GB

NATIONAL STANDARD OF THE

PEOPLE'S REPUBLIC OF CHINA

ICS 13.310
A 91

GB/T 25724-2017

Replacing GB/T 25724-2010

Technical specifications for surveillance video and audio coding

公共安全视频监控数字视音频编解码技术要求

Issued on: March 9, 2017 **Implemented on: June 1, 2017**

Issued by: General Administration of Quality Supervision, Inspection and Quarantine of the People's Republic of China;

Standardization Administration of the People's Republic of China.

Table of Contents

Technical specifications for surveillance video and audio coding

1 Scope

This Standard specifies the decoding process of digital video and audio compression coding for public security video surveillance.

This Standard applies to the audio and video real-time compression, transmission, playback and storage services of the field of public security; other fields that need audio and video coding may also refer to this Standard.

2 Normative reference

The following document is indispensable for the application of this document. For dated references, the only dated edition applies to this document. For undated references, the latest edition (including all modifications) applies to this document.

rfc 3548 The Base 16, Base 32, and Base 64 Data Encodings

3 Terms, definitions and abbreviations

3.1 Terms and definitions

For the purpose of this document, the following terms and definitions apply.

3.1.1

NAL unit

A syntax structure that contains the instruction type and the number of bytes contained in the subsequent data. The data appears in RBSP form and, if necessary, contains the scattered emulation prevention bytes.

3.1.2

NAL unit stream

A sequence of NAL units.

3.1.3

4.9 Process

The process is used to describe the decoding of the syntax elements. All the syntax elements and uppercase variables that belong to the current syntax structure, as well as the associated syntax structures, are available in both the specification and the call of the process. The specification of the process may also contain lowercase variables that are explicitly specified as input. Each specification explicitly specifies the output. The output can be uppercase variables or lowercase variables.

In the specification of the process, a particular macroblock can be represented by a variable name whose value is equal to its macroblock index.

5 Video section

5.1 Coded bitstream and output data format

5.1.1 Bitstream format

This clause specifies the relationship between the NAL unit stream and the byte stream, both of which are referred to as bitstreams.

The NAL unit stream format consists of a series of syntax structures called NAL units, arranged by decoding order. The decoding order and contents of the NAL units in the NAL unit stream are constrained.

The byte stream can be constructed by the NAL unit stream, by arranging the NAL units in the decoding order, and adding a start code prefix and a number of zero bytes to each NAL unit to form a byte stream. The NAL unit stream format can be extracted from the byte stream format by searching a unique start code prefix in the byte stream. Except for the byte stream format, other methods of constructing the NAL unit are not specified in this Standard. The byte stream format is specified in Annex B.

5.1.2 Picture format

This clause specifies the relationship between the source determined by the bitstream and the decoded frame.

The video stream represented by the bitstream is a series of frames arranged in decoding order.

Each source or decoded frame is composed of one or more video sample point arrays:

- array of only luma (Y) (monochrome);

- array of luma and two chroma (YCbCr);

The following functions are used for the syntax description. These functions assume that there is a bitstream pointer in the decoder that points to the next bit position in the bitstream where the decoding process is to be read. Specific requirements are as follows:

Specification for byte_aligned ():

- If the current position of the bitstream is at the boundary of the byte, that is, the next bit in the bitstream is the first bit of the byte, then the return value of byte_aligned () is TRUE;

- otherwise, the return value of byte_aligned () is FALSE.

Specification for get_left_ae_bits ():

- The value of the counter count in the entropy decoder plus 8 and then perform the modulo operation on 8, if it is equal to 0, continue to parse through the fixed probability of 128;

- if it is not equal to 0, then continue to parse through the fixed probability of 128 to obtain the value after modulo operation and plus 8 bits.

Specification for more_data_in_byte_stream (), which is used in the byte stream NAL unit syntax specified in Annex B:

- If there is more data in the byte stream, the return value of more_data_in_byte_stream () is TRUE;

- otherwise, the return value of more_data_in_byte_stream () is FALSE.

Specification for more_rbsp_data ():

- If there is more data in RBSP before rbsp_trailing_bits (), the return value of more_rbsp_data () is TRUE;

- otherwise, the return value of more_rbsp_data () is FALSE.

The method of determining whether there is more data in RBSP is not specified in this Standard.

next_bits (n) provides the next n bits in the bitstream without changing the bitstream pointer. This function makes the next n bits in the bitstream visible. When it is used in the byte stream specified in Annex B, the return value of next_bits (n) is 0 if the remaining byte stream has less than n bits.

Read_bits (n) reads the following n bits from the bitstream, and moves the bitstream pointer forward by n bits. When n is equal to 0, the return value of read_bits (n) is 0

may also contain some emulation_prevention_three_byte. NumBytesInNALunit is required for decoding the NAL unit. In order to be able to export NumBytesInNALunit, the boundary of the NAL unit needs to be divided. Annex B specifies a method for dividing the byte stream type. Other partition methods may be given outside this Standard.

forbidden_zero_bit indicates the version of the SVAC standard that the video stream supports. forbidden_zero_bit shall be equal to 1.

forbidden_zero_bit equal to 0 indicates that the video stream supports GB/T 25724-2010 standard.

When **nal_ref_idc** is not equal to 0, the contents of the NAL unit contain a sequence parameter set, or a picture parameter set, or a security parameter set, or tiles of a reference picture. When the nal_ref_idc of a tile NAL unit of a coded picture is equal to 0, the nal_ref_idc of all the tile NAL units of the coded picture shall be equal to 0.

nal_unit_type indicates the type of RBSP data structure in the NAL unit, see Table 30. The VCL NAL unit refers to those NAL units with the value of nal_unit_type equal to 1, 2, 3 or 4. All other NAL units are called non-VCL NAL units.

NOTE 1: The VCL specification is for effectively representing the contents of the video data. The NAL specification is for formatting the data and providing header information for storage or transmission over a variety of communication channels. Each NAL unit contains integer bytes. The NAL unit specifies a general format that applies to both packet-oriented and bitstream systems.

Without affecting the decoding process of NAL units with nal_unit_type not equal to 5 and without affecting the consistency of this Standard, NAL units with nal_unit_type equal to 5 can be discarded by the decoder.

NOTE 2: This Standard does not specify the decoding process of NAL units with the value nal_unit_type is reserved. The decoder can ignore (removed from the bitstream and discarded) all contents of NAL unit with the value nal_unit_type is reserved.

When the value of nal_unit_type of a tile NAL unit is equal to 2, the value of nal_unit_type of all other tile NAL units encoding the same picture shall be the same, and the value of nal_unit_type of all the tile NAL units of the corresponding SVC enhance layer coded picture shall be equal to 4. Such a picture is called an IDR picture.

The NAL unit type is as shown in Table 30.

Parameters included in a sequence parameter set RBSP can be used by one or more pictures or SEI NAL units containing buffer cycle SEI messages. Each sequence parameter set RBSP takes effect at the same time as it is received by the decoder, and the previously valid sequence parameter set RBSP (if any) will be fail. Not more than one sequence parameter set RBSP is valid at the specified time in the decoding process.

When a sequence parameter set RBSP is used by the SEI NAL unit containing a buffer cycle SEI message, the SEI NAL unit shall be located after the sequence parameter set RBSP.

Parameters included in the picture parameter set RBSP can be used by the coding tile NAL unit of a coded picture. Each picture parameter set RBSP takes effect at the same time as it is received by the decoder, and the previously valid sequence parameter set RBSP (if any) will be fail. For a layer picture of SVC, not more than one picture parameter set RBSP is valid at the specified time in the decoding process.

The specification for the relationship between the syntax element values and the other syntax elements in the sequence parameter sets and the picture parameter set is only for the valid sequence parameter set and the valid picture parameter set.

During the decoding process, the parameter values of the valid picture parameter set and the valid sequence parameter set shall remain valid.

5.2.4.3.2.2 Taking effect of security parameter set RBSP

Parameters included in the security parameter set RBSP can be used by one or more other types of NAL units. At the beginning of the decoding process, each security parameter set RBSP takes effect at the same time as it is received by the decoder, and the previously valid sequence parameter set RBSP (if any) will be fail. When ldp_mode_flag of the sequence parameter set is equal to 1, the security parameter set shall only appear before the IDR picture. Not more than one security parameter set RBSP is valid at the specified time in the decoding process.

NOTE: In some applications, the security parameter set can also be passed to the decoder via other reliable mechanisms.

5.2.4.3.2.3 Order of VCL NAL unit and its relationship with encoded pictures

Each VCL NAL unit is part of a coded picture.

The order of the VCL NAL units in a coded picture is defined as follows:

- the tile order of a picture shall be in ascending order of the first CTU index of the tile;

lf_mode_delta_enable [i] indicates that the mode related loop filter parameter difference update is enabled, equal to 0 indicates that the mode related loop filter parameter difference update is closed, equal to 1 indicates that the mode related loop filter parameter difference update is opened.

lf_mode_deltas [i] indicates the mode related loop filter parameter difference.

lf_mode_deltas_sign [i] indicates the sign of the mode related loop filter parameter difference.

$$mode\_deltas[i] = lf\_mode\_deltas[i] \times lf\_mode\_deltas\_sign[i]$$

picture_sao_enable [i] indicates whether the sample adaptive offset of the luma and chroma components is opened. picture_sao_enable [i] equal to 0 indicates that it does not open, equal to 1 indicates open, where i equal to 0 indicates the luma component; i equal to 1 or 2 indicates the chroma component.

picture_alf_enable [i] is the permission sign of picture adaptive loop filter, indicating whether the adaptive loop filter of the luma and chroma components of the current picture is opened. picture_alf_enable [i] equal to 0 indicates that the ith component of the current picture shall not use adaptive loop filter; equal to 1 indicates that the ith component of the current ipicture uses adaptive loop filter, where i equal to 0 indicates the luma component; i equal to 1 or 2 indicates the chroma component.

The value of **alf_filter_num_minus1** plus 1 indicates the number of current picture's luma component adaptive loop filter.

The value of alf_filter_num_minus1 shall be 0 to 15.

alf_region_distance [i] indicates the difference between the base unit start sign of luma component's ith adaptive loop filter region and the base unit start sign of luma component's i-1th adaptive loop filter region. The value of alf_region_distance [i] shall be 1 to 15.

If alf_region_distance [i] is not exist in the bitstream, when i is equal to 0, the value of alf_region_distance [i] is 0. when i is not equal to 0 and the value of alf_filter_num_minus1 is 15, the value of alf_region_distance [i] is 1. The bitstream shall satisfy that the sum of alf_region_distance [i] (i = 0 ~ alf_filter_num_minus1) is less than or equal to 15.

alf_coeff_luma [i] [j] indicates the jth coefficient of the luma component of the ith adaptive loop filter. The value range of alf_coeff_luma [i] [j] (j = 0 ~ 8) obtained from decoding in the bitstream shall be -64 to 63, and the value range of alf_coeff_luma [i] [9] shall be -1088 ~ 1071.

alf_coeff_chroma [0] [j] indicates the coefficient of the jth adaptive loop filter of the

sao_merge_flag equal to 0 indicates that the parameter is not merged; equal to 1 indicates that the parameter is merged, and the SAO parameter is the same as the SAO parameter of the CTU adjacent to its left or adjacent to its upper.

sao_merge_type equal to 1 indicates that the SAO parameter of the current CTU uses the SAO parameter of the adjacent CTU on the left; equal to 0 indicates that the SAO parameter of the current CTU uses the SAO parameter of the upper adjacent CTU on the upper side.

sao_mode [compIdx] equal to 0 indicates that the SAO mode of the compIdxth component in the current CTU is SAO_OFF; equal to 1 indicates that the SAO mode of the compIdxth component in the current CTU is determined by sao_type [compIdx].

sao_type [compIdx] equal to 0 indicates that the SAO mode of the compIdxth component in the current CTU is SAO_BO; equal to 1 indicates that the SAO mode of the compIdxth component in the current CTU is SAO_EO.

sao_start_band [compIdx] indicates the start compensation interval of the compIdxth component in the current CTU in SAO_BO mode, and the value shall be 0 ~ 31.

sao_offset_sign [compIdx] [j] indicates the sign of sao_offset [compIdx] [j] in SAO_BO mode. sao_offset_sign [compIdx] [j] equal to 0 indicates that the value of corresponding sao_offset [compIdx] [j] is positive, equal to 1 indicateds that the value of corresponding sao_offset [compIdx] [j] is negative.

sao_offset_abs [compIdx] [j] indicates the absolute value of the compensation value **sao_offset [compIdx] [j]** in SAO_BO mode, the value shall be 0 ~ (1 << (Min (bit_depth, 10) - 5)) - 1.

sao_edge_type [compIdx] indicates the angular direction of the compIdxth component in the current CTU in SAO_EO mode. sao_edge_type [compIdx] equal to 0 indicates EO_0°; equal to 1 indicates EO_90°; equal to 2 indicates EO_135°; equal to 3 indicates EO_45°.

sao_edge_offset [compIdx] [j] indicates the corresponding compensation value in SAO_EO mode.

alf_ctu_enable [compIdx] equal to 0 indicates that the compIdxth component of the current CTU does not perform adaptive loop filter. alf_ctu_enable [compIdx] equal to 1 indicates that the compIdxth component of the current CTU performs adaptive loop filter.

5.2.4.4.6 Authentication data RBSP semantics

frame_num indicates that the picture of authentication data shall be included; the picture is the same picture as the authentication data frame_num which is closest before the authentication data NAL unit. When frame_num is equal to 0, frame_num

inter_block indicates whether the current block is an inter coded block.

skip_flag indicates whether the current block is skipped.

coeff_value indicates the value of the block coefficients.

coeff_sign indicates the sign of the block coefficients.

tx_size indicates the size of the transform matrix used by the current block. tx_size equal to 0 indicates that the transform matrix is TX_4 × 4; equal to 1 indicates that the transform matrix is TX_8 × 8; equal to 2 indicates that the transform matrix is TX_16 × 16; equal to 3 indicates that the transform matrix is TX_32 × 32.

prev_intra_luma_pred_flag indicates whether the luma intra prediction mode is in the intra prediction mode prediction list and the prediction list contains 5 most likely prediction modes.

mpm_idx0 equal to 0 indicates that the current luma prediction mode is the first mode in the prediction list; equal to 1 indicates that the current luma prediction mode is not the first mode in the prediction list.

mpm_idx1, when mpm_idx0 is 1, mpm_idx1 + 1 indicates the position where the current prediction mode is in the prediction list. The value of mpm_idx1 shall be 0 ~ 3.

rem_pred_intra_mode indicates the index in the remaining 32 prediction modes except for the 5 prediction modes in the prediction list in the current luma prediction mode. The value of rem_pred_intra_mode shall be 0 ~ 31.

uv_fllow_y_flag equal to 1 indicates that the chroma intra prediction mode is consistent with the luma intra prediction mode of its corresponding position, and uv_fllow_y_flag equal to 0 indicates that the chroma intra prediction mode does not coincide with the luma intra prediction mode of its corresponding position.

chroma_intra_mode indicates the chroma intra prediction mode index.

block_reference_mode indicates the reference frame mode of the current block, the value is SINGLE_REFERENCE or COM-POUND_REFERENCE. If block_reference_mode does not exist in the code stream, the value of block_reference_mode is equal to frame_reference_mode. If block_reference_mode is equal to COMPOUND_REFERENCE, is_compound is equal to 1, otherwise is_compound is equal to 0.

ref_frame indicates the current prediction block reference frame index. When block_reference_mode is equal to SINGLE_REFERENCE, ref_frame has five possible values, namely DYNAMIC_REF, STATIC_REF, OPTIONAL_REF, DY-NAMIC_REF_1 and DYNAMIC_REF_2. When block_reference_mode is equal to

OSD extension information with two or more sub_type of the same value shall not appear in one NAL unit.

NOTE: The last extension information is valid when OSD extension information with more than one sub_type of the same value appears in the same NAL unit.

code_type is an 8-bit unsigned integer, representing the encoding type of the OSD character. The value of code_type equal to 0 indicates encoding with UTF-8.

align_type is an 8-bit unsigned integer, representing the alignment type of the OSD character. The value of align_type equal to 0 indicates left alignment; equal to 1 indicates right alignment.

char_size is an 8-bit unsigned integer, representing the OSD character size, expressed in sample units.

char_type is an 8-bit unsigned integer, representing the OSD character type. char_type equal to 0 indicates white background with black edges; equal to 1 indicates black background with white edges; equal to 2 indicates white; equal to 3 indicates black; equal to 4 indicates automatic anti-color.

top_low8 and **top_high8** form a 16-bit unsigned integer top, representing the position of the upper border of the OSD character information in the picture, expressed in sample points. The value of top is calculated as follows:

$$top = (top\_high8 << 8) + top\_low8$$

left_low8 and **top_high8** form a 16-bit unsigned integer left, representing the position of the left border of the OSD character information in the picture, expressed in sample points. The value of left is calculated as follows:

$$left = (left\_high8 << 8) + left\_low8$$

len is an 8-bit unsigned integer that indicates the length of the byte occupied by osd_data, which shall be 0 ~ 243.

res is an 8-bit unsigned integer, and the value shall be between 0 ~ 255.

osd_data is OSD character data. Where '\n' is defined as the row break and '\0' is the end character. The length of osd_data is len bytes.

5.2.4.7.5 Geographic information extension semantics

extension_id is an 8-bit unsigned integer, and the extension_id of the geographic information extension shall be equal to 0x10.

longitude_type equal to 0 indicates east longitude; equal to 1 indicates west longitude.

syntax structure encapsulated in the NAL unit. This process is extracting the RBSP syntax structure from the NAL unit. If encryption_idc is equal to 1, when extracting the RBSP syntax structure from the NAL unit, it needs to decrypt the encrypted RBSP to obtain an unencrypted RBSP. The decryption process is not specified in this Standard.

The RBSP syntax structure in the NAL unit is decoded in the following manner:

- The decoding process of NAL units when the value of nal_unit_type is 1, 2, 3 and 4, see 5.3.3;

- The intra prediction process of NAL units when the value of nal_unit_type is 1 and 2, see 5.3.4;

- The inter prediction process of NAL units when the value of nal_unit_type is 1, see 5.3.5;

- The decoding process and the picture reconstruction process for the coding tree unit in the NAL unit transforms the coefficient before the deblocking filter when the value of nal_unit_type is 1 and 2, see 5.3.6;

- The deblocking filter process of the reconstructed picture of the NAL unit when the value of nal_unit_type is 1, 2, 3 and 4, see 5.3.7;

- The offset compensation process of sample points of the reconstructed picture of the NAL unit when the value of nal_unit_type is 1, 2, 3 and 4, see 5.3.8;

- The filter compensation process of sample points of the reconstructed picture of the NAL unit when the value of nal_unit_type is 1, 2, 3 and 4, see 5.3.9;

- The decoding process of the coding tree unit in the NAL unit before the deblocking filter when the value of nal_unit_type is 3 and 4, see 5.3.10;

- When the value of nal_unit_type is 7, 8 and 9, the RBSP in the NAL unit is the sequence parameter set, the picture parameter set and the security parameter set, respectively. Effective sequence parameter set, picture parameter set and security parameter set are used in the decoding process of other NAL units;

- The decoding process of the NAL unit when the value of nal_unit_type is 13, see Clause 6;

- The decoding process of the NAL unit when the value of nal_unit_type is 0, 12, 14 and 15 is not specified in this Standard.

5.3.3 Decoding process of pictures

5.3.3.1 Classification and correspondence of pictures

For intra prediction, the prediction block size is bound to the transform block size; because there is only N × N transform, the prediction block size is also N × N. See 5.1.3.3 for adjacent block availability.

5.3.4.4 Acquisition of luma reference sample points

For N × N luma blocks, the reference sample point in the upper corner of the current block is marked as r [i], and the reference sample point in the left corner of the current block is marked as c [j], where r [0] = c [0].

Use *I* to represent the luma sample value matrix of the picture where the current block is after compensation (that is, before filter).

Let the coordinate of the sample point in the upper left corner of the current block in the picture be (x_0, y_0), and the reference sample is obtained by the following rules:

a) Initialize r [i], c [j] is $2^{bitdepth-1}$, i = 0 ~ 2N, j = 0 ~ 2N;

b) If the upper block is available, then r [i] = i [x_0 + i - 1, y_0 - 1], i = 1 ~ N, r [i] is available; otherwise r [i] is not available;

c) If the upper right block is available, then r [i] = i [x_0 + i - 1, y_0 - 1], I = N + 1 ~ 2N, r [i] is available; otherwise r [i] is equal to r [N], whether r [i] is available depends on whether r [N] is available;

d) If the left block is available, then c [j] = I [x_0 - 1, y_0 + j - 1], j = 1 ~ N, c [j] is available; otherwise c [j] is not available;

e) If the lower left block is available, then c [j] = I [x_0 - 1, y_0 + j - 1], j = N + 1 ~ 2N, c [j] is available; otherwise c [j] is equal to c [N], whether c [j] is available depends on whether c [N] is available;

f) If the sample point (x_0 - 1, y_0 - 1) is available, then r [0] = I [x_0 - 1, y_0 - 1], r [0] is available, otherwise r [0] is not available.

5.3.4.5 Acquisition of chroma reference sample positions

The acquisition method of chroma reference sample points is the same with that of luma reference sample points, except that the luma block becomes the corresponding chroma block.

5.3.4.6 Calculation of prediction sample positions

The prediction sample point matrix predMatrix of the luma blocks and chroma blocks in each intra prediction mode is exported as follows:

a) Horizontal_PRED

candidate motion vector set. When all the candidate positions are searched, enter the fourth step. Wherein the candidate position is the same as the first step.

The fourth step, if in the previous block in decoding order, a reference frame used by the block with the same position of the current block is different from the reference frame of the current block, then the MV of the corresponding reference frame of this block is added to the candidate motion vector set.

If this reference frame is in different direction with the reference frame of the current block, the MV sign of this block is negated (-mvx, -mvy) and added to the candidate motion vector set.

5.3.5.2.2 Export of luma motion vector

If the skip_flag of the current block is equal to 1, the MV of the current block is {0,0} and the corresponding reference frame is DYNAMIC_REF. otherwise:

if the block_reference_mode of the current block is equal to SINGLE_REFERENCE, then:

a) If the mv_mode of the current block is ZEROMV, the MV of the current block is {0,0};

b) If the mv_mode of the current block is NEARESTMV, the MV of the current block is PMV [0];

c) If the mv_mode of the current block is NEARMV, the MV of the current block is PMV [1];

d) If the mv_mode of the current block is NEWMV, the MV of the current block is MVP [0] + MVD [0].

If the block_reference_mode of the current block is equal to COMPOUND_REFERENCE, the current block is in bidirectional prediction mode, and there are two reference frames of inter prediction, where the first reference frame is read from the code stream and the second reference frame is fixed to OPTIONAL_REF. Two motion vectors are exported in two reference frames, namely MV [0] and MV [1] respectively, and are calculated as follows:

a) If the mv_mode of the current block is ZEROMV, both MV [0] and MV [1] are {0,0};

b) If the mv_mode of the current block is NEARESTMV, both MV [0] and MV [1] are PMV [0];

c) If the mv_mode of the current block is NEARMV, both MV [0] and MV [1] are PMV [1];

$ah_{0,0}$ = Clip1 (($ah'_{0,0}$ + 64) >> 7)

$ha_{0,0}$ = Clip1 (($ha'_{0,0}$ + 64) >> 7)

The fraction sample points of other chroma components, e.g.: $bb_{0,0}$, $bc_{0,0}$ ... $bh_{0,0}$, ... $hb_{0,0}$, $hc_{0,0}$, ... $hh_{0,0}$, need to be calculated using the fraction sample point value ($ab'_{0,0}$, $ac'_{0,0}$, $ad'_{0,0}$, $ae'_{0,0}$, $af'_{0,0}$, $ag'_{0,0}$, $ah'_{0,0}$) of the row where the integer sample points that have been calculated in the first step locate, the calculation method is as follows:

$hh'_{0,0} = -ah'_{0,-3} + 6 \times ah'_{0,-2} - 19 \times ah'_{0,-1} + 78 \times ah'_{0,0} + 78 \times ah'_{0,1} - 19 \times ah'_{0,2} + 6 \times ah'_{0,3} - ah'_{0,4}$

The final predictor of $hh_{0,0}$ is calculated as follows:

$hh_{0,0}$ = Clip1 (($hh'_{0,0}$ + 8192) >> 14)

Similarly, the prediction method of the other chroma sample positions is similar, and are calculated using the interpolation coefficients of the corresponding positions.

5.3.6 Transform factor decoding process and picture reconstruction process

5.3.6.1 General

This process is carried out before the deblocking filtering process, including block coefficient code stream, inverse scan, inverse quantization, inverse transform and reconstruction.

The input of the inverse scan is the array QuantCoeffArray generated by the block code stream, and the output is the quantization coefficient matrix QuantCoeffMatrix.

The input of the inverse quantization is the quantization coefficient matrix QuantCoeffMatrix, the quantization parameter QP of the current block, and the output is the inverse transform coefficient matrix CoeffMatrix.

The input of the inverse transform is 4 × 4 or 8 × 8 or 16 × 16 or 32 × 32 transform coefficient matrix CoeffMatrix, and output is 4 × 4 or 8 × 8 or 16 × 16 or 32 × 32 residue sample matrix ResidueMatrix.

The input of the reconstruction process is the residue sample matrix ResidueMatrix and the prediction sample matrix predMatrix, and the output is the reconstruction sample matrix RecMatrix.

5.3.6.2 Block coefficient code stream

Set max_eob as the number of samples in the current block, initialize the block coefficient array QuantCoeffArray to all zeros, and the block coefficient index i is equal to zero. The block code stream is performed as follows:

The first step, parse coeff_value [i], see 5.4.2 for the process, if the value is EOB, the block coefficient with the index value greater than or equal to i is 0, and the block code stream process ends, otherwise enter the next step;

The second step, if coeff_value [i] is equal to 0, then QuantCoeffArray [i] is equal to 0, the block coefficient index i is incremented by one, and continue to parse the next coeff_value [i] until an coeff_value [i] is greater than 0, then enter the next step;

The third step, if coeff_sign [i] is equal to 0, the value of the block coefficient QuantCoeffArray [i] is equal to coeff_value [i]; if coeff_sign [i] is equal to 1, the value of the block coefficient QuantCoeffArray [i] is equal to -coeff_value [i];

The fourth step, the block coefficient index i is incremented by one. If all the coefficients in the current block have been parsed, the block code stream process ends, otherwise return to the first step.

5.3.6.3 Reverse scan

Let the address of an element in the QuantCoeffArray array be k, find the column number i and row number j corresponding to the unit with address k by reverse scan, and then assign QuantCoeffArray [k] to QuantCoeffMatrix [i, j]. According to the block size and coding mode, the scan method is defined as shown in Figure 4.

When the coding block is coded by inter coding or when the coding block is a chroma block, according to the block size, respectively use 4 × 4 scan, 8 × 8 scan, 16 × 16 scan and 32 × 32 scan.

When the coding block is coded by intra coding and is a luma block:

1) If the value of luma_intra_mode is greater than or equal to 9 and is less than or equal to 17, according to the block size, respectively use 4 × 4 row scan, 8 × 8 row scan, 16 × 16 row scan and 32 × 32 scan;

2) If the value of luma_intra_mode is greater than or equal to 24 and is less than or equal to 31, according to the block size, respectively use 4 × 4 column scan, 8 × 8 column scan, 16 × 16 column scan and 32 × 32 scan;

3) Otherwise, according to the block size, respectively use 4 × 4 scan, 8 × 8 scan, 16 × 16 scan and 32 × 32 scan.

boundary, REMOVE the part E2 and E4 from the Figure 17) to obtain the region S2; otherwise MAKE S2 equal to S1;

- if the current coding tree unit C contains the samples at the rightmost column of the picture BUT does not contain the samples at the lowest row of the picture, EXTEND rightwards the right boundary of the region S2 to the right boundary of the picture (that is, the E5 (if any) in Figure 17, then the E6 is included into the current SAO region), to obtain the region S3;

- Otherwise, if the current coding tree unit C contains the samples of the lowest row of the picture BUT does not contain the samples of the rightmost column of the picture, EXTEND the lower boundary of the region S2 downwards to the boundary of the picture (that is, E7 (if any) in the Figure 17, then the E8 is included into the current SAO region), to obtain the region S3;

- Otherwise, if the current coding tree unit C contains both the samples of the rightmost column and the samples of the lowest row, EXTEND the right boundary of the region S2 rightwards to the right boundary of the picture, then EXTEND the lower boundary of the new region downwards to the lower boundary of the picture (the E5 (if any), E6, E7 (if any), E8, E9 are included into the current SAO region), to obtain the region S3;

- Otherwise, MAKE S3 equal to S2;

- USE the region S3 as the current sample adaptive offset unit region.

5.3.8.3 Sample adaptive offset information export

The following information is used for the sample adaptive offset process: parameter merge flag sao_merge_flag, merge type sao_merge_type, sample adaptive offset merge mode sao_merge_mode, sample adaptive offset mode sao_mode [compIdx], sample adaptive offset value sao_offset [compIdx] [J], sample adaptive offset edge mode type sao_edge_type [compIdx], and the start offset subinterval of sample offset offset interval mode type sao_start_band [compIdx], and so on.

If the current coding tree unit is neither the coding tree unit of the picture nor the coding tree unit of the left boundary of the Tile, the MergeLeftAvail is equal to 1; otherwise the MergeLeftAvail is equal to 0. If the current coding tree unit is not the coding tree unit of the upper boundary of the picture, the MergeUpAvail is equal to 1, otherwise MergeUpAvail is equal to 0.

In accordance with the value of MergeLeftAvail, MergeUpAvail, sao_merge_flag, and sao_merge_type, CHECK Table 56 to obtain the sample adaptive offset merge mode sao_merge_mode.

The residual sample of the enhance layer is obtained by inverse scanning, inverse quantization, and inverse transform of the residual coefficient matrix which is parsed from the code stream, with the detailed process as shown in 5.3.6.2 ~ 5.3.6.5.

The enhance layer prediction matrix is calculated as follows:

The reference picture set of the enhance layer shall correspond to the reference picture set of the base layer, that is, the same reference frame type corresponds to the enhance layer and the base layer of the same picture. In addition, it is judged whether it is cross-layer prediction or intra-layer prediction in accordance with the reference frame type of the enhance layer picture:

- If the reference frame is static_ref, the enhance layer uses cross-layer prediction. USE the base layer decoded picture which is enlarged by interpolation in accordance with 5.3.10.2 as the reference frame, and based on the motion vector to calculate the prediction matrix, wherein the calculation process of the motion vector is as shown in 5.3.10.4;

- If the reference frame type is not static_ref, the enhance layer adopts the intra-layer prediction AND the inter-frame prediction is same as those specified in 5.3.5, to obtain the prediction matrix.

The reconstruction process of the enhance layer is the same as 5.3.6.6.

5.3.10.4 Calculation of motion vectors for cross-layer prediction

In case of cross-layer prediction and when the width ratio and the height ratio of the enhance layer to the base layer are 2:1, the luma motion vector of each 8 x 8 block in every 16 x 16 block of the enlarged picture of the base layer is obtained by extending the luma motion vector of the corresponding 8 x 8 block in the base layer picture, as follows:

EL_MV = BL_MV x 2

EL_REF = BL_REF

Where EL_MV and EL_REF are the MV and reference frame types of the enhance layer; BL_MV and BL_REF are the MV and reference frame types of the base layer, respectively.

When the enhance layer is subjected to candidate motion vector set derivation AND when the width ratio and the height ratio of the enhance layer to the base layer is 2:1, INSERT one step in between the step 1 and the step 2 in clause 5.3.5.2.1: if the reference frame used in the block at the same position of the current block after the base layer enlarging is same as the reference frame of

- block_size: the size of the coded block;

- AvailU: the upper block is available if it is equal to 1; AND the upper block is unavailable if it is equal to 0;

- AvailL: the left block is available if it is equal to 1; AND the left block is unavailable if it is equal to 0;

- AboveIntra: the upper block is Intra if it is equal to 1; AND the upper block is non-Intra if it is equal to 0;

- LeftIntra: the left block is Intra if it is equal to 1; AND the left block is non-Intra if it is equal to 0;

- AboveSegIdPredicted: the seg_id_predicted of the upper block;

- LeftSegIdPredicted: the seg_id_predicted of the left block;

- AboveSkip: the upper block is skip if it is equal to 1; AND the upper block is non-skip if it is equal to 0;

- LeftSkip: the left block is skip if it is equal to 1; AND the left block is non-skip if it is equal to 0;

- maxTxSize: the maximum transform size;

- AboveTxSize: the transform size of the upper block;

- LeftTxSize: the transform size of the left block;

- idy: the vertical coordinates of the sub-block;

- idx: the horizontal coordinates of the sub-block;

- AbovePredMode: the prediction mode of the upper block;

- LeftPredMode: the prediction mode of the left block;

- AboveSingle: the upper block is in single reference mode if it is equal to 1; AND the upper block is in compound reference mode if it is equal to 0;

- LeftSingle: the left block is in single reference mode if it is equal to 1; AND the left block is in compound reference mode if it is equal to 0;

- CompFixedRef: the reference frame type obtained based on context when reference frame is predicted;

- CurRefFrame []: the reference frame for the current block;
- AboveNonzeroContext []: the context index of the non-zero coefficient of uplink;
- LeftNonzeroContext []: the context index of the non-zero coefficient of left column.

5.4.2.4.3 Calculation process of treeIdx

As for some of the syntax elements, in searching the probability table, it is required to use the treeIdx and the corresponding array to calculate the corresponding binIdx.

These syntax elements include: partition, segment_id, mv_joint, mvd_value_0 and mvd_value_1, interp_filter_mode, dqp_abs, and token.

partition:

When both hasRows and hasCols are equal to 1, USE the array partition_tree to calculate the treeIdx corresponding to binIdx.

partition_tree [6] = {-0, 2, -1, 4, -2, -3}

segment_id: USE the segment_tree:

segment_tree [14] = {2, 4, 6, 8, 10, 12, 0, -1, -2, -3, -4, -5, -6, -7}

interp_filter_mode: USE the interp_filter_tree:

Interp_filter_tree [6] = {-0, 2, 4, -2, -1, -3}

mv_joint: The tree selection depends on is_compound:

- If is_compound is equal to 1, SELECT the mv_joint_tree.
- Otherwise if is_compound is equal to 0, SELECT the mv_joint_tree2.

mv_joint_tree [6] = {-0, 2, -1, 4, -2, -3}

mv_joint_tree2 [4] = {-0, 2, -1, -2}

mvd_value_0 and **mvd_value_1** are mainly mv_class, mv_class0_fp and mv_fp, which requires using treeIdx to calculate the binIdx.

mv_class: USE the mv_class_tree:

a) The autocorrelation function value corresponding to this candidate value is greater than the autocorrelation function value corresponding to the previous candidate value, AND it is greater than the autocorrelation function value corresponding to the next candidate value;

b) As for the pitch period candidate value satisfying the condition a), SELECT up to six maximum values of the corresponding autocorrelation function (if the autocorrelation function value is the same, the smaller candidate value is preferred), ARRANGE and SAVE them in accordance with its corresponding autocorrelation function value from large to small, and SAVE its corresponding pitch period candidate sequence.

DETERMINE the ordered autocorrelation function sequence maxcorr [6] and the corresponding pitch period candidate value sequence peakpos [6] from the above two conditions.

6.2.3.4.1.2 Determination of pitch period global reference

INTRODUCE the pitch period global reference global_pitch to support judgment, so that the pitch period has smoothness. The pitch period global reference determination method is shown in Figure 30, as follows:

DETERMINE the pitch period candidate sequence peakpos [6] and the autocorrelation function sequence maxcorr [6] in accordance with 6.2.3.4.1.1.

Firstly, SELECT the pitch period candidate values that are close to the pitch reference global reference of the previous frame; MULTIPLY its corresponding autocorrelation function value by 1.2 for weighting. RE-ARRANGE the pitch period candidate values sequence peakpos [6] and the autocorrelation function sequence maxcorr [6].

Then ELIMINATE the pitch period doubling of the peakpos [6] and maxcorr [6]. The elimination of the doubling period is achieved by a fixed weighting method, in order to find an optimal pitch period candidate value, with the algorithm as follows:

a) SET the best pitch period candidate value to the pitch period candidate value corresponding to the maximum value of the autocorrelation function. CHECK the pitch period candidate sequence; SELECT one scaling factor of the autocorrelation function value for each pitch period candidate value. SELECT the scaling factor in accordance with the magnitude of the pitch period candidate value; the scaling factor is 1.2 when the pitch period candidate value is greater than the threshold value 25; otherwise, the scale factor is 1.11;

b) COMPARE the autocorrelation function value corresponding to this pitch period candidate value and the ratio of the maximum value in the autocorrelation function value sequence to the scaling factor, and if:

1) The currently considered pitch period candidate value is less than the current best pitch period;

2) The value of the autocorrelation function corresponding to the current pitch period candidate value is greater than the ratio of the maximum value and the scaling factor in the autocorrelation function sequence.

SET the best candidate value for the pitch period as the current pitch period candidate value. MAKE it cycle like this, until finishing the calculation of each pitch period candidate value in the pitch period candidate sequence;

c) DETERMINE whether the pitch period candidate value corresponding to the maximum value in the autocorrelation function sequence is the doubling of the current best pitch period candidate value. If it is, MAINTAIN the current pitch period best candidate value; otherwise, SET the pitch period candidate value corresponding to the maximum value of the autocorrelation function sequence as the best pitch period candidate value.

After the best pitch period candidate value is obtained, a reliable pitch period global reference is to be made for the determination. The algorithm for determining the global reference of the pitch period is:

a) If satisfying with one of the following four conditions, it may be able to determine the reliable pitch period reference:

1) In the pitch period candidate sequence, the maximum value of the autocorrelation function is not the doubling of the autocorrelation function value corresponding to the best pitch period candidate, AND the absolute value of the difference between the best pitch period candidate value and the cureent pitch period global reference (continuation of the previous frame) is less than 8;

2) The ratio of the maximum value to the other values in the autocorrelation function sequence is greater than 1.7;

3) The pitch period candidate value in the pitch period candidate value sequence is the doubling of the best pitch period candidate value;

4) The current pitch period global reference (continuation of the previous frame) is the doubling of the current best pitch period candidate value,

COUNT the number of pulses to be encoded on the track in accordance with the position, to determine the number of the pulsed positions, the distribution of the pulsed positions on the track, and the number of pulses on each pulsed position.

The pulse_num represents the total number of pulses to be coded, which is corresponding to the code rate mode, AND assuming that the pulse_num = ω. The pos_num presents the number of pulsed positions, assuming that pos_num = N. Since there may be a position overlap of the distribution of ω pulses in the track, there is obviously N ∈ [1, ω]. The pulsed position vector P (N) = {p (0), p (1), ..., p (N - 1)} represents the distribution of the pulsed position on the track. The p (n) represents the position sequence number of the pulsed positions on the track, n ∈ [0, N - 1], p [n] ∈ [0, M - 1], M = 16 represents the total number of positions on the track, and 0 (0) ≤ p (0) < p (1) <... < p (N - 1) ≤ 15. The pulse number vector SU (N) = {su (0), su (1), ..., su (N - 1)} represents the number of pulses at each pulsed position. The su (n) represents the number of pulses at the p (n) position, su (0) + su (1) + ... + su (N - 1) = ω.

It is also necessary to obtain pulse symbol information for each pulsed position when the pulses to be encoded on the track are counted in accordance with the position. The pulse symbol vector S (N) = {s (0), s (1), ..., s (N - 1)} represents the pulse symbol information of each pulsed position, and s (n) represents the pulse symbol at the p (n) position. USE s (n) = 0 to represent the positive pulses, and s (n) = 1 for to represent the negative pulse coding method.

6.2.3.4.8.3 First index I1 (classified by number of positions)

DETERMINE the first index I1 in accordance with the number of pulsed positions pos_num = N.

There are 16 positions on each track. When the number of pulses is 6, the number of the pulsed positions is 1, 2, 3, 4, 5, 6, respectively, and the corresponding first indexes are 0x1E0000, 0x1D0000, 0x1C0000, 0x08000, 0x100000, and 0x000000; when the number of coded pulses is 5, the number of pulsed positions is 1, 2, 3, 4, 5, and the corresponding first index is 0x00000, 0x08000, 0x10000, 0x20000, 0x40000; when the number of coded pulses is 4, the number of pulsed positions is 1, 2, 3, 4, and the corresponding first index is 0x0000, 0x2000, 0x4000, 0x8000; when the number of coded pulses is 3, the number of pulsed positions is 1, 2, and 3, and the corresponding first index is 0x1C00, 0x1800, 0x0000; when the number of coded pulses is 2, the number of pulsed positions is 1, 2, and the corresponding first index is 0x1E0,0x000; when the number of coded pulses is 1, the number of pulsed positions is 1, and the corresponding first index is 0x00.

6.2.3.4.8.4 Second index I2 (position code)

e) USE the scaling factor g_1 and g_2 to scale the sequence X. The scaled spectrum is X' = [X (0, 1, ..., N/2) / g_1, X (N/2 + 1, ..., N) / g_2], and finally X' is sent into the vector quantizer;

f) The g_1/g_2 is subjected to 7-bit coded transmission, AND in order to keep the number of coded bits unchanged, it shall reduce accordingly the number of available bits for quantization.

6.2.4.8 Variable length splitting table vector quantization

6.2.4.8.1 Variable length splitting table vector quantization process

The TVC uses the trellis quantizer to quantize the scaled spectrum X', which is divided into 8-dimensional vectors AND quantized using a vector code table consisting of a Gosset lattice set (also called RE_8). The lattice generation matrix G produces all the points in a lattice, c = kG, wherein k is the row vector consisting of integer values, AND c is the resulting grid. In order to generate a vector code table that satisfies a given code rate, only the lattice points on the sphere of given radius are considered. Using multiple different radii can generate a multi-rate code table.

The block diagram of the vector quantization method based on the splitting table is as shown in Figure 38. The variable header in the figure represents the header, the split header represents the header of the split amount, AND the even_flag represents the even flag. The vector quantization method based on splitting table is as follows:

a) FIND the nearest neighbor point y of the data x to be encoded in the lattice;

b) JUDGE whether y is in the base codebook C; if it is, then CALCULATE the index i directly, ENCODE and OUTPUT to the code stream;

c) If y is not in the base codebook C, JUDGE whether or not y belongs to $2D_8$. If y belongs to $2D_8$, it is subtracted by 1 and SET the even_flag at 1; if y does not belong to $2D_8$, USE the splitting table for encoding;

d) Then JUDGE whether y is in the base codebook C, if it is, then CALCULATE the index i directly, ENCODE and OUTPUT to the code stream; if not, USE the splitting table for encoding;

e) When splitting table is used for encoding, each component y (i) in y is splitted into the sum of c (i) and a value y' (i) in the splitting table. It shall select the appropriate y' (i) so that the absolute value of c (i) is minimized, AND the 8-dimensional vector as composed of the generated eight c (i) is a certain vector c in the base codebook C. After the splitting, CHECK the size of the eight y' (i) values again; if all of them is less than or equal to

In order to indicate whether the signature code word is hit, it is necessary to add the buffer flag bit in the code stream, which can be realized by modifying the flag bit header of the base codebook. The modified header is as follows:

Header = 0: The base codebook is Q_0

Header = 10: The base codebook is Q_2

Header = 1100: The base codebook is Q_3

Header = 1101: The data is the hit buffer data

Header = 1110: The base codebook is Q_4

Step 5: Coding output

As for the coding output of the buffer type data, if hit, then it outputs to the address in the buffer, as shown in Figure 43a) ~ b); if the leader 3 is hit, it will output the address number “3” of the leader 3 in the buffer; if it is not hit, it will calculate its codebook index as the output.

6.2.4.8.5 Encoding with splitting tables

If the data to be encoded is not in the base codebook (also includes $2D_8$ data and non-special signature code words that are still not in the base codebook after singular), USE the splitting table for encoding.

If a code word is not in the base codebook due to the relatively large absolute value of its component, the corresponding vector encoding method is as follows: the component of large absolute value is subtracted by a split amount to obtain a difference having a small enough absolute value, AND this difference is made to be a code word in the base codebook. The index of this code word in the base codebook AND the index of the split amount of this code word in the splitting table are encoded and output.

The encoding of the splitting table is as shown in Figure 38. In the extended processing of the splitting table, it includes two extended level encoding methods: level I extended encoding AND level II extended encoding. The encoding methods of the two extended levels are detailed as follows:

a) Level I extended encoding:

The level I extended encoding uses the equal number of bits to encode the split amount of each dimension. The level I extended splitting table is as shown in Table 159, with the split amount taken as 0 or 4. The third column in this table is the split amount index using the binary encoding,

formula (37) in 6.2.3.2.7, thus obtaining the analysis filter $\hat{A}_{HF}(z)$ for the high frequency signal.

b) Residual calculation

As for the low frequency signal x_{LF} (i, n), the low frequency residual e_{LF} (i, n) is obtained by quantizing the analysis filter $\hat{A}_{LF}(z)$;

As for the high frequency signal x_{HF} (i, n), the high frequency residual e_{HF} (i, n) is obtained by quantizing the analysis filter $\hat{A}_{HF}(z)$.

c) FFT transform

The low frequency residual e_{LF} (i, n) is subjected to FFT transform to obtain the low frequency residual spectrum E_{LF} (i, k), AND the FFT transform length is 64 points. The high frequency residual e_{HF} (i, n) is also subjected to the FFT transform to obtain the high frequency residual spectrum E_{HF} (i, k), AND the FFT transform length is also 64 points.

d) Spectrum copy

The low frequency signal uses 64 points as a subframe on time, AND equally divide the low frequency residual spectrum E_{FL} (i, k) into the low-low frequency residual spectrum E_{LFL} (i, k) and the low high frequency residual spectrum E_{LFH} (i, k) on frequency. Because the low-low frequency residual spectrum E_{LFL} (i, k) has a strong string, it is not used in the high frequency copy. BUT the low high frequency residual spectrum E_{LFH} (i, k) is reproduced twice to obtain the estimated high frequency residual spectrum $\tilde{E}_{HF}(i,k)$.

e) Time frequency band grid division of high frequency signals

When the low frequency encoder uses ACELP, the time frequency division of the high frequency signal always adopts the highest time resolution, that is, 64 points; the corresponding frequency resolution is the smallest, AND the time frequency division is as shown in Figure 48.

When the low frequency encoder uses TAC, the high frequency time frequency grid division depends on the superframe length. When the superframe length is 512, only the time frequency grid division structure of Figure 48 can be used.

A.3 Decoded picture buffer (DPB)

A.3.1 Picture output and removal

After the picture n is decoded, its DPB output time $t_{o,\ dpb}$ (n) is as follows:

$$t_{o,\ dpb}(n) = t_r(n) + t_c \times dpb\_output\_delay(n)$$

If there is no delay, then $t_{o,\ dpb}$ (n) = t_r (n), AND the current picture is output directly. If it is a reference picture, it is also required to store the current picture in DPB; otherwise ($t_{o,\ dpb}$ (n)> t_r (n)), the current picture output is delayed AND stored in the DPB.

A certain picture m in the DPB will be removed when the following two conditions are satisfied:

a) The picture m is not used as a reference picture for the subsequent decoded picture;

b) The DPB output time of the picture m is less than or equal to the removal time of the current picture n in the CPB, i.e., $t_{o,\ dpb}$ (m) <= t_r (n).

When all the data in the frame buffer is removed from the DPB, the DPB stuffing degree is subtracted by 1.

A.3.2 Insertion of IDR pictures

If the decoded picture is an IDR picture, the following operations are performed:

a) All pictures in the DPB are not used as reference pictures for subsequent decoded pictures;

b) When it is not the first IDR picture, AND the values of FrameWidth, FrameHeight and max_dec_frame_buffering as obtained in accordance with the sequence parameter set are different from those defined in the sequence parameter set corresponding to the preceding picture, all the frame buffers in the DPB are cleared AND not output, with DPB padding reset to zero.

A.3.3 Marking and storage of pictures

If the current picture is a reference picture, OR it is a non-reference picture but $t_{o,\ dpb}$ (n)> t_r (n), it is required to store the current picture in the DPB. If it requires storage, the current decoded picture is stored in an empty frame buffer, with the DPB stuffing degree plus 1.

The order of the NAL units in the byte stream NAL unit shall follow the NAL unit decoding order, as shown in 5.2.4.3.2.

The **leading_zero_8bits** shall be equal to 0x00.

Note: The leading_zero_8bits syntax element may only appear in the first byte stream NAL unit of the stream.

The **zero_byte** shall be equal to 0x00.

When any of the following conditions are met, there shall be a zero_byte syntax element.

- The nal_unit_type in the NAL unit is equal to 7 (sequence parameter set) or 8 (picture parameter set).

- The byte stream NAL unit contains the first NAL unit of a basic coded picture, as shown in 5.2.4.3.2.3.

The **start_code_prefix_one_3bytes** is a 3-byte fixed value sequence which is equal to 0x000001, AND this syntax element is called the start code prefix.

The **trailing_zero_8bits** shall be equal to 0x00.

B.3 Byte stream NAL unit decoding process

The input of this process is a byte stream, which consists of a series of byte stream NAL unit syntax structures.

The output of this process is a series of NAL units.

At the beginning of the decoding process, the decoder initializes its current location as the starting position of the byte stream. AND then it extracts and discards each leading_zero_8bits syntax element (if it exists) AND move the current position accordingly until the four bytes closely adjacent to the current position is 0x00000001.

The decoder, at this time, repeats the following step-by-step procedure to extract and decode each NAL unit syntax structure from the byte stream, until the last NAL unit in the byte stream has been decoded:

a) When the immediately adjacent four bytes in the byte stream form the four-byte sequence 0x00000001, to extract and discard the next byte in the bit stream (zero_byte syntax element), the current position of the byte stream is set as the byte position closely adjacent to the discarded byte;

Appendix C

(Normative)

Video profile and level

C.1 Overview

The profile and level specify the limits on the bit stream AND therefore the ability as required by the bit stream for decoding. Each profile defines a subset of the algorithm characteristics AND limits that all decoders consistent with this profile shall be supported. Each level defines a set of restrictions on the value of the syntax elements in this standard. The same level definition sets are used for all profiles, BUT individual applications may support different levels for supported profiles. In general, for a particular profile, the different levels correspond to the different requirements for the decoder load and the memory capacity.

This Appendix describes the various restrictions corresponding to the different profiles and levels of the video. All undefined syntax elements and parameters may take any values allowed by this standard. If a decoder can correctly decode all permissible values of the syntax elements specified for a profile and level, it is said that the decoder conforms to this standard at this profile and level. If there is no syntax element in a bit stream that is not allow by a certain profile and level, AND the value of the syntax element it contains does not exceed the range as allowed by this profile and level, the bit stream is considered to comply with this standard at this profile and level.

The profile_id and level_id define the profile and level of the bit stream.

Note: If the decoder falls between the values as specified in this standard due to the value of profile_id or level_id, it is not preferable to presume that the ability as represented by this value is between the specified profile and level.

C.2 Video profile

C.2.1 Definition of video profile

The definition of video profile is as shown in Table C.1.

that there is no HRD parameter in the subsequent code stream. Meanwhile, the values of the two temporary variables HrdBpPresentFlag (for the buffer period SEI message) and CpbDpbDelaysPresentFlag (for picture timing SEI messages) shall be the same as that of the hrd_parameters_present_flag.

The **low_delay_hrd_flag** defines an operation mode for HRD. When fixed_frame_rate_flag is equal to 1, low_delay_hrd_flag shall be equal to 0.

The **max_dec_frame_buffering** defines the maximum number of frame buffers for HRD DPB. Its value ranges from 0 to MaxDPB, or MaxDPB if there is no such syntax element in the code stream (SEE Appendix C).

D.2.2 Hypothetical reference decoder (HRD) parameter semantics

The **cpb_cnt_minus1** plus 1 defines the number of CPB parameters that can be selected. The value range of **cpb_cnt_minus1** is 0 ~ 31 (including 0 and 31). If low_delay_hrd_flag is equal to 1, cpb_cnt_minus1 shall be equal to 0. If there is no cpb_cnt_minus1 in the code stream, its value is defaulted to be equal to 0.

The two parameters **bit_rate_scale** and **bit_rate_value_minus1 [SchedSelIdx]** together define the maximum input bit rate for the SchedSelIdxth CPB. The value range of the bit_rate_value_minus1 [SchedSelIdx] is 0 ~ (2^{32} - 2) (including 0 and 2^{32} - 2), AND it becomes larger as the index number SchedSelIdx becomes larger. The code rate formula is:

BitRate [SchedSelIdx] = (bit_rate_value_minus1 [SchedSelIdx] +1) x $2^{(6 + bit\_rate\_scale)}$

If bit_rate_value_minus1 [SchedSelIdx] does not exist, its value is defaulted to be equal to 1200 x MaxBR.

The two parameters **cpb_size_scale** and **cpb_size_value_minus1 [SchedSelIdx]** together define the size of the CPB of the SchedSelIdxth CPB. The value range of the cpb_size_value_minus1 [SchedSelIdx] is 0 ~ (2^{32} - 2) (including 0 and 2^{32} - 2), AND it becomes larger as the index number SchedSelIdx becomes larger. The CPB size formula is:

CpbSize [SchedSelIdx] = (cpb_size_value_minus1 [SchedSelIdx] +1) x $2^{(4 + cpb\_size\_scale)}$

If cpb_size_value_minus1 [SchedSelIdx] does not exist, its value is defaulted to be equal to 1200 x MaxCPB.

The **cbr_flag [SchedSelIdx]** defines the HRD operation mode as defined by the SchedSelIdxth CPB parameter. If the value is equal to 0, it is a variable bit

If the application requires, the buffer period SEI message shall appear together with the IDR picture. This message provides the HRD initialization information.

The **initial_cpb_removal_delay** [SchedSelIdx] defines the initial delay of the Sched-SelIdxth CPB of the first buffer period after the HRD initialization, that is, the time from the first bit of the picture arriving to the CPB, to the time at which the data of this picture is started to be removed from CPB. The code word length is initial_cpb_removal_delay_length_minus1 + 1, AND in this standard, it specifies using 90kHz as its unit.

The value shall be greater than 0 AND less than 90000 x CpbSize [SchedSelIdx] ÷ BitRate [SchedSelIdx].

E.2.2 Picture timing SEI message definition

The **cpb_removal_delay** defines the waiting time of the picture associated to this SEI message before it is removed from the CPB. This value may also be used to calculate the earliest possible time for picture data to reach the CPB. The code word length is cpb_removal_delay_length_minus1 + 1.

For the first picture in the code stream, cpb_removal_delay shall be 0.

The **dpb_output_delay** defines the wait time for the picture data from being removed from the CPB to the time at which the decoded picture is output from DPB, which is used to calculate the DPB output time of the picture. The code word length is dpb_output_delay_length_minus1 + 1.

E.2.3 Panorama video parameter SEI message definition

The **spherical_pano_flag** spherical panorama video flag bit, 0 - other, 1 spherical panoramic video.

The **projection_type** projection mode flag bit, 0 - cube, 1 - cylinder, 2 - pyramid.

The **arrangement_raster_mode** indicates that when the projection_type = 0, that is, when the projection mode is a cube, the 6 square pictures (front, rear, left, right, top and bottom, with the index value in 1, 2, 3, 4, 5, and 6) adopt the raster arrangement method. The value range is: 0 ~ 719, AND there are 720 arrangements in total.

The **arrangement_matrix_mode** indicates the arrangement method of the 6 square pictures when projection_type = 0, that is, when the projection mode is cube. The value range is: 0 ~ 3, AND there are 4 array types in total: 0-1x6, 1-6x1, 2-2x3, 3-3x2.

The **object_speed_rad** [i] is a 9-bit unsigned integer that represents the direction of movement of the i^{th} object, in the unit of angle, the value range is [0, 359], the horizontal rightwards is 0, AND the angle is increased in case of counterclockwise rotation.

The **object_traipnt_x** [i] is a 16-bit unsigned integer that represents the X-coordinate of the trajectory of the i^{th} object, in the unit of sample values.

The **object_traipnt_y** [i] is a 16-bit unsigned integer that represents the Y-coordinate of the trajectory of the i^{th} object, in the unit of sample values.

The **object_traipnt_x** [i] and the **object_traipnt_y** [i] represent the abscissa and ordinate values of the trajectory of the i^{th} object, respectively. The abscissa and ordinate value of the trajectory position of the i^{th} object calculated using the sample as unit is as follows:

ObjectTraipntSamplePositionX [i] = object_traipnt_x [i]

ObjectTraipntSamplePositionY [i] = object_traipnt_y [i]

The **reserved_bit** shall be equal to 0.

F.2.3 Personnel attribute analysis

The **human_num** is an 8-bit unsigned integer that represents the number of person identified.

The **human_id** [i] is a 16-bit unsigned integer that represents the number of the i^{th} person identified.

The **human_property_num** [i] is an 8-bit unsigned integer that represents the number of attributes of the i^{th} person identified.

The **human_Property** [i, j] is an 8-bit unsigned integer that represents the j^{th} attribute of the i^{th} person identified.

F.2.4 Motor vehicle characteristics analysis

The **vehicle_num** is an 8-bit unsigned integer that represents the number of vehicles identified.

The **vehicle_id** [i] is a 16-bit unsigned integer that represents the identified vehicle number.

The **vehicle_property_num** [i] is an 8-bit unsigned integer that represents the number of attributes of the i^{th} vehicle identified.

The **vehicle_licence_modify_flag** [i] is a 1-bit unsigned integer indicating that the license plate is altered if it is equal to 1; AND that the license plate is not altered if it is equal to 0.

The **vehicle_licence_cover_flag** [i] is a 1-bit unsigned integer indicating that the license plate is covered if it is equal to 1; AND that the license plate is not covered if it is equal to 0.

The reserved_bit shall be equal to 0.

F.2.7 Pass line detection

The **pass_num** indicates the number of objects which pass the cordon.

The **object_category** indicates the category of the object which passes the cordon, if object_category is equal to 0, it is human; if object_category is equal to 1, it is vehicle; AND if object_category is equal to 2, it is other object.

The **object_size** indicates the size of the object which passes the cordon, if object_size is equal to 0, it indicates small size; if object_size is equal to 1, it indicates medium size; if object_size is equal to 2, it indicates large size; AND if object_size is equal to 3, it indicates huge size.

The moving_direction indicates the direction of movement of the object which passes the cordon, if the moving_direction is equal to 0, it indicates the north; if the moving_direction is equal to 1, it indicates the northeast; if the moving_direction is equal to 2, it indicates the east; if the moving_direction is equal to 3, it indicates the southeast; if the moving_direction is equal to 4, it indicates the south; if the moving_direction is equal to 5, it indicates the southwest; if the moving_direction is equal to 6, it indicates the west; if the moving_direction is equal to 7, it indicates the northwest; if the moving_direction is equal to 8, it indicates the upper side; if the moving_direction is equal to 9, it indicates the upper right side; if the moving_direction is equal to 10, it indicates the lower right side; if the moving_direction is equal to 11, it indicates the lower side; if the moving_direction is equal to 12, it indicates the lower left side; if the moving_direction is equal to 13, it indicates the upper left side; if the moving_direction is equal to 14, it indicates the left side; AND if the moving_direction is equal to 15, it indicates the right side.

The **position_top_left_x** [i] and the **position_top_left_y** [i] represents respectively the abscissa value and the ordinate value of the upper left corner of the i^{th} object which passes the cordon. The abscissa value and the ordinate value of the upper left corner position of the i^{th} object which passes the cordon using the sample as unit are as follows:

PassTopLeftSamplePositionX [i] = position_top_left_x [i]

PassTopLeftSamplePositionY [i] = position_top_left_y [i]

The **position_width_minus1** [i] plus 1 and the **position_height_minus1** [i] plus 1 are respectively equal to the width and height of the i^{th} object which passes the cordon. The width and height of the i^{th} object which passes the cordon as calculated using the sample as unit are as follows:

PassWidthInSample [i] = position_width_minus1 [i] +1

passHeightInSample[i] = position_height_minus1 [i] +1

F.2.8 Intrusion detection

The **invade_num** indicates the number of objects entering the forbidden area.

The **object_category** indicates the category of objects entering the forbidden area, if object_category is equal to 0, it is human; if object_category is equal to 1, it is vehicle; AND if object_category is equal to 2, it is other object.

The **object_size** indicates the size of the object entering the forbidden area, if object_size is equal to 0, it indicates small size; if object_size is equal to 1, it indicates medium size; if object_size is equal to 2, it indicates large size; AND if object_size is equal to 3, it indicates huge size.

The **moving_direction** indicates the direction of movement of the object entering the prohibited area, if the moving_direction is equal to 0, it indicates the north; if the moving_direction is equal to 1, it indicates the northeast; if the moving_direction is equal to 2, it indicates the east; if the moving_direction is equal to 3, it indicates the southeast; if the moving_direction is equal to 4, it indicates the south; if the moving_direction is equal to 5, it indicates the southwest; if the moving_direction is equal to 6, it indicates the west; if the moving_direction is equal to 7, it indicates the northwest; if the moving_direction is equal to 8, it indicates the upper side; if the moving_direction is equal to 9, it indicates the upper right side; if the moving_direction is equal to 10, it indicates the lower right side; if the moving_direction is equal to 11, it indicates the lower side; if the moving_direction is equal to 12, it indicates the lower left side; if the moving_direction is equal to 13, it indicates the upper left side; if the moving_direction is equal to 14, it indicates the left side; AND if the moving_direction is equal to 15, it indicates the right side.

The **position_top_left_x [i]** and the **position_top_left_y [i]** represent respectively the abscissa and ordinate values of upper left corner of the i^{th} object entering the forbidden area. The abscissa and ordinate values of the

This is an excerpt of the PDF (Some pages are marked off intentionally)

------ The End ------