

怎么解析HEVC/H265解码参数 HEVCDecoderConfigurationRecord?

如何从HEVC的VPS,SPS,PPS中获取HEVCDecoderConfigurationRecord (ISO-IEC-14496-15) 参数集? ...显示全部

关注问题 写回答 邀请回答 好问题 添加评论 分享

2 个回答

默认排序



天涯来客

+ 关注

HEVCDecoderConfigurationRecord的定义参考:

ffmpeg.org/DOXYGEN/trun...

具体解析部分参考: ffmpeg.org/DOXYGEN/trun...

发布于 2018-03-04 19:15

赞同 添加评论 分享 收藏 喜欢



后端开发老鸽

+ 关注

H265出现的原因:

我们视频的分辨率 从 720p到1080p,而且电视的屏幕也越来越大

视频帧率从30帧 到60帧,再到120帧

这就会导致我们cpu在编解码的时候,会出现宏块个数爆发式增长,运动矢量复杂度增加的后果,直接导致我们编码后的视频文件依旧很大,所以H264编码方式已经不能满足现在的需求了,H265也就应运而生。

下面是通过Elecard HEVCAnalyzer 软件分析Q宏块的效果,可以看到我们的宏块大小在颜色差异大的地方越小,这样我们的画面在界面明显差异的地方就会越清晰。

- H265和H264编码数据比较:
 - 编码后H265 I帧数据大小 比H264 I帧数据大
 - 编码H265 P帧、B帧数据大小 比 H264P帧、B帧数据小
- 所以H265的编码数据小的主要原因在于 P帧、B帧数据很小。

下图是典型的占用字节数Q。(前面依旧是四个字节的分隔符)

7.3.1.2 NAL unit header syntax

nal_unit_header()	Descriptor
forbidden_zero_bit	fl(1)
nal_unit_type	u(6)
nuh_layer_id	u(6)
nuh_temporal_id_plus1	u(3)

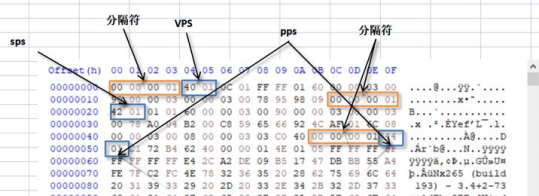
H265-NALU-Type

接下来看一下H265的NALU类型的种类。如下图。

Table 7-1 – NAL unit type codes and NAL unit type classes			
nal_unit_type	Name of nal_unit_type	Content of NAL unit and RBSP syntax structure	NAL unit type class
0 1	TRAIL_N TRAIL_R	Coded slice segment of a non-TSA, non-STSA trailing picture slice_segment_layer_rbsp()	VCL
2 3	TSA_N TSA_R	Coded slice segment of a TSA picture slice_segment_layer_rbsp()	VCL
4 5	STSA_N STSA_R	Coded slice segment of an STSA picture slice_segment_layer_rbsp()	VCL
6 7	RADL_N RADL_R	Coded slice segment of a RADL picture slice_segment_layer_rbsp()	VCL
8 9	RASL_N RASL_R	Coded slice segment of a RASL picture slice_segment_layer_rbsp()	VCL
10 12 14	RSV_VCL_N10 RSV_VCL_N12 RSV_VCL_N14	Reserved non-IRAP SLNR VCL NAL unit types	VCL
11 13 15	RSV_VCL_R11 RSV_VCL_R13 RSV_VCL_R15	Reserved non-IRAP sub-layer reference VCL NAL unit types	VCL
16 17 18	BLA_W_LP BLA_W_RADL BLA_N_LP	Coded slice segment of a BLA picture slice_segment_layer_rbsp()	VCL
19 20	IDR_W_RADL IDR_N_LP	Coded slice segment of an IDR picture slice_segment_layer_rbsp()	VCL
21	CRA_NUT	Coded slice segment of a CRA picture slice_segment_layer_rbsp()	VCL
22 23	RSV_IRAP_VCL22 RSV_IRAP_VCL23	Reserved IRAP VCL NAL unit types	VCL

24..31	RSV_VCL24.. RSV_VCL31	Reserved non-IRAP VCL NAL unit types	VCL
32	VPS_NUT	Video parameter set video_parameter_set_rbsp()	non-VCL
33	SPS_NUT	Sequence parameter set seq_parameter_set_rbsp()	non-VCL
34	PPS_NUT	Picture parameter set pic_parameter_set_rbsp()	non-VCL
35	AUD_NUT	Access unit delimiter access_unit_delimiter_rbsp()	non-VCL
36	EOS_NUT	End of sequence end_of_seq_rbsp()	non-VCL
37	EOB_NUT	End of bitstream end_of_bitstream_rbsp()	non-VCL
38	FD_NUT	Filler data filler_data_rbsp()	non-VCL
39 40	PREFIX_SEI_NUT SUFFIX_SEI_NUT	Supplemental enhancement information sei_rbsp()	non-VCL
41..47	RSV_NVCL41.. RSV_NVCL47	Reserved	non-VCL
48..63	UNSPEC48.. UNSPEC63	Unspecified	non-VCL

接下来直接来分析H265编码数据中的 NALU_TYPE;



下载知乎客户端
与世界分享知识、经验和见解



相关问题

有大佬知道4090在pr里面使用hevc格式导出视频只使用一个编码器是什么原因吗? 0 个回答

把HEVC视频导入pr,播放卡顿,曾经删掉过编码器,后来又回来了,怎么办? 0 个回答

hevc编码器中如何设定mv的值呢,例如将每个p帧mv设为 (1, 1)? 0 个回答

devc++运行时打开stdio.h文件,而且原文件没有运行,怎么办恢复? 0 个回答

PR,导出视频失败,显示必须安装hevc编码器才能使用此功能,但是也安装失败,如何安装? 3 个回答



帮助中心

知乎隐私保护指引申请开通机构号联系我们



举报中心

涉未成年举报网络谣言举报涉企虚假举报更多



关于知乎

下载知乎知乎招聘知乎指南知乎协议更多

京 ICP 证 110745 号 · 京 ICP 备 13052560 号 - 1 · 京公网安备 11010802020088 号 · 京网文[2022]2674-081 号 · 药品医疗器械网络信息服务备案 (京) 网药械信息备字 (2022) 第00334号 · 广播电视节目制作经营许可证: (京) 字第06591号 · 服务热线: 400-919-0001 · Investor Relations · © 2023 知乎 北京智者天下科技有限公司版权所有 · 违法和不良信息举报: 010-82716601 · 举报邮箱: jubao@zhihu.com



专业化 无边界服务

00000000 63 61 31 59 62 65 33 37 3A 5B 5B 69 6E 64 6A10 00000000 6F 77 73 5D 58 47 43 43 20 31 30 2E 32 2E 31 5D 00000000 5B 36 34 20 62 69 74 5D 20 38 62 69 74 2B 31 30 000000C0 62 69 74 20 2D 20 48 2E 32 36 35 2F 48 45 56 43 000000D0 20 63 6F 64 65 63 20 2D 20 43 6F 70 79 72 69 67 000000E0 68 74 20 32 30 31 33 2D 32 30 31 38 20 28 63 29 000000F0 20 4D 75 6C 74 69 63 6F 72 65 77 61 72 65 2C 20 00000100 49 6E 63 20 2D 20 68 74 74 70 3A 2F 2F 78 32 36 00000110 35 2E 6F 72 67 20 2D 20 6F 70 74 69 6E 6E 73 3A 00000120 20 63 70 75 69 64 3D 31 31 31 31 30 33 39 20 66 00000130 72 61 6D 65 2D 74 68 72 65 61 64 73 3D 33 20 6E 00000140 75 6D 61 2D 70 6F 6C 73 3D 38 20 77 70 70 20

来举个例子，解释怎么计算的吧。

```
40 01
//根据上面的NAL_unit_header_type
16进制:01000000 000 00001
//第二位包括第二位往后6位都是nal_Type,所以计算40 里的 6位 就行了
01000000 & 0x7E << 1
计算结果:32
//查表得知
这个帧类型是 vps
```

上面是热身，接下来就进入实战，我们来解析sps里的高度和宽度。

首先我们必须知道那两个字段是高度和宽度，这就要找官方文档了。

首先我们需要看几个字段的含义，不然看文档的时候会一脸懵

H264编码原理，H264源码解析_哔哩哔哩_bilibili

www.bilibili.com/video/BV163411J7Fe/

H264编码原理
你学，我还敢讲7

名称	含义
ae (v)	基于上下文自适应的二元算术编码
b (8)	读进连续的 8 bit
f (n)	读进连续的nbit
se (v)	符号号指数 Golomb 熵编码
u (n)	读进连续的 n bit，且它们解码后的值为无符号整数
e (n)	无符号指数 Golomb熵编码

好的我们了解了这几个字符之后，就可以看懂点文档了，贴sps编码Q格式

seq_parameter_set_rbsp() {

sps_video_parameter_set_id

sps_max_sub_layers_minus1

sps_temporal_id_nesting_flag

profile_tier_level(sps_max_sub_layers_minus1)

Descriptor

u(4)

u(3)

u(1)

• 56 •

所占字节数

第 3 章 编码结构

续表

sps_seq_parameter_set_id

chroma_format_idc

if(chroma_format_idc == 3)

separate_colour_plane_flag

pic_width_in_luma_samples

pic_height_in_luma_samples

ue(v)

ue(v)

u(1)

ue(v)

ue(v)

从图中我们可以很容易的发现，宽度和高度在sps编码格式的所处位置和占用字节数，我们只需要按照这张表格的规范，逐个解析即可，但是，我们需要注意，当我们解析的时候遇到 00 00 03的数值时，我们需要跳过03 ,接下来，看一个表格(sps模拟数据读取宽高表格分解图Q)

1	2	3	4	5	6	7	8	9
42	01	01	01	60	00	00	03	00
0100 0010	0000 00001	0000 00001	0000 0001	0110 0000	0000 0000	0000 0000	0000 0011	0000 0000
10	11	12	13	14	15	16	17	18
90	00	00	03	00	00	03	00	78
1001 0000	0000 0000	0000 0000	0000 0011	0000 0000	0000 0000	0000 0011	00 0000	0111 1000
19	20	21	22	23	24	25	26	27
A0	04	B2	00	C8	59	65	66	92
010 0000	0000 0100	1011 0011	0000 0000	1100 0000	0 01 1001	0110 0101	0110 0110	1001 0010
4C								
0100 1100								

上图就是我截取的数据(一段真实的视频编码Q)，图中灰色的区域，我们解析的时候需要跳过这个字段，不去解析，这个是转义符Q的意思。红色字体就是我们的宽度(pic_width_in_luma_samples)，而橘黄色字体就是我们的高度(pic_height_in_luma_samples),图中其他不同颜色的是解析的时候比较混淆的地方，我标出来，可以更好的区分。

LinuxC++音视频开发视频：免费】Ffmpeg/WebRTC/RTMP/NDK/Android音视频流媒体高级开发

【文章福利】：音视频学习资料、视频和学习路线图资料、以及面试题（资料包括C/C++，Linux，Ffmpeg webRTC rtmp hls rtsp ffmpeg srs等），免费分享，有需要的可以加君羊领取哦！~学习交流裙994289133加入领取资料

- rtmp_test
- 03-avpacket-avframe-api-test.rar
- 04-clock-test.rar
- ffmpeg-pro.rar
- ffplay-pro.rar
- rtmp_test.rar
- 01-课程介绍和基础知识讲解-20180918.pdf
- 02-FFMPEG入门.pdf
- 03-FFMPEG内存模型和播放引擎架构讲解-20180926.pdf
- 04-音视频同步分析-基于ffplay-20180928.pdf
- 07-播放器开发系列01.pdf
- 08-播放器开发系列(2)-播放音频.pdf
- 09-播放器开发系列(3)-音视频同步(上).pdf
- 10-播放器开发系列(4)-播放控制-20181019.pdf
- 11-播放器开发系列(5)-音视频同步(下).pdf
- 12-Advanced Audio Coding (AAC).pdf
- 13-新一代视频压缩标准-H.264_AVC(第二版).毕厚杰，王健编著.扫描版.pdf
- 13-新一代高效视频编码H.265/HEVC原理、标准与实现 [万帅，杨付正 编著] 2014年版.pdf
- 16-FLV标准文档.pdf
- 16-rtmp_specification_1.0.pdf
- 16-rtmp_specification_1.0_cn.pdf
- 19-RTSP协议详解.pdf

- 20-RTCP3550(英文)-RTP A Transport Protocol for Real-time Applications.pdf
- 20-RFC3640(英文)-RTP Payload Format for Transport of MPEG-4 Elementary Streams.pdf
- 20-RFC3984(英文)-RTP Payload Format for H.264 Video.pdf
- 21-RFC4566-SDP Session Description Protocol.pdf
- Ffmpeg命令.pdf
- FLV解复用实战-FlvParser源码阅读.pdf
- H.264协议英文版.pdf
- QT+FFmpeg4.x Windows开发环境搭建.docx
- WebRTC开源项目-手把手教你搭建AppRTC-20200423.pdf
- WebRTC音视频通话必备手册-20200423.pdf
- Windows Ffmpeg命令行环境搭建.docx

企鵝君羊994289133领取资料

- 13-新一代高效视频编码H.265/HEVC原理、标准与实现 [万帅, 杨付正 编辑] 2014年版.pdf
- h265.mp4

解析代码如下:

```
#include <iostream>
#include <memory.h>
#include <bh.h>
typedef signed char int8;

typedef unsigned char uint8Q;

typedef unsigned short uint16Q;

typedef unsigned long uint32Q;
typedef signed char int8;
typedef signed short int16;
typedef signed long int32;
struct vc_params_t
{
    long width,height;
    int8 profile, level;
    int8 nal_length_size;
    void clear()
    {
        memset(this, 0, sizeof(*this));
    }
};

class NALBitstream
{
public:
    int m_idxQ;
    NALBitstream() : m_data(NULL), m_len(0), m_idx(0), m_bits(0), m_byte(0), m_z
    {

    };
    NALBitstream(void * data, int lenQ)
    {
        Init(data, len);
    };
    void Init(void * data, int len)
    {
        m_data = (LPBYTE)data;
        m_len = len;
        m_idx = 0;
        m_bits = 0;
        m_byte = 0;
        m_zeros = 0;
    };
    BYTE GetBYTE()
    {
        if ( m_idx >= m_len )
            return 0;
        BYTE b = m_data[m_idx++];
        if ( b == 0 )
        {
            m_zeros++;
            if ( (m_idx < m_len) && (m_zeros == 2) && (m_data[m_idx] == 0x03) )
            {
                m_idx++;
                m_zeros=0Q;
            }
        }
        else m_zeros = 0;

        return b;
    };

    UINT32 GetBit()
    {
        if (m_bits == 0)
        {
            m_byte = GetBYTE();
            m_bits = 8;
        }
        m_bits--;
        return (m_byte >> m_bits) & 0x1;
    };

    UINT32 GetWord(int bitsQ)
    {
        UINT32 u = 0;
        while ( bits > 0 )
        {
            u <<= 1;
            u |= GetBit();
            bits--;
        }
        printfQ("end index is :%d\n",m_idx);
        return u;
    };

    //返回哥伦布编码Q对应的实际值
    UINT32 GetUE()
    {
        int zeros = 0;
        while (m_idx < m_len && GetBit() == 0 ) zeros++;
        UINT32 value=GetWord(zeros) + ((1 << zeros) - 1);
        return value;
    };

    INT32 GetSE()
    {
        UINT32 UE = GetUE();
        bool positive = UE & 1;
        INT32 SE = (UE + 1) >> 1;
        if ( lpositive )
```

```

    {
        SE = -SE;
    }
    return SE;
};
private:
    LPBYTE m_data;
    int m_len;
    int m_bits;
    BYTE m_byte;
    int m_zeros;
};

bool ParseSequenceParameterSet(BYTE* data,int size, vc_params_t& params)
{
    if (size < 20)
    {
        return false;
    }
    NALBitstream bs(data, size);
    // seq_parameter_set_rbsp()
    int test =bs.GetWord(4);// sps_video_parameter_set_id
    int sps_max_sub_layers_minus1 = bs.GetWord(3);
    if (sps_max_sub_layers_minus1Q > 6)
    {
        return false;
    }
    test =bs.GetWord(1);
    {
        test =bs.GetWord(2);
        test =bs.GetWord(1);
        test = params.profile = bs.GetWord(5);
        test = bs.GetWord(32);//6
        test =bs.GetWord(1);//
        test =bs.GetWord(1);//
        test = bs.GetWord(1);//
        test =bs.GetWord(1);//
        test = bs.GetWord(44);//
        test= params.level = bs.GetWord(8);// general_level_idc
        uint8 sub_layer_profile_present_flag[6] = {0};
        uint8 sub_layer_level_present_flag[6] = {0};
        for (int i = 0; i < sps_max_sub_layers_minus1; i++) {
            sub_layer_profile_present_flag[i]= bs.GetWord(1);
            sub_layer_level_present_flag[i]= bs.GetWord(1);
        }
        if (sps_max_sub_layers_minus1 > 0)
        {
            for (int i = sps_max_sub_layers_minus1; i < 8; i++) {
                uint8 reserved_zero_2bits = bs.GetWord(2);
                printf("-----");
            }
        }
        for (int i = 0; i < sps_max_sub_layers_minus1; i++)
        {
            if (sub_layer_profile_present_flag[i]) {
                test =bs.GetWord(2);
                test = bs.GetWord(1);
                test = bs.GetWord(5);
                test =bs.GetWord(32);
                test = bs.GetWord(1);
                test = bs.GetWord(1);
                test = bs.GetWord(1);
                test = bs.GetWord(1);
                test = bs.GetWordQ(1);
                test = bs.GetWord(44);
            }
            if (sub_layer_level_present_flag[i]) {
                test = bs.GetWord(8);// sub_layer_level_idc[i]
            }
        }
    }
    uint32 sps_seq_parameter_set_id= bs.GetUE();
    if (sps_seq_parameter_set_id > 15) {
        return false;
    }
    uint32 chroma_format_idc = bs.GetUE();
    if (sps_seq_parameter_set_id > 3) {
        return false;
    }
    if (chroma_format_idc == 3) {
        test = bs.GetWord(1);//
    }

    params.width = bs.GetUE(); // pic_width_in_luma_samples
    params.height = bs.GetUE(); // pic_height_in_luma_samples
    if (bs.GetWord(1)) {
        bs.GetUE();
        bs.GetUE();
        bs.GetUE();
        bs.GetUE();
    }
    uint32 bit_depth_luma_minus8Q= bs.GetUE();
    uint32 bit_depth_chroma_minus8= bs.GetUE();
    if (bit_depth_luma_minus8 != bit_depth_chroma_minus8) {
        return false;
    }
    return true;
}

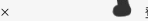
int main() {
    vc_params_t params = {0};
    BYTE Sps[41] = {0x42,0x01,0x01,0x01,0x60,0x00,0x00,0x03,0x00,0x90,0x00,0x03,0x00,0x00,0x00,0x03,0x00,0x78,0xA0,0x04,0xB2,0x00,0xC8,0x59,0x65,0x66,0x92,0x4C,0xAF,0x01,0x6C,0x00,0x00,0x03,0x00,0x00,0x00};
    ParseSequenceParameterSet(Sps,41,params);
    printf("%d-%d-%d\n",params.width,params.height,params.level);
    system("pause");
    return 0Q;
}

```

发布于 2022-05-23 20:23

写回答





登录即可查看 超5亿 专业优质内容
超 5 千万创作者的优质提问、专业回答、深度文章和精彩视频尽在知乎

▲ 赞同

● 添加评论

分享

★ 收藏

♥ 喜欢

收起 ^

