DwyaneTalk

Just talk to express yourself

H.264学习笔记5——熵编码之CAVLC

H.264中，4x4的像素块经过变换和量化之后，低频信号集中在左上角，大量高频信号集中在右下角。左边的低频信号相对数值较大，而右下角的大量高频信号都被量化成0、1和-1；变换量化后的残差信息有一定的统计特性和规律。

CAVLC（Context-based Adaptive Variable-Length Code）:基于上下文的可变长度编码，是H.264中进行4x4像素块进行熵编码的方法，基本(baseline)档次中只能使用CAVLC，只有主要档次和扩展档次才能使用CABAC（见笔记：熵编码之CABAC）。

一、 对于经过Zigzag扫描（Z扫描）后的4x4像素残差，**编码过程**包括：

A、非零系数数目(TotalCoeffs)和拖尾系数数目(TrailingOnes)的编码：

拖尾系数：就是指Z扫描后，末尾高频信号中出现连续1或-1的个数（中间可以隔任意多个0），拖尾系数最多有5个。当连续1或-1的个数超过3个，只有最后3个1或-1是拖尾系数，其他的当作普通的非零系数。

TotalCoeffs和TrailingOnes通过查表方式进行编码，H.264针对TotalCoeffs和TrailingOnes，提供了4张变长表和1张定长表（部分见附表9-5）。编码表的选择是由NC确定的，NC的值是由上下文信息确定的。对于色度直流信号，NC=-1；对于其他的NC，根据当前块左边4x4块和上面4x4块的非零系数的个数A和B决定。如表一：其中X表示该块与当前块属于同一slice且可用。根据NC选择编码表的策略如表二：

表一

A（左边块）	B（上面块）	NC
X	X	(A+B)/2
X	—	A
—	X	B
—	—	0

表二

NC	编码表编号
0，1	变长表1
2，3	变长表2
4，5，6，7	变长表3
>=8	定长表
-1	变长表4

通过NC确定所选择的表之后，将编码的二进制输出。

B、每一个拖尾系数的符号正负性编码（按照Z扫面结果的逆序编码）：

按照逆序对每一个拖尾系数的符号进行编码，用0表示1（正）、1表示-1（负），将编码结果输出

C、除拖尾系数外的每一个非零系数幅值(Level，包含正负号信息)编码（按照Z扫描结果的逆序编码）：

拖尾系数幅值的编码包括前缀Level_prefix和后缀Level_suffix。另外编码过程中suffixLength基于上下文信息，根据Level_suffix和Level实时更新。具体过程如下：

1、设置初始SuffixLength：

当TotalCoeffs>10且TrailingOnes<=1时，SuffixLength设为1；否则设为0；

2、将有符号的Level转换成无符号的LevelCode：

若Level > 0：LevelCode = (Level << 1)-2;

若Level < 0：LevelCode = -(Level << 1) - 1;

这样解码时，就可以根据LevelCode的奇偶性判断Level的正负性，从在根据LevelCode解码出有符号的Level。

3、计算Level_prefix 和 Level_suffix：

昵称：DwyaneTalk
园龄：2年5个月
粉丝：10
关注：0
+加关注

< 2015年10月 >

日	一	二	三	四	五	六
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

最新随笔

1. 算法笔记——整数划分3
2. 算法笔记——整数划分2
3. 算法笔记——整数划分1
4. 算法笔记——硬币找零之找钱方案数
5. 算法笔记——硬币找零之最少硬币数
6. VPN介绍及PPTP、L2TP、IPSec等的比较
7. 视频测试序列的下载地址【转】
8. RDO、SAD、SATD、λ相关概念【转】
9. RGB、YUV和YCbCr介绍【转】
10. H.264和HEVC分析软件和工具【转】

随笔分类(28)

C/C++(1)
ubuntu使用(1)
开发管理-Development(7)
视频编码-AVC/HEVC/AVS(10)
数据库(1)
搜索引擎-Search Engine
算法、数据结构(5)
网络系统-Network System(1)
学习笔记-Study Note(2)

随笔档案(28)

2015年7月 (5)
2015年3月 (1)
2014年12月 (3)
2014年11月 (3)
2014年10月 (8)
2014年9月 (6)
2014年3月 (2)

积分与排名

积分 - 4313
排名 - 27625

最新评论

1. Re:H.264和HEVC分析软件和工具【转】

Level_prefix = LevelCode / (1 << SuffixLength);

Level_suffix = LevelCode% (1 << SuffixLength);

4、编码Level_prefix和Level_suffix：

编码Level_prefix是通过查标准表9-6（部分见附表9-6），编码得到的是前缀码，所以解码时可以即使、唯一译码。Level_suffix的编码就是Level_suffix的二进制无符号形式。然后将Level_prefix和Level_suffix的编码结果依次输出，但当SuffixLength=0时，没有Level_suffix，不需要输出。

5、更新SuffixLength的值，回到步骤2继续编码下一个非零系数。更新过程可以用下面代码表示：

```
1 if(SuffixLength == 0)
2     SuffixLength++;
3 else if (abs(Level) > (3 << (SuffixLength -1)) && SuffixLength < 6)
4     SuffixLength++;
```

即：当SuffixLength为0时，SuffixLength加1；当SuffixLength达到6之后不再更新SuffixLength；当SuffixLength在1和6之间，如果当前已编码的非零系数的绝对值(abs(Level))大于给定的阈值S，那么SuffixLength增1，其中阈值S的大小为：S = 3 * 2 ^ (SuffixLength-1) = 3<<(SuffixLength-1)。

D、最后一个非零系数前0的数目(TotalZeros)编码：查找标准表9-7（部分见附表9-7）~9-9

E、每一个非零系数前连续0的数目（RunBefore）编码（按照Z扫面结果的逆序编码）：查找标准表9-10（部分见附表9-10）

编码过程中，ZerosLeft表示当前编码非零系数左边所有0的个数，对于最后一个（逆序的最后一个）非零系数前0的个数不需要编码。

二、例如：对于4x4的残差块，如下图：

0	3	-1	0
0	-1	1	0
1	0	0	0
0	0	0	0

经过Z扫描得到序列：0，3，0，1，-1，-1，0，1，0，0，0，0，0，0，0，0。

对该序列进行**编码**如下：

1、初始值设定：

TotalCoeffs = 5；TrailingOnes = 3；TotalZeros = 3；假设 NC = 1；SuffixLength = 0；最终编码输出码流为out。

2、编码TotalCoeffs和TrailingOnes：

查附表9-5得，TotalCoeffs = 5、TrailingOnes = 3和0 <= NC <2时；编码结果为：0000 100。

此时out = 0000 100。

3、编码拖尾系数符号：

拖尾系数为1,-1,-1（扫描逆序），对应的符号编码为0,1,1。所以此时out = 0000 1000 11。

4、编码每个非拖尾非零系数的幅值Level：

需要编码的非零系数有1，3（Z扫描逆序），初始i=2，过程如下：

Level[i--] = 1; 得到LevelCode = 0，Level_prefix = 0，没有Level_suffix(因为SuffixLength=0)，查表9-6得编码结果为1。此时out = 0000 1000 111。

然后更新SuffixLength，SuffixLength++得SuffixLength=1；

Level[i--] = 3;得到LevelCode=4,Level_prefix = 2,Level_suffix = 0,查表9-6得Level_prefix编码结果为001，然后Level_suffix的二进制表示为0。

out = 0000 1000 1110 010。此时i=0，此步骤编码结束。

5、编码TotalZeros：查表9-7得编码结果为编码结果为111，此时out = 0000 1000 1110 0101 11。

6、编码每一个非零系数前的连续o的数目：有四个非零系数前连续0的个数要编码，初始i=5。查表9-10的编码过程如下：

ZerosLeft = 3， RunBefore = 1，Level[i--]=1，编码结果为10；

ZerosLeft = 2， RunBefore = 0，Level[i--]=-1,编码结果为1；

ZerosLeft = 2， RunBefore = 0，Level[i--]=-1,编码结果为1；

ZerosLeft = 2， RunBefore = 1，Level[i--]=1,编码结果为01；

ZerosLeft = 1， RunBefore = 1，Level[i--]=3,此时对应第一个非零系数，不需要编码；

@Dennis Gao谢提醒，因为是从360doc那边转帖，所以图片被360doc给屏了，现已修复！...

--DwyaneTalk

2. Re:H.264和HEVC分析软件和工具【转】
楼主，看不到图

--Dennis Gao

3. Re:C/C++语言学习——内存分配管理
作者似乎已经说的很清楚了。

--liuwenstudio

4. Re:C/C++语言学习——内存分配管理
脱离具体环境谈内存管理毫无意义
因为C语言根本就对这些方面做过任何规定

--garbageMan

5. Re:C/C++语言学习——内存分配管理
mark

--红涛

阅读排行榜

1. C/C++语言学习——内存分配管理(812)
2. Mysql——Innodb和Myisam概念与数据恢复(586)
3. H.264和HEVC分析软件和工具【转】(448)
4. H.264学习笔记5——熵编码之CAVLC(371)
5. H.264学习笔记4——变换量化(357)

最后输出结果为 out = 0000 1000 1110 0101 1110 1101。

相应**解码**如下：读入out = 0000 1000 1110 0101 1110 1101

1、解码TotalCoeffs和TrailingOnes：

初始NC=1，根据表9-5，可以从0000100中解码得到TotalCoeffs=5；TrailingOnes=3。这里因为表9-5的编码可以及时、唯一译码，所以遇到的第一个合法的01串就是TotalCoeffs和TrailingOnes编码的结果。

2、解码拖尾系数：此时out = 0 1110 0101 1110 1101

由TrailingOnes=3知，有3个拖尾系数，所以对应的正负号编码为011，所以3个拖尾系数是1，-1，-1。所以解码输出 in是 -1,-1,1（因为编码是逆序的）。

3、解码除拖尾系数外的非零系数：此时out = 10 0101 1110 1101

由TotalCoeffs=5；TrailingOnes=3知除拖尾系数外，还有两个非零系数。初始SuffixLength=0，所以根据表9-6（及时、唯一译码）解码如下：

SuffixLength=0 查表得比特串为1，level_prefix=0，LevelCode=0（偶数），Level=1，没有Level_suffix；(消耗码流0)

SuffixLength=1 查表得比特串为001，level_prefix=2，LevelCode=4（偶数），Level=3，Level_suffix=0；（消耗码流0010）

所以输出 in是 3，1，-1，-1，1

4、解码每个非零系数前0的个数：此时out = 1 1110 1101

TotalCoeffs = 5；根据表9-7解码得 TotalZeros=3，对应码流：111。

然后查表9-10，得到每一个非零系数前连续0的个数，过程如下：此时out = 10 1101

TotalZeros=3，根据码流查表得10对应 RunBefore=1，in是 3，1，-1，-1，0，1

TotalZeros=3-1=2，根据码流查表得1对应 RunBefore=0，in是 3，1，-1，-1，0，1

TotalZeros=2-0=2，根据码流查表得1对应 RunBefore=0，in是 3，1，-1，-1，0，1

TotalZeros=2-0=2，根据码流查表得01对应 RunBefore=1，in是 3，0，1，-1，-1，0，1

TotalZeros=2-1=1，out码流解码完，所以TotalZeros=1表示3之前的0数目，in是 0，3，1，-1，-1，0，1

然后在结尾补0组成16个残差系数，得解码结果0，3，1，-1，-1，0，1，0，0，0，0，0，0，0，0，0

三、附表：

表 9-5— 映射到TotalCoeff(coeff_token) 和TrailingOnes(coeff_token) 的coeff_token

TrailingOnes (coeff_token)	TotalCoeff (coeff_token)	0 <= nC < 2	2 <= nC < 4	4 <= nC < 8	8 <= nC	nC = = −1	nC = = −2
0	0	1	11	1111	0000 11	01	1
0	1	0001 01	0010 11	0011 11	0000 00	0001 11	0001 111
1	1	01	10	1110	0000 01	1	01
0	2	0000 0111	0001 11	0010 11	0001 00	0001 00	0001 110
1	2	0001 00	0011 1	0111 1	0001 01	0001 10	0001 101
2	2	001	011	1101	0001 10	001	001
0	3	0000 0011 1	0000 111	0010 00	0010 00	0000 11	0000 0011 1
1	3	0000 0110	0010 10	0110 0	0010 01	0000 011	0001 100
2	3	0000 101	0010 01	0111 0	0010 10	0000 010	0001 011
3	3	0001 1	0101	1100	0010 11	0001 01	0000 1
0	4	0000 0001 11	0000 0111	0001 111	0011 00	0000 10	0000 0011 0
1	4	0000 0011 0	0001 10	0101 0	0011 01	0000 0011	0000 0010 1
2	4	0000 0101	0001 01	0101 1	0011 10	0000 0010	0001 010
3	4	0000 11	0100	1011	0011 11	0000 000	0000 01

表 9-6—level_prefix的码字表格 (资料性)

level_prefix	比特串
0	1
1	01
2	001
3	0001
4	0000 1
5	0000 01
6	0000 001
7	0000 0001
8	0000 0000 1
9	0000 0000 01
10	0000 0000 001

表 9-7—TotalCoeff(coeff_token) 1到 7的4x4块total_zeros表格

total_zeros	TotalCoeff(coeff_token)						
	1	2	3	4	5	6	7
0	1	111	0101	0001 1	0101	0000 01	0000 01
1	011	110	111	111	0100	0000 1	0000 1
2	010	101	110	0101	0011	111	101
3	0011	100	101	0100	111	110	100
4	0010	011	0100	110	110	101	011
5	0001 1	0101	0011	101	101	100	11
6	0001 0	0100	100	100	100	011	010
7	0000 11	0011	011	0011	011	010	0001
8	0000 10	0010	0010	011	0010	0001	001
9	0000 011	0001 1	0001 1	0010	0000 1	001	0000 00
10	0000 010	0001 0	0001 0	0001 0	0001	0000 00	

表 9-10—run_before表格

run_before	zerosLeft						
	1	2	3	4	5	6	>6
0	1	1	11	11	11	11	111
1	0	01	10	10	10	000	110
2	-	00	01	01	011	001	101
3	-	-	00	001	010	011	100
4	-	-	-	000	001	010	011
5	-	-	-	-	000	101	010
6	-	-	-	-	-	100	001
7	-	-	-	-	-	-	0001
8		-	-	-	-	-	00001
9	-	-	-	-	-	-	000001
10	-	-	-	-	-	-	0000001

分类: 视频编码-AVC/HEVC/AVS

标签: H.264, 熵编码, 变长编码, 视频编码