【H.264/AVC视频编解码技术详解】十一、H.264的Slice Header解析

《H.264/AVC视频编解码技术详解》视频教程已经在“CSDN学院”上线，视频中详述了H.264的背景、标准协议和实现，并通过一个实战工程的形式对H.264的标准进行解析和实现，欢迎观看！

“纸上得来终觉浅，绝知此事要躬行”，只有自己按照标准文档以代码的形式操作一遍，才能对视频压缩编码标准的思想和方法有足够深刻的理解和体会！

链接地址：H.264/AVC视频编解码技术详解

GitHub代码地址：点击这里

H.264中的条带(Slice)

1. Slice的概念

我们已经知道，整个H.264的码流结构可以分为两层：网络抽象层NAL和视频编码层VCL。在NAL层，H.264的码流表示为一系列的NAL Units，不同的NAL Units中包含不同类型的语法元素。前面两篇中所解析的序列参数集SPS和图像参数集PPS就是其中重要的两个部分，其中包含了控制解码过程的一些通用的参数。

实际保存原始视频的图像数据的部分保存在其他的VCL层的NAL Units中。这部分数据在码流中称作是条带(Slice)。一个Slice包含一帧图像的部分或全部数据，换言之，一帧视频图像可以编码为一个或若干个Slice。一个Slice最少包含一个宏块，最多包含整帧图像的数据。在不同的编码实现中，同一帧图像中所构成的Slice数目不一定相同。

在H.264中设计Slice的目的主要在于防止误码的扩散。因为不同的slice之间，其解码操作是独立的。某一个slice的解码过程所参考的数据（例如预测编码）不能越过slice的边界。

2. Slice的类型

根据码流中不同的数据类型，H.264标准中共定义了5总Slice类型：

- I slice: 帧内编码的条带；
- P slice: 单向帧间编码的条带；

- B slice: 双向帧间编码的条带；
- SI slice: 切换I条带，用于扩展档次中码流切换使用；
- SP slice: 切换P条带，用于扩展档次中码流切换使用；

在I slice中只包含I宏块，不能包含P或B宏块；在P和B slice中，除了相应的P和B类型宏块之外，还可以包含I类型宏块。

3. Slice的组成

每一个Slice总体来看都由两部分组成，一部分作为Slice header，用于保存Slice的总体信息（如当前Slice的类型等），另一部分为Slice body，通常是一组连续的宏块结构（或者宏块跳过信息），如下图所示：

4. Slice Header结构

Slice header中主要保存了当前slice的一些全局的信息，slice body中的宏块在进行解码时需依赖这些信息。其中比较常见的一些语法元素有：

1. **first_mb_in_slice**: 当前slice中包含的第一个宏块在整帧中的位置；
2. **slice_type**：当前slice的类型；
3. **pic_parameter_set_id**：当前slice所依赖的pps的id；
4. **colour_plane_id**:当标识位separate_colour_plane_flag为true时，colour_plane_id表示当前的颜色分量，0、1、2分别表示Y、U、V分量。
5. **frame_num**:表示当前帧序号的一种计量方式。
6. **field_pic_flag**:场编码标识位。当该标识位为1时表示当前slice按照场进行编码；该标识位为0时表示当前slice按照帧进行编码。
7. **bottom_field_flag**:底场标识位。该标志位为1表示当前slice是某一帧的底场；为0表示当前slice为某一帧的顶场。
8. **idr_pic_id**:表示IDR帧的序号。某一个IDR帧所属的所有slice，其idr_pic_id应保持一致。该值的取值范围为[0,65535]。
9. **pic_order_cnt_lsb**:表示当前帧序号的另一种计量方式。
10. **delta_pic_order_cnt_bottom**:表示顶场与底场POC差值的计算方法，不存在则默认为0；
11. **slice_qp_delta**：用于计算当前slice内所使用的初始qp值。

整个slice header的结构如下表所示：

slice_header( ) {	C	Descriptor
first_mb_in_slice	2	ue(v)
slice_type	2	ue(v)
pic_parameter_set_id	2	ue(v)
if(separate_colour_plane_flag == 1)		
colour_plane_id	2	u(2)
frame_num	2	u(v)
if(!frame_mbs_only_flag) {		
field_pic_flag	2	u(1)
if(field_pic_flag)		
bottom_field_flag	2	u(1)
}		
if(IdrPicFlag)		
idr_pic_id	2	ue(v)
if(pic_order_cnt_type == 0) {		
pic_order_cnt_lsb	2	u(v)
if(bottom_field_pic_order_in_frame_present_flag && !field_pic_flag)		
delta_pic_order_cnt_bottom	2	se(v)
}		
if(pic_order_cnt_type == 1 && !delta_pic_order_always_zero_flag) {		
delta_pic_order_cnt[0 **]**	2	se(v)
if(bottom_field_pic_order_in_frame_present_flag && !field_pic_flag)		
delta_pic_order_cnt[1 **]**	2	se(v)

}		
if(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	2	ue(v)
if(slice_type == B)		
direct_spatial_mv_pred_flag	2	u(1)
if(slice_type == P || slice_type == SP || slice_type == B) {		
num_ref_idx_active_override_flag	2	u(1)
if(num_ref_idx_active_override_flag) {		
num_ref_idx_l0_active_minus1	2	ue(v)
if(slice_type == B)		
num_ref_idx_l1_active_minus1	2	ue(v)
}		
}		
if(nal_unit_type == 20)		
ref_pic_list_mvc_modification( ) /* specified in Annex H */	2	
else		
ref_pic_list_modification( )	2	
if((weighted_pred_flag && (slice_type == P || slice_type == SP)) || (weighted_bipred_idc == 1 && slice_type == B))		
pred_weight_table( )	2	
if(nal_ref_idc != 0)		
dec_ref_pic_marking( )	2	
if(entropy_coding_mode_flag && slice_type != I && slice_type != SI)		
cabac_init_idc	2	ue(v)
slice_qp_delta	2	se(v)

}		
if(deblocking_filter_control_present_flag) {		
disable_deblocking_filter_idc	2	ue(v)
if(disable_deblocking_filter_idc != 1) {		
slice_alpha_c0_offset_div2	2	se(v)
slice_beta_offset_div2	2	se(v)
}		
}		
if(num_slice_groups_minus1 > 0 && slice_group_map_type >= 3 && slice_group_map_type <= 5)		
slice_group_change_cycle	2	u(v)
}		