.4 Quantization

There are 5 quantization tables for the 5 sample differences, both the number of quantization steps and their distribution are stored in the bitstream. Each quantization table has exactly 256 entries, and the 8 least significant bits of the sample difference are used as index

$Q_i[a - b] = Table_i[(a - b)\&255]$

3.5 Colorspace

3.5.1 JPEG2000-RCT

$Cb = b - g$

$Cr = r - g$

$Y = g + (Cb + Cr) >> 2$

$g = Y - (Cb + Cr) >> 2$

$r = Cr + g$

$b = Cb + g$

[JPEG2000]

3.6 Coding of the sample difference

Instead of coding the 9 (or 10 in the case of RCT) bits of the sample difference with huffman or CABAC coding only the 8 (or 9) least significant bits are used as thats enough the recover the original sample

$coder\_input = \left[\left(sample\_difference + 2^{bits-1}\right) \,\&\, \left(2^{bits} - 1\right)\right] - 2^{bits-1}$

3.6.1 Arithmetic coding mode

Context Adaptive Binary Arithmetic Coding (CABAC) The coder is borrowed from[H264] and the reader is referred to that for further information, it is also noted that the author doesn't know if the coder is patented, so care must be taken if FFV1 is used in countries where software patents are legal. But even if CABAC turns out to be patent free there is still some risk that other parts of FFV1 are covered by patents. By just looking at the very long list of patents which cover other relatively simple standards it is shown that the patent offices seem to pass nearly everything. In many cases the patents cover basic operations like subtraction or taking the median of 3 integers in a specific case like motion vectors.

Non binary values To encode 8 or 9bit numbers as is needed in our case, we could simply encode each bit separately and use the past bits as context, but that would mean 255 contexts per 8bit symbol which is not only a waste of memory but also requires more past data to reach reasonable good estimate of the probabilities. Alternatively simply assuming a laplacian distribution and only dealing with its variance and mean like we do in huffman coding mode would be another possibility, but due to flexibility and simplicity, another method was chosen, which simply uses a single symbol to encode if a number is 0 and if not encodes the number using its exponent,mantisse and sign, the exact contexts which are used can probably better be described by the following code then by some english text

```
put_symbol(CABACContext *c, uint8_t *state, int v, int is_signed, int max_exp){
    int i;
    if(v){
        const int a= ABS(v);
        const int e= av_log2(a);
        put_cabac(c, state+0, 0);
        for(i=0; i<e; i++){
            put_cabac(c, state+1+i, 1);  //1..8
        }
        if(e<max_exp){
            put_cabac(c, state+1+i, 0);      //1..8
```

```
                for(i=e-1; i>=0; i--){
                    put_cabac(c, state+16+e+i, (a>>i)&1); //17..29
                }
                if(is_signed)
                    put_cabac(c, state+9 + e, v < 0); //9..16
            }
        }else{
            put_cabac(c, state+0, 1);
        }
    }
```

max_exp is 7 unless the RCT is used, in which case it is 8 for the samples in a plane and 7 for the elements in the header

is_signed is 1 for encoding of the samples within a plane or line, and 0 for the elements of the header

Initial values for the context model At keyframes all CABAC state variables are set to 0 except number 7 which is set to 124, number 7 is used to select between -127..-64,64..127 and -128 which should explain the strong favoring of the earlier

3.6.2 Huffman coding mode

Uses golomb rice codes. The VLC code is split in 2 parts, the prefix stores the most significant bits, the suffix stores the k least significant bits or stores the whole number in the ESC case

Prefix

bits	value
1	0
01	1
...	...
0000 0000 0001	11
0000 0000 0000	ESC

Suffix

non ESC   the k least significant bits MSB first

ESC   the value - 11, in MSB first order, ESC may only be used if the value cannot be coded as non ESC

Examples

k	bits	value
0	1	0
0	001	2
2	1 00	0
2	1 10	2
2	01 01	5
any	000000000000 10000000	139

run mode run mode is entered when the context is 0, and left as soon as a non 0 difference is found, the level is identical to the predicted one, the run and the first different level is coded

run length coding

level coding is identical to the normal difference coding with the exception that the 0 value is removed as it cant occur

```
if(diff>0) diff--;
encode(diff);
```

Note, this is different from JPEG-LS, which doesn't use prediction in run mode, and uses a different encoding and context model for the last difference, on a small set of test samples the use of prediction improved the compression rate a bit

4 Bitstream

b CABAC 1-bit symbol

v 7bit unsigned symbol coded with the method described in 3.6.1

The same context which is initialized to 0 is used for all fields in the header

4.1 Frame

b keyframe

if(keyframe)

Header

if(JPEG2000_RCT){

for(y=0; y<height; y++){

LumaLine[y]
CbLine[y]
CrLine[y]

}

}else{

LumaPlane

CbPlane

CrPlane

}

4.2 Header

v	Version
v	coder type (0-> golomb rice, 1-> CABAC)
v	Colorspace type (0->YCbCr, 1->JPEG2000_RCT)
b	grayscale
v	log2_h_chroma_subsample ($chroma\_width = 2^{-log2\_h\_chroma\_subsample} luma\_width$)
v	log2_v_chroma_subsample ($chroma\_height = 2^{-log2\_v\_chroma\_subsample} luma\_height$)
b	alpha plane
x	Quantization tables

4.3 Quant Table

The quantization tables are simply stored by storing the number of equal entries -1 of the first half of the table using the method described in 3.6.1, the second half doesn't need to be stored as it is identical to the first with flipped sign

example:
Table: 0 0 1 1 1 1 2 2-2-2-2-1-1-1-1 0
Stored values: 1, 3, 1

5 Changelog

0.01 2003-06-09
initial version by Michael Niedermayer

0.02 2004-02-15
update to match implementation (significant changes)

0.03 2004-02-18
"Boarder"->"Border" by Jim Leonard
huffyuv url by Felix Buenemann
minor fixes by me

0.04 2004-03-19
new huffyuv url by Raphael Clifford as old one is down

6 ToDo

- mean,k estimation for the golomb rice codes
- bitstream end handling
- spelling errors

7 Bibliography

References

[JPEGLS] JPEG-LS FCD 14495 `http://www.jpeg.org/public/fcd14495p.pdf`

[H264] H.264 Draft `http://bs.hhi.de/~wiegand/JVT-G050.pdf`

[HUFFYUV] Huffyuv `http://cultact-server.novi.dk/kpo/huffyuv/huffyuv.html`

[FFMPEG] FFmpeg `http://ffmpeg.org`

[JPEG2000] JPEG2000 url

8 Copyright