

SUBPICTURE DEPENDENT SIGNALING IN VIDEO BITSTREAMS

CROSS REFERENCE TO RELATED APPLICATIONS

[001] Under the applicable patent law and/or rules pursuant to the Paris Convention, this application is made to timely claim the priority to and benefits of International Patent Application No. PCT/CN2019/100114, filed on August 10, 2019. For all purposes under the law, the entire disclosure of the aforementioned application is incorporated by reference as part of the disclosure of this application.

TECHNICAL FIELD

[002] This document is related to video and image coding and decoding technologies.

BACKGROUND

[003] Digital video accounts for the largest bandwidth use on the internet and other digital communication networks. As the number of connected user devices capable of receiving and displaying video increases, it is expected that the bandwidth demand for digital video usage will continue to grow.

SUMMARY

[004] The disclosed techniques may be used by video or image decoder or encoder embodiments for in which sub-picture based coding or decoding is performed.

[005] In one example aspect a method of video processing is disclosed. The method includes determining, for a conversion between a current block of a first picture of a video and a bitstream representation of the video, a motion candidate based on motion information from a second picture according to a rule. The rule specifies that a position from which the motion information is accessed is constrained to be within a specific subpicture of the second picture. The method also includes performing the conversion based on the determining.

[006] In another example aspect a method of video processing is disclosed. The method includes determining, for a conversion of a current block of a first picture of a video and a bitstream representation of the video, an integer sample from a second picture according to a rule. The second picture comprises a reference picture that is not used in an interpolation process. The rule specifies that a position from which the integer sample is accessed is constrained to be within a specific

subpicture of the second picture. The method also includes performing the conversion based on the determining.

[007] In another example aspect a method of video processing is disclosed. The method includes determining, for a conversion of a current block of a video and a bitstream representation of the video, a position at which a reconstructed luma sample is accessed according to a rule. The rule specifies that the position is constrained to be within a specific subpicture of a video picture. The method also includes performing the conversion based on the determining.

[008] In another example aspect a method of video processing is disclosed. The method includes determining, for a conversion of a current block of a video and a bitstream representation of the video, a position at which a picture boundary check is performed according to a rule. The rule specifies that the position is constrained to be within a specific subpicture of a video picture. The method also includes performing the conversion based on the determining.

[009] In another example aspect a method of video processing is disclosed. The method includes resetting, after a conversion of a sub-picture of a video picture of a video and a bitstream representation of the video, a table of motion candidates derived based on past conversions and performing a conversion of a subsequent sub-picture of the video picture and the bitstream representation using the table after the resetting.

[0010] In another example aspect a method of video processing is disclosed. The method includes performing a conversion between a video comprising a video picture that includes multiple sub-pictures and multiple video blocks and a coded representation of the video according to a rule. The rule specifies that a boundary between any two sub-pictures is also a boundary between two video blocks. A video block in the video picture is covered by a single subpicture of the video picture.

[0011] In another example aspect a method of video processing is disclosed. The method includes performing a conversion between a video unit of a video and a coded representation of the video using at least a video picture, where only one of a sub-picture coding mode or a resolution-changing coding mode is enabled for the video unit. The sub-picture coding mode is a mode in which the video picture is divided into multiple sub-pictures, and the resolution-changing coding mode is a mode in which a resolution of the video picture is adjusted during the conversion.

[0012] In another example aspect a method of video processing is disclosed. The method includes performing a conversion between a video unit of a video and a coded representation of the video using at least a video picture, where both a sub-picture coding mode and a resolution-changing

coding mode are enabled for the video unit. The sub-picture coding mode is a mode in which the video picture is divided into multiple sub-pictures, and the resolution-changing coding mode is a mode in which a resolution of the video picture is adjusted during the conversion.

[0013] In another example aspect a method of video processing is disclosed. The method includes performing a conversion between a video comprising one or more video pictures and a coded representation of the video, where a dimension of an individual video picture is constrained to be greater than or equal to 8. In some embodiments, the dimension is a width of the individual video picture.

[0014] In another example aspect a method of video processing is disclosed. The method includes performing a conversion between a video picture of a video and a coded representation of the video according to a rule. The video picture comprises at least one sub-picture, and the rule specifies that a characteristic of a sub-picture is represented as at least one syntax element in the coded representation, the at least one syntax element being different than an index value of the sub-picture grid in the video picture.

[0015] In another example aspect a method of video processing is disclosed. The method includes performing a conversion between a video picture of a video and a coded representation of the video according to a rule. The video picture comprises multiple sub-pictures, each sub-picture comprising multiple elements. The rule specifies that a dimension of individual elements in a sub-picture satisfies a constraint.

[0016] In another example aspect a method of video processing is disclosed. The method includes performing a conversion between a video comprising a picture that includes multiple sub-pictures and a coded representation of the video using a coding mode according to a rule. The rule specifies that certain stored information about a previous sub-picture is reset prior to processing each next sub-picture of the multiple sub-pictures.

[0017] In another example aspect a method of video processing is disclosed. The method includes performing a temporal filtering operation in a conversion between a video and a coded representation of the video according to a rule. The video comprises multiple video pictures, each comprising multiple sub-pictures. The rule specifies that, for a temporal filtering a current sample in a current sub-picture of a video picture, only samples within the same current sub-picture or a sub-picture in a different video picture corresponding to the current sub-picture are available.

[0018] In another example aspect a method of video processing is disclosed. The method includes

determining, for a conversion between a block in a video picture of a video and a coded representation of the video, a manner of applying a partitioning method to the block based on whether the block crosses one or more sub-picture boundaries of the video picture. The method also includes performing the conversion based on the determining.

[0019] In another example aspect a method of video processing is disclosed. The method includes for a conversion between a video picture of a video and a coded representation of the video, two sub-regions of the video picture. A first sub-region comprises multiple sub-pictures of the video picture and a second sub-region comprises remaining samples in the video picture. The method also includes performing the conversion based on the determining.

[0020] In another example aspect a method of video processing is disclosed. The method includes determining, for a video block in a first video region of a video, whether a position at which a temporal motion vector predictor is determined for a conversion between the video block and a bitstream representation of the current video block using an affine mode is within a second video region; and performing the conversion based on the determining.

[0021] In another example aspect, another method of video processing is disclosed. The method includes determining, for a video block in a first video region of a video, whether a position at which an integer sample in a reference picture is fetched for a conversion between the video block and a bitstream representation of the current video block is within a second video region, wherein the reference picture is not used in an interpolation process during the conversion; and performing the conversion based on the determining.

[0022] In another example aspect, another method of video processing is disclosed. The method includes determining, for a video block in a first video region of a video, whether a position at which a reconstructed luma sample value is fetched for a conversion between the video block and a bitstream representation of the current video block is within a second video region; and performing the conversion based on the determining.

[0023] In another example aspect, another method of video processing is disclosed. The method includes determining, for a video block in a first video region of a video, whether a position at which a check regarding splitting, depth derivation or split flag signaling for the video block is performed during a conversion between the video block and a bitstream representation of the current video block is within a second video region; and performing the conversion based on the determining.

[0024] In another example aspect, another method of video processing is disclosed. The method includes performing a conversion between a video comprising one or more video pictures comprising one or more video blocks, and a coded representation of the video, wherein the coded representation complies with a coding syntax requirement that the conversion is not to use sub-picture coding/decoding and a dynamic resolution conversion coding/decoding tool or a reference picture resampling tool within a video unit.

[0025] In another example aspect, another method of video processing is disclosed. The method includes performing a conversion between a video comprising one or more video pictures comprising one or more video blocks, and a coded representation of the video, wherein the coded representation complies with a coding syntax requirement that a first syntax element `subpic_grid_idx[i][j]` is not larger than a second syntax element `max_subpics_minus1`.

[0026] In yet another example aspect, the above-described method may be implemented by a video encoder apparatus that comprises a processor.

[0027] In yet another example aspect, the above-described method may be implemented by a video decoder apparatus that comprises a processor.

[0028] In yet another example aspect, these methods may be embodied in the form of processor-executable instructions and stored on a computer-readable program medium.

[0029] These, and other, aspects are further described in the present document.

BRIEF DESCRIPTION OF THE DRAWINGS

[0030] FIG. 1 shows an example of region constraint in temporal motion vector prediction (TMVP) and sub-block TMVP.

[0031] FIG. 2 shows an example of a hierarchical motion estimation scheme.

[0032] FIG. 3 is a block diagram of an example of a hardware platform used for implementing techniques described in the present document.

[0033] FIG. 4 is a flowchart for an example method of video processing.

[0034] FIG. 5 is a block diagram showing an example video processing system in which various techniques disclosed herein may be implemented.

[0035] FIG. 6 is a flowchart representation of a method for video processing in accordance with the present technology.

[0036] FIG. 7 is a flowchart representation of another method for video processing in accordance with the present technology.

[0037] FIG. 8 is a flowchart representation of another method for video processing in accordance with the present technology.

[0038] FIG. 9 is a flowchart representation of another method for video processing in accordance with the present technology.

[0039] FIG. 10 is a flowchart representation of another method for video processing in accordance with the present technology.

[0040] FIG. 11 is a flowchart representation of another method for video processing in accordance with the present technology.

[0041] FIG. 12 is a flowchart representation of another method for video processing in accordance with the present technology.

[0042] FIG. 13 is a flowchart representation of another method for video processing in accordance with the present technology.

[0043] FIG. 14 is a flowchart representation of another method for video processing in accordance with the present technology.

[0044] FIG. 15 is a flowchart representation of another method for video processing in accordance with the present technology.

[0045] FIG. 16 is a flowchart representation of another method for video processing in accordance with the present technology.

[0046] FIG. 17 is a flowchart representation of another method for video processing in accordance with the present technology.

[0047] FIG. 18 is a flowchart representation of another method for video processing in accordance with the present technology.

[0048] FIG. 19 is a flowchart representation of another method for video processing in accordance with the present technology.

[0049] FIG. 20 is a flowchart representation of yet another method for video processing in accordance with the present technology.

DETAILED DESCRIPTION

[0050] The present document provides various techniques that can be used by a decoder of image or video bitstreams to improve the quality of decompressed or decoded digital video or images. For brevity, the term “video” is used herein to include both a sequence of pictures (traditionally called video) and individual images. Furthermore, a video encoder may also implement these techniques during the process of encoding in order to reconstruct decoded frames used for further encoding.

[0051] Section headings are used in the present document for ease of understanding and do not limit the embodiments and techniques to the corresponding sections. As such, embodiments from one section can be combined with embodiments from other sections.

[0052] 1. Summary

[0053] This document is related to video coding technologies. Specifically, it is related to palette coding with employing base colors based representation in video coding. It may be applied to the existing video coding standard like HEVC, or the standard (Versatile Video Coding) to be finalized. It may be also applicable to future video coding standards or video codec.

[0054] 2. Initial Discussion

[0055] Video coding standards have evolved primarily through the development of the well-known ITU-T and ISO/IEC standards. The ITU-T produced H.261 and H.263, ISO/IEC produced MPEG-1 and MPEG-4 Visual, and the two organizations jointly produced the H.262/MPEG-2 Video and H.264/MPEG-4 Advanced Video Coding (AVC) and H.265/HEVC standards [1,2]. Since H.262, the video coding standards are based on the hybrid video coding structure wherein temporal prediction plus transform coding are utilized. To explore the future video coding technologies beyond HEVC, Joint Video Exploration Team (JVET) was founded by VCEG and MPEG jointly in 2015. Since then, many new methods have been adopted by JVET and put into the reference software named Joint Exploration Model (JEM). In April 2018, the Joint Video Expert Team (JVET) between VCEG (Q6/16) and ISO/IEC JTC1 SC29/WG11 (MPEG) was created to work on the VVC standard targeting at 50% bitrate reduction compared to HEVC.

[0056] 2.1 The region constraint in TMVP and sub-block TMVP in VVC

[0057] FIG. 1 illustrates example region constraint in TMVP and sub-block TMVP. In TMVP and sub-block TMVP, it is constrained that a temporal MV can only be fetched from the collocated CTU plus a column of 4×4 blocks as shown in FIG. 1.

[0058] 2.2 Example Sub-picture

[0059] In some embodiments, sub-picture-based coding techniques based on flexible tiling approach can be implemented. Summary of the sub-picture-based coding techniques includes the following:

[0060] (1) Pictures can be divided into sub-pictures.

[0061] (2) The indication of existence of sub-pictures is indicated in the SPS, along with other sequence-level information of sub-pictures.

[0062] (3) Whether a sub-picture is treated as a picture in the decoding process (excluding in-loop filtering operations) can be controlled by the bitstream.

[0063] (4) Whether in-loop filtering across sub-picture boundaries is disabled can be controlled by the bitstream for each sub-picture. The DBF, SAO, and ALF processes are updated for controlling of in-loop filtering operations across sub-picture boundaries.

[0064] (5) For simplicity, as a starting point, the sub-picture width, height, horizontal offset, and vertical offset are signalled in units of luma samples in SPS. Sub-picture boundaries are constrained to be slice boundaries.

[0065] (6) Treating a sub-picture as a picture in the decoding process (excluding in-loop filtering operations) is specified by slightly updating the `coding_tree_unit()` syntax, and updates to the following decoding processes:

[0066] - The derivation process for (advanced) temporal luma motion vector prediction

[0067] - The luma sample bilinear interpolation process

[0068] - The luma sample 8-tap interpolation filtering process

[0069] - The chroma sample interpolation process

[0070] (7) Sub-picture IDs are explicitly specified in the SPS and included in the tile group headers to enable extraction of sub-picture sequences without the need of changing VCL NAL units.

[0071] (8) Output sub-picture sets (OSPS) are proposed to specify normative extraction and conformance points for sub-pictures and sets thereof.

[0072] 2.3 Example Sub-pictures in Versatile Video Coding

Sequence parameter set RBSP syntax

seq_parameter_set_rbsp() {		Descriptor
sps_decoding_parameter_set_id		u(4)
sps_video_parameter_set_id		u(4)
...		

pic_width_max_in_luma_samples	ue(v)
pic_height_max_in_luma_samples	ue(v)
subpics_present_flag	u(1)
if(subpics_present_flag) {	
max_subpics_minus1	u(8)
subpic_grid_col_width_minus1	u(v)
subpic_grid_row_height_minus1	u(v)
for(i = 0; i < NumSubPicGridRows; i++)	
for(j = 0; j < NumSubPicGridCols; j++)	
subpic_grid_idx[i][j]	u(v)
for(i = 0; i <= NumSubPics; i++) {	
subpic_treated_as_pic_flag[i]	u(1)
loop_filter_across_subpic_enabled_flag[i]	u(1)
}	
}	
...	
}	

subpics_present_flag equal to 1 indicates that subpicture parameters are present in the present in the SPS RBSP syntax. **subpics_present_flag** equal to 0 indicates that subpicture parameters are not present in the present in the SPS RBSP syntax.

NOTE 2 – When a bitstream is the result of a sub-bitstream extraction process and contains only a subset of the subpictures of the input bitstream to the sub-bitstream extraction process, it might be required to set the value of **subpics_present_flag** equal to 1 in the RBSP of the SPSs.

max_subpics_minus1 plus 1 specifies the maximum number of subpictures that may be present in the CVS.

max_subpics_minus1 shall be in the range of 0 to 254. The value of 255 is reserved for future use by ITU-T | ISO/IEC.

subpic_grid_col_width_minus1 plus 1 specifies the width of each element of the subpicture identifier grid in units of 4 samples. The length of the syntax element is $\text{Ceil}(\text{Log}_2(\text{pic_width_max_in_luma_samples} / 4))$ bits.

The variable NumSubPicGridCols is derived as follows:

$$\text{NumSubPicGridCols} = (\text{pic_width_max_in_luma_samples} + \text{subpic_grid_col_width_minus1} * 4 + 3) / (\text{subpic_grid_col_width_minus1} * 4 + 4) \quad (7-5)$$

subpic_grid_row_height_minus1 plus 1 specifies the height of each element of the subpicture identifier grid in units of 4 samples. The length of the syntax element is $\text{Ceil}(\text{Log}_2(\text{pic_height_max_in_luma_samples} / 4))$ bits.

The variable NumSubPicGridRows is derived as follows:

$$\text{NumSubPicGridRows} = (\text{pic_height_max_in_luma_samples} + \text{subpic_grid_row_height_minus1} * 4 + 3) / (\text{subpic_grid_row_height_minus1} * 4 + 4) \quad (7-6)$$

subpic_grid_idx[i][j] specifies the subpicture index of the grid position (i, j). The length of the syntax element is $\text{Ceil}(\text{Log2}(\text{max_subpics_minus1} + 1))$ bits.

The variables **SubPicTop**[subpic_grid_idx[i][j]], **SubPicLeft**[subpic_grid_idx[i][j]], **SubPicWidth**[subpic_grid_idx[i][j]], **SubPicHeight**[subpic_grid_idx[i][j]], and **NumSubPics** are derived as follows:

```

NumSubPics = 0
for( i = 0; i < NumSubPicGridRows; i++ ) {
    for( j = 0; j < NumSubPicGridCols; j++ ) {
        if ( i == 0 )
            SubPicTop[ subpic_grid_idx[ i ][ j ] ] = 0
        else if( subpic_grid_idx[ i ][ j ] != subpic_grid_idx[ i - 1 ][ j ] ) {
            SubPicTop[ subpic_grid_idx[ i ][ j ] ] = i

            SubPicHeight[ subpic_grid_idx[ i - 1 ][ j ] ] = i - SubPicTop[ subpic_grid_idx[ i - 1 ][ j ] ]
        }
        if ( j == 0 )
            SubPicLeft[ subpic_grid_idx[ i ][ j ] ] = 0
        else if( subpic_grid_idx[ i ][ j ] != subpic_grid_idx[ i ][ j - 1 ] ) {
            SubPicLeft[ subpic_grid_idx[ i ][ j ] ] = j
            SubPicWidth[ subpic_grid_idx[ i ][ j ] ] = j - SubPicLeft[ subpic_grid_idx[ i ][ j - 1 ] ]
        }
        if ( i == NumSubPicGridRows - 1 )
            SubPicHeight[ subpic_grid_idx[ i ][ j ] ] = i - SubPicTop[ subpic_grid_idx[ i - 1 ][ j ] ] + 1
        if ( j == NumSubPicGridCols - 1 )
            SubPicWidth[ subpic_grid_idx[ i ][ j ] ] = j - SubPicLeft[ subpic_grid_idx[ i ][ j - 1 ] ] + 1
        if( subpic_grid_idx[ i ][ j ] > NumSubPics )
            NumSubPics = subpic_grid_idx[ i ][ j ]
    }
}

```

subpic_treated_as_pic_flag[i] equal to 1 specifies that the i-th subpicture of each coded picture in the CVS is treated as a picture in the decoding process excluding in-loop filtering operations. **subpic_treated_as_pic_flag**[i] equal to 0 specifies that the i-th subpicture of each coded picture in the CVS is not treated as a picture in the decoding process excluding in-loop filtering operations. When not present, the value of **subpic_treated_as_pic_flag**[i] is inferred to be equal to 0.

loop_filter_across_subpic_enabled_flag[i] equal to 1 specifies that in-loop filtering operations may be performed across the boundaries of the i-th subpicture in each coded picture in the CVS.

loop_filter_across_subpic_enabled_flag[i] equal to 0 specifies that in-loop filtering operations are not performed

across the boundaries of the i -th subpicture in each coded picture in the CVS. When not present, the value of `loop_filter_across_subpic_enabled_pic_flag[ i ]` is inferred to be equal to 1.

It is a requirement of bitstream conformance that the following constraints apply:

- For any two subpictures `subpicA` and `subpicB`, when the index of `subpicA` is less than the index of `subpicB`, any coded NAL unit of `subPicA` shall succeed any coded NAL unit of `subPicB` in decoding order.
- The shapes of the subpictures shall be such that each subpicture, when decoded, shall have its entire left boundary and entire top boundary consisting of picture boundaries or consisting of boundaries of previously decoded subpictures.

The list `CtbToSubPicIdx[ ctbAddrRs ]` for `ctbAddrRs` ranging from 0 to `PicSizeInCtbsY - 1`, inclusive, specifying the conversion from a CTB address in picture raster scan to a subpicture index, is derived as follows:

```

for( ctbAddrRs = 0; ctbAddrRs < PicSizeInCtbsY; ctbAddrRs++ ) {
    posX = ctbAddrRs % PicWidthInCtbsY * CtbSizeY
    posY = ctbAddrRs / PicWidthInCtbsY * CtbSizeY

    CtbToSubPicIdx[ ctbAddrRs ] = -1
    for( i = 0; CtbToSubPicIdx[ ctbAddrRs ] < 0 && i < NumSubPics; i++ ) {
        if( ( posX >= SubPicLeft[ i ] * ( subpic_grid_col_width_minus1 + 1 ) * 4 ) &&
            ( posX < ( SubPicLeft[ i ] + SubPicWidth[ i ] ) *
              ( subpic_grid_col_width_minus1 + 1 ) * 4 ) &&
            ( posY >= SubPicTop[ i ] *
              ( subpic_grid_row_height_minus1 + 1 ) * 4 ) &&
            ( posY < ( SubPicTop[ i ] + SubPicHeight[ i ] ) *
              ( subpic_grid_row_height_minus1 + 1 ) * 4 ) )
            CtbToSubPicIdx[ ctbAddrRs ] = i
    }
}

```

num_bricks_in_slice_minus1, when present, specifies the number of bricks in the slice minus 1. The value of `num_bricks_in_slice_minus1` shall be in the range of 0 to `NumBricksInPic - 1`, inclusive. When `rect_slice_flag` is equal to 0 and `single_brick_per_slice_flag` is equal to 1, the value of `num_bricks_in_slice_minus1` is inferred to be equal to 0. When `single_brick_per_slice_flag` is equal to 1, the value of `num_bricks_in_slice_minus1` is inferred to be equal to 0.

The variable `NumBricksInCurrSlice`, which specifies the number of bricks in the current slice, and

`SliceBrickIdx[ i ]`, which specifies the brick index of the i -th brick in the current slice, are derived as follows:

```

if( rect_slice_flag ) {
    sliceIdx = 0
    while( slice_address != slice_id[ sliceIdx ] )

```

```

        sliceIdx++
        NumBricksInCurrSlice = NumBricksInSlice[ sliceIdx ]
        brickIdx = TopLeftBrickIdx[ sliceIdx ]
        for( bIdx = 0; brickIdx <= BottomRightBrickIdx[ sliceIdx ]; brickIdx++ ) (7-92)
            if( BricksToSliceMap[ brickIdx ] == sliceIdx )
                SliceBrickIdx[ bIdx++ ] = brickIdx
            else {
                NumBricksInCurrSlice = num_bricks_in_slice_minus1 + 1
                SliceBrickIdx[ 0 ] = slice_address
                for( i = 1; i < NumBricksInCurrSlice; i++ )
                    SliceBrickIdx[ i ] = SliceBrickIdx[ i - 1 ] + 1
            }
    }

```

The variables SubPicIdx, SubPicLeftBoundaryPos, SubPicTopBoundaryPos, SubPicRightBoundaryPos, and SubPicBotBoundaryPos are derived as follows:

```

SubPicIdx = CtbToSubPicIdx[ CtbAddrBsToRs[ FirstCtbAddrBs[ SliceBrickIdx[ 0 ] ] ] ]
if( subpic_treated_as_pic_flag[ SubPicIdx ] ) {
    SubPicLeftBoundaryPos = SubPicLeft[ SubPicIdx ] * ( subpic_grid_col_width_minus1 + 1 ) * 4
    SubPicRightBoundaryPos = ( SubPicLeft[ SubPicIdx ] + SubPicWidth[ SubPicIdx ] ) *
        ( subpic_grid_col_width_minus1 + 1 ) * 4
    SubPicTopBoundaryPos = SubPicTop[ SubPicIdx ] * ( subpic_grid_row_height_minus1 + 1 ) * 4
    SubPicBotBoundaryPos = ( SubPicTop[ SubPicIdx ] + SubPicHeight[ SubPicIdx ] ) *
        ( subpic_grid_row_height_minus1 + 1 ) * 4
}
...

```

Derivation process for temporal luma motion vector prediction

Inputs to this process are:

- a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable cbWidth specifying the width of the current coding block in luma samples,
- a variable cbHeight specifying the height of the current coding block in luma samples,
- a reference index refIdxLX, with X being 0 or 1.

Outputs of this process are:

- the motion vector prediction mvLXCol in 1/16 fractional-sample accuracy,
- the availability flag availableFlagLXCol.

The variable currCb specifies the current luma coding block at luma location (xCb, yCb).

The variables mvLXCol and availableFlagLXCol are derived as follows:

- If slice_temporal_mvp_enabled_flag is equal to 0 or (cbWidth * cbHeight) is less than or equal to 32, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
- Otherwise (slice_temporal_mvp_enabled_flag is equal to 1), the following ordered steps apply:
 1. The bottom right collocated motion vector and the bottom and right boundary sample locations are derived as follows:

$$xColBr = xCb + cbWidth \quad (8-421)$$

$$yColBr = yCb + cbHeight \quad (8-422)$$

$$rightBoundaryPos = subpic_treated_as_pic_flag[SubPicIdx] ?$$

$$SubPicRightBoundaryPos : pic_width_in_luma_samples - 1 \quad (8-423)$$

$$botBoundaryPos = subpic_treated_as_pic_flag[SubPicIdx] ?$$

$$SubPicBotBoundaryPos : pic_height_in_luma_samples - 1 \quad (8-424)$$
 - If yCb >> CtbLog2SizeY is equal to yColBr >> CtbLog2SizeY, yColBr is less than or equal to botBoundaryPos and xColBr is less than or equal to rightBoundaryPos, the following applies:
 - The variable colCb specifies the luma coding block covering the modified location given by ((xColBr >> 3) << 3, (yColBr >> 3) << 3) inside the collocated picture specified by ColPic.
 - The luma location (xColCb, yColCb) is set equal to the top-left sample of the collocated luma coding block specified by colCb relative to the top-left luma sample of the collocated picture specified by ColPic.
 - The derivation process for collocated motion vectors as specified in clause 8.5.2.12 is invoked with currCb, colCb, (xColCb, yColCb), refIdxLX and sbFlag set equal to 0 as inputs, and the output is assigned to mvLXCol and availableFlagLXCol.

Otherwise, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.

...

Luma sample bilinear interpolation process

Inputs to this process are:

- a luma location in full-sample units (xInt_L, yInt_L),
- a luma location in fractional-sample units (xFrac_L, yFrac_L),
- the luma reference sample array refPicLX_L.

Output of this process is a predicted luma sample value predSampleLX_L.

The variables shift1, shift2, shift3, shift4, offset1, offset2 and offset3 are derived as follows:

$$shift1 = BitDepth_Y - 6 \quad (8-453)$$

$$offset1 = 1 << (shift1 - 1) \quad (8-454)$$

$$shift2 = 4 \quad (8-455)$$

$$offset2 = 1 << (shift2 - 1) \quad (8-456)$$

$$shift3 = 10 - BitDepth_Y \quad (8-457)$$

$$shift4 = BitDepth_Y - 10 \quad (8-458)$$

$$\text{offset4} = 1 \ll (\text{shift4} - 1) \quad (8-459)$$

The variable picW is set equal to pic_width_in_luma_samples and the variable picH is set equal to pic_height_in_luma_samples.

The luma interpolation filter coefficients $\text{fb}_L[p]$ for each 1/16 fractional sample position p equal to xFrac_L or yFrac_L are specified in Table 8-10.

The luma locations in full-sample units (xInt_i , yInt_i) are derived as follows for $i = 0..1$:

- If subpic_treated_as_pic_flag[SubPicIdx] is equal to 1, the following applies:

$$\text{xInt}_i = \text{Clip3}(\text{SubPicLeftBoundaryPos}, \text{SubPicRightBoundaryPos}, \text{xInt}_L + i) \quad (8-460)$$

$$\text{yInt}_i = \text{Clip3}(\text{SubPicTopBoundaryPos}, \text{SubPicBotBoundaryPos}, \text{yInt}_L + i) \quad (8-461)$$

- Otherwise (subpic_treated_as_pic_flag[SubPicIdx] is equal to 0), the following applies:

$$\begin{aligned} \text{xInt}_i &= \text{Clip3}(0, \text{picW} - 1, \text{sps_ref_wraparound_enabled_flag} \text{ ?} \\ &\quad \text{ClipH}((\text{sps_ref_wraparound_offset_minus1} + 1) * \text{MinCbSizeY}, \text{picW}, (\text{xInt}_L + i)) : (8-462) \\ &\quad \text{xInt}_L + i) \end{aligned}$$

$$\text{yInt}_i = \text{Clip3}(0, \text{picH} - 1, \text{yInt}_L + i) \quad (8-463)$$

...

Derivation process for subblock-based temporal merging candidates

Inputs to this process are:

- a luma location (xCb , yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable cbWidth specifying the width of the current coding block in luma samples,
- a variable cbHeight specifying the height of the current coding block in luma samples.
- the availability flag availableFlagA₁ of the neighbouring coding unit,
- the reference index refIdxLXA₁ of the neighbouring coding unit with X being 0 or 1,
- the prediction list utilization flag predFlagLXA₁ of the neighbouring coding unit with X being 0 or 1,
- the motion vector in 1/16 fractional-sample accuracy mvLXA₁ of the neighbouring coding unit with X being 0 or 1.

Outputs of this process are:

- the availability flag availableFlagSbCol,
- the number of luma coding subblocks in horizontal direction numSbX and in vertical direction numSbY,
- the reference indices refIdxL0SbCol and refIdxL1SbCol,
- the luma motion vectors in 1/16 fractional-sample accuracy mvL0SbCol[xSbIdx][ySbIdx] and mvL1SbCol[xSbIdx][ySbIdx] with xSbIdx = 0..numSbX - 1, ySbIdx = 0.. numSbY - 1,
- the prediction list utilization flags predFlagL0SbCol[xSbIdx][ySbIdx] and predFlagL1SbCol[xSbIdx][ySbIdx] with xSbIdx = 0..numSbX - 1, ySbIdx = 0.. numSbY - 1.

The availability flag availableFlagSbCol is derived as follows.

- If one or more of the following conditions is true, availableFlagSbCol is set equal to 0.
 - slice_temporal_mvp_enabled_flag is equal to 0.

- sps_sbtmvp_enabled_flag is equal to 0.
- cbWidth is less than 8.
- cbHeight is less than 8.
- Otherwise, the following ordered steps apply:
 1. The location (xCtb, yCtb) of the top-left sample of the luma coding tree block that contains the current coding block and the location (xCtr, yCtr) of the below-right center sample of the current luma coding block are derived as follows:

$$xCtb = (xCb \gg CtuLog2Size) \ll CtuLog2Size \quad (8-542)$$

$$yCtb = (yCb \gg CtuLog2Size) \ll CtuLog2Size \quad (8-543)$$

$$xCtr = xCb + (cbWidth / 2) \quad (8-544)$$

$$yCtr = yCb + (cbHeight / 2) \quad (8-545)$$
 2. The luma location (xColCtrCb, yColCtrCb) is set equal to the top-left sample of the collocated luma coding block covering the location given by (xCtr, yCtr) inside ColPic relative to the top-left luma sample of the collocated picture specified by ColPic.
 3. The derivation process for subblock-based temporal merging base motion data as specified in clause 8.5.5.4 is invoked with the location (xCtb, yCtb), the location (xColCtrCb, yColCtrCb), the availability flag availableFlagA₁, and the prediction list utilization flag predFlagLXA₁, and the reference index refIdxLXA₁, and the motion vector mvLXA₁, with X being 0 and 1 as inputs and the motion vectors ctrMvLX, and the prediction list utilization flags ctrPredFlagLX of the collocated block, with X being 0 and 1, and the temporal motion vector tempMv as outputs.
 4. The variable availableFlagSbCol is derived as follows:
 - If both ctrPredFlagL0 and ctrPredFlagL1 are equal to 0, availableFlagSbCol is set equal to 0.
 - Otherwise, availableFlagSbCol is set equal to 1.

When availableFlagSbCol is equal to 1, the following applies:

- The variables numSbX, numSbY, sbWidth, sbHeight and refIdxLXSbCol are derived as follows:

$$\text{numSbX} = \text{cbWidth} \gg 3 \quad (8-546)$$

$$\text{numSbY} = \text{cbHeight} \gg 3 \quad (8-547)$$

$$\text{sbWidth} = \text{cbWidth} / \text{numSbX} \quad (8-548)$$

$$\text{sbHeight} = \text{cbHeight} / \text{numSbY} \quad (8-549)$$

$$\text{refIdxLXSbCol} = 0 \quad (8-550)$$
- For xSbIdx = 0..numSbX – 1 and ySbIdx = 0 .. numSbY – 1, the motion vectors mvLXSbCol[xSbIdx][ySbIdx] and prediction list utilization flags predFlagLXSbCol[xSbIdx][ySbIdx] are derived as follows:
 - The luma location (xSb, ySb) specifying the top-left sample of the current coding subblock relative to the top-left luma sample of the current picture is derived as follows:

$$xSb = xCb + xSbIdx * sbWidth + sbWidth / 2 \quad (8-551)$$

$$ySb = yCb + ySbIdx * sbHeight + sbHeight / 2 \quad (8-552)$$

- The location (xColSb, yColSb) of the collocated subblock inside ColPic is derived as follows.
 - The following applies:

$$\begin{aligned} yColSb &= Clip3(yCtb, \\ &\quad Min(CurPicHeightInSamplesY - 1, yCtb + (1 \ll CtbLog2SizeY) - 1), (8-553) \\ &\quad ySb + (tempMv[1] \gg 4)) \end{aligned}$$
 - If subpic_treated_as_pic_flag[SubPicIdx] is equal to 1, the following applies:

$$\begin{aligned} xColSb &= Clip3(xCtb, \\ &\quad Min(SubPicRightBoundaryPos, xCtb + (1 \ll CtbLog2SizeY) + 3), (8-554) \\ &\quad xSb + (tempMv[0] \gg 4)) \end{aligned}$$
 - Otherwise (subpic_treated_as_pic_flag[SubPicIdx] is equal to 0), the following applies:

$$\begin{aligned} xColSb &= Clip3(xCtb, \\ &\quad Min(CurPicWidthInSamplesY - 1, xCtb + (1 \ll CtbLog2SizeY) + 3), (8-555) \\ &\quad xSb + (tempMv[0] \gg 4)) \end{aligned}$$

...

Derivation process for subblock-based temporal merging base motion data

Inputs to this process are:

- the location (xCtb, yCtb) of the top-left sample of the luma coding tree block that contains the current coding block,
- the location (xColCtCb, yColCtCb) of the top-left sample of the collocated luma coding block that covers the below-right center sample.
- the availability flag availableFlagA₁ of the neighbouring coding unit,
- the reference index refIdxLXA₁ of the neighbouring coding unit,
- the prediction list utilization flag predFlagLXA₁ of the neighbouring coding unit,
- the motion vector in 1/16 fractional-sample accuracy mvLXA₁ of the neighbouring coding unit.

Outputs of this process are:

- the motion vectors ctrMvL0 and ctrMvL1,
- the prediction list utilization flags ctrPredFlagL0 and ctrPredFlagL1,
- the temporal motion vector tempMv.

The variable tempMv is set as follows:

$$tempMv[0] = 0 \quad (8-558)$$

$$tempMv[1] = 0 \quad (8-559)$$

The variable currPic specifies the current picture.

When availableFlagA₁ is equal to TRUE, the following applies:

- If all of the following conditions are true, tempMv is set equal to mvL0A₁:
 - predFlagL0A₁ is equal to 1,
 - DiffPicOrderCnt(ColPic, RefPicList[0][refIdxL0A₁]) is equal to 0,
- Otherwise, if all of the following conditions are true, tempMv is set equal to mvL1A₁:

- slice_type is equal to B,
- predFlagL1A₁ is equal to 1,
- DiffPicOrderCnt(ColPic, RefPicList[1][refIdxL1A₁]) is equal to 0.

The location (xColCb, yColCb) of the collocated block inside ColPic is derived as follows.

- The following applies:

$$\begin{aligned} yColCb &= Clip3(yCtb, \\ &\quad Min(CurPicHeightInSamplesY - 1, yCtb + (1 \ll CtbLog2SizeY) - 1), (8-560) \\ &\quad yColCtrCb + (tempMv[1] \gg 4)) \end{aligned}$$

- If subpic_treated_as_pic_flag[SubPicIdx] is equal to 1, the following applies:

$$\begin{aligned} xColCb &= Clip3(xCtb, \\ &\quad Min(SubPicRightBoundaryPos, xCtb + (1 \ll CtbLog2SizeY) + 3), (8-561) \\ &\quad xColCtrCb + (tempMv[0] \gg 4)) \end{aligned}$$

- Otherwise (subpic_treated_as_pic_flag[SubPicIdx] is equal to 0, the following applies:

$$\begin{aligned} xColCb &= Clip3(xCtb, \\ &\quad Min(CurPicWidthInSamplesY - 1, xCtb + (1 \ll CtbLog2SizeY) + 3), (8-562) \\ &\quad xColCtrCb + (tempMv[0] \gg 4)) \end{aligned}$$

...

Luma sample interpolation filtering process

Inputs to this process are:

- a luma location in full-sample units (xInt_L, yInt_L),
- a luma location in fractional-sample units (xFrac_L, yFrac_L),
- a luma location in full-sample units (xSbInt_L, ySbInt_L) specifying the top-left sample of the bounding block for reference sample padding relative to the top-left luma sample of the reference picture,
- the luma reference sample array refPicLX_L,
- the half sample interpolation filter index hpelIfIdx,
- a variable sbWidth specifying the width of the current subblock,
- a variable sbHeight specifying the height of the current subblock,
- a luma location (xSb, ySb) specifying the top-left sample of the current subblock relative to the top-left luma sample of the current picture,

Output of this process is a predicted luma sample value predSampleLX_L

The variables shift1, shift2 and shift3 are derived as follows:

- The variable shift1 is set equal to Min(4, BitDepth_Y - 8), the variable shift2 is set equal to 6 and the variable shift3 is set equal to Max(2, 14 - BitDepth_Y).
- The variable picW is set equal to pic_width_in_luma_samples and the variable picH is set equal to pic_height_in_luma_samples.

The luma interpolation filter coefficients f_L[p] for each 1/16 fractional sample position p equal to xFrac_L or yFrac_L are derived as follows:

- If MotionModelIdx[xSb][ySb] is greater than 0, and sbWidth and sbHeight are both equal to 4, the luma interpolation filter coefficients $f_L[p]$ are specified in Table 8-12.

- Otherwise, the luma interpolation filter coefficients $f_L[p]$ are specified in Table 8-11 depending on hpelIfIdx.

The luma locations in full-sample units ($xInt_i$, $yInt_i$) are derived as follows for $i = 0..7$:

- If subpic_treated_as_pic_flag[SubPicIdx] is equal to 1, the following applies:

$$xInt_i = \text{Clip3}(\text{SubPicLeftBoundaryPos}, \text{SubPicRightBoundaryPos}, xInt_L + i - 3) \quad (8-771)$$

$$yInt_i = \text{Clip3}(\text{SubPicTopBoundaryPos}, \text{SubPicBotBoundaryPos}, yInt_L + i - 3) \quad (8-772)$$

- Otherwise (subpic_treated_as_pic_flag[SubPicIdx] is equal to 0), the following applies:

$$xInt_i = \text{Clip3}(0, \text{picW} - 1, \text{sps_ref_wraparound_enabled_flag} ? \text{ClipH}((\text{sps_ref_wraparound_offset_minus1} + 1) * \text{MinCbSizeY}, \text{picW}, xInt_L + i - 3) : (8-773) \\ xInt_L + i - 3)$$

$$yInt_i = \text{Clip3}(0, \text{picH} - 1, yInt_L + i - 3) \quad (8-774)$$

...

Chroma sample interpolation process

Inputs to this process are:

- a chroma location in full-sample units ($xInt_C$, $yInt_C$),
- a chroma location in 1/32 fractional-sample units ($xFrac_C$, $yFrac_C$),
- a chroma location in full-sample units ($xSbInt_C$, $ySbInt_C$) specifying the top-left sample of the bounding block for reference sample padding relative to the top-left chroma sample of the reference picture,
- a variable sbWidth specifying the width of the current subblock,
- a variable sbHeight specifying the height of the current subblock,
- the chroma reference sample array refPicLXC.

Output of this process is a predicted chroma sample value predSampleLXC

The variables shift1, shift2 and shift3 are derived as follows:

- The variable shift1 is set equal to $\text{Min}(4, \text{BitDepth}_C - 8)$, the variable shift2 is set equal to 6 and the variable shift3 is set equal to $\text{Max}(2, 14 - \text{BitDepth}_C)$.
- The variable picWC is set equal to $\text{pic_width_in_luma_samples} / \text{SubWidthC}$ and the variable picHC is set equal to $\text{pic_height_in_luma_samples} / \text{SubHeightC}$.

The chroma interpolation filter coefficients $f_C[p]$ for each 1/32 fractional sample position p equal to $xFrac_C$ or $yFrac_C$ are specified in Table 8-13.

The variable xOffset is set equal to $(\text{sps_ref_wraparound_offset_minus1} + 1) * \text{MinCbSizeY} / \text{SubWidthC}$.

The chroma locations in full-sample units ($xInt_i$, $yInt_i$) are derived as follows for $i = 0..3$:

- If subpic_treated_as_pic_flag[SubPicIdx] is equal to 1, the following applies:

$$xInt_i = \text{Clip3}(\text{SubPicLeftBoundaryPos} / \text{SubWidthC}, \text{SubPicRightBoundaryPos} / \text{SubWidthC}, xInt_L + i) \quad (8-785)$$

$$yInt_i = \text{Clip3}(\text{SubPicTopBoundaryPos} / \text{SubHeightC}, \text{SubPicBotBoundaryPos} / \text{SubHeightC}, yInt_L + i) \quad (8-786)$$

- Otherwise (subpic_treated_as_pic_flag[SubPicIdx] is equal to 0), the following applies:

$$\begin{aligned} xInt_i &= \text{Clip3}(0, \text{picW}_C - 1, \text{sps_ref_wraparound_enabled_flag} ? \text{ClipH}(xOffset, \text{picW}_C, xInt_C + i - 1) : (8-787) \\ &\quad xInt_C + i - 1) \\ yInt_i &= \text{Clip3}(0, \text{picH}_C - 1, yInt_C + i - 1) \end{aligned} \quad (8-788)$$

[0073] 2.4 Example Encoder-only GOP-based temporal filter

[0074] In some embodiments, an encoder-only temporal filter can be implemented. The filtering is done at the encoder side as a pre-processing step. Source pictures before and after the selected picture to encode are read and a block based motion compensation method relative to the selected picture is applied on those source pictures. Samples in the selected picture are temporally filtered using sample values after motion compensation.

[0075] The overall filter strength is set depending on the temporal sub layer of the selected picture as well as the QP. Only pictures at temporal sub layers 0 and 1 are filtered and pictures of layer 0 are filter by a stronger filter than pictures of layer 1. The per sample filter strength is adjusted depending on the difference between the sample value in the selected picture and the co-located samples in motion compensated pictures so that small differences between a motion compensated picture and the selected picture are filtered more strongly than larger differences.

[0076] GOP Based Temporal Filter

[0077] A temporal filter is introduced directly after reading picture and before encoding. Below are the steps described in more detail.

[0078] Operation 1: Pictures are read by the encoder

[0079] Operation 2: If a picture is low enough in the coding hierarchy, it is filtered before encoding. Otherwise the picture is encoded without filtering. RA pictures with POC % 8 == 0 are filtered as well as LD pictures with POC % 4 == 0. AI pictures are never filtered.

[0080] The overall filter strength, s_o , is set according to the equation below for RA.

$$s_o(n) = \begin{cases} 1.5, & n \bmod 16 = 0 \\ 0.95, & n \bmod 16 \neq 0 \end{cases}$$

[0081] where n is the number of pictures read.

[0082] For the LD case, $s_o(n) = 0.95$ is used.

[0083] Operation 3: Two pictures before and/or after the selected picture (referred to as original picture further down) are read. In the edge cases e.g. if is the first picture or close to the last picture, only the available pictures are read.

[0084] Operation 4: Motion of the read pictures before and after, relative to the original picture is estimated per 8x8 picture block.

[0085] A hierarchical motion estimation scheme is used and the layers L0, L1 and L2, are illustrated in FIG. 2. Subsampled pictures are generated by averaging each 2x2 block for all read pictures and the original picture, e.g. L1 in FIG. 1. L2 is derived from L1 using the same subsampling method.

[0086] FIG. 2 shows examples of different layers of the hierarchical motion estimation. L0 is the original resolution. L1 is a subsampled version of L0. L2 is a subsampled version of L1.

[0087] First, motion estimation is done for each 16x16 block in L2. The squared difference is calculated for each selected motion vector and the motion vector corresponding to the smallest difference is selected. The selected motion vector is then used as initial value when estimating the motion in L1. Then the same is done for estimating motion in L0. As a final step, subpixel motion is estimated for each 8x8 block by using an interpolation filter on L0.

[0088] The VTM 6-tap interpolation filter can used:

0:	0,	0,	64,	0,	0,	0
1:	1,	-3,	64,	4,	-2,	0
2:	1,	-6,	62,	9,	-3,	1
3:	2,	-8,	60,	14,	-5,	1
4:	2,	-9,	57,	19,	-7,	2
5:	3,	-10,	53,	24,	-8,	2
6:	3,	-11,	50,	29,	-9,	2
7:	3,	-11,	44,	35,	-10,	3
8:	1,	-7,	38,	38,	-7,	1
9:	3,	-10,	35,	44,	-11,	3
10:	2,	-9,	29,	50,	-11,	3
11:	2,	-8,	24,	53,	-10,	3
12:	2,	-7,	19,	57,	-9,	2
13:	1,	-5,	14,	60,	-8,	2
14:	1,	-3,	9,	62,	-6,	1
15:	0,	-2,	4,	64,	-3,	1

[0089] Operation 5: Motion compensation is applied on the pictures before and after the original picture according to the best matching motion for each block, e.g., so that the sample coordinates of the original picture in each block have the best matching coordinates in the referenced pictures.

[0090] Operation 6: The samples of the processed one by one for the luma and chroma channels as described in the following steps.

[0091] Operation 7: The new sample value, I_n , is calculated using the following formula.

$$I_n = \frac{I_o + \sum_{i=0}^3 w_r(i, a) I_r(i)}{1 + \sum_{i=0}^3 w_r(i, a)}$$

[0092] Where I_o is the sample value of the original sample, $I_r(i)$ is the intensity of the corresponding sample of motion compensated picture i and $w_r(i, a)$ is the weight of motion compensated picture i when the number of available motion compensated pictures is a .

[0093] In the luma channel, the weights, $w_r(i, a)$, is defined as follows:

$$w_r(i, a) = s_l s_o(n) s_r(i, a) e^{-\frac{\Delta I(i)^2}{2\sigma_l(QP)^2}}$$

[0094] Where

$$s_l = 0.4$$

$$s_r(i, 2) = \begin{cases} 1.2, & i = 0 \\ 1.0, & i = 1 \end{cases}$$

$$s_r(i, 4) = \begin{cases} 0.60, & i = 0 \\ 0.85, & i = 1 \\ 0.85, & i = 2 \\ 0.60, & i = 3 \end{cases}$$

[0095] For all other cases of i , and a : $s_r(i, a) = 0.3$

$$\sigma_l(QP) = 3 * (QP - 10)$$

$$\Delta I(i) = I_r(i) - I_o$$

[0096] For the chroma channels, the weights, $w_r(i, a)$, is defined as follows:

$$w_r(i, a) = s_c s_o(n) s_r(i, a) e^{-\frac{\Delta I(i)^2}{2\sigma_c^2}}$$

[0097] Where $s_c = 0.55$ and $\sigma_c = 30$

[0098] Operation 8: The filter is applied for the current sample. The resulting sample value is stored separately.

[0099] Operation 9: The filtered picture is encoded.

[00100] 3. Examples of technical problems solved by disclosed embodiments

[00101] (1) There are some designs that can violate the sub-picture constrain.

[00102] A. TMVP in the affine constructed candidates may fetch a MV in the collocated picture out of the range of the current sub-picture.

[00103] B. When deriving gradients in Bi-Directional Optical Flow (BDOF) and Prediction Refinement Optical Flow (PROF), two extended rows and two extended columns of integer reference samples are required to be fetched. These reference samples may be out of the range of the current sub-picture.

[00104] C. When deriving the chroma residual scaling factor in luma mapping chroma scaling (LMCS), the accessed reconstructed luma samples may be out of the range of the range of the current sub-picture.

[00105] D. The neighboring block may be out of the range of the current sub-picture, when deriving the luma intra prediction mode, reference samples for intra prediction, reference samples for CCLM, neighboring block availability for spatial neighboring candidates for merge/AMVP/CIIP/IBC/LMCS, quantization parameters, CABAC initialization process, ctxInc derivation using left and above syntax elements, and ctxInc for the syntax element mtt_split_cu_vertical_flag. The representation of subpicture may lead to subpicture with incomplete CTUs. The CTU partitions and CU splitting process may need to consider incomplete CTUs.

[00106] (2) The signaled syntax elements related to sub-picture may be arbitrarily large, which may cause an overflow problem.

[00107] (3) The representation of sub-pictures may lead to non-rectangular sub-pictures.

[00108] (4) Currently the subpicture and subpicture grid is defined in units of 4 samples. And the length of syntax element is dependent on the picture height divided by 4. However, since the current pic_width_in_luma_samples and pic_height_in_luma_samples shall be an integer multiple of Max(8, MinCbSizeY), the subpicture grid may need to be defined in units of 8 samples.

[00109] (5) The SPS syntax, `pic_width_max_in_luma_samples` and `pic_height_max_in_luma_samples` may need to be restricted to be no smaller than 8.

[00110] (6) Interaction between reference picture resampling/scalability and subpicture is not considered in the current design.

[00111] (7) In temporal filtering, samples across different sub-pictures may be required.

[00112] 4. Example techniques and embodiments

[00113] The detailed listing below should be considered as examples to explain general concepts. These items should not be interpreted in a narrow way. Furthermore, these items can be combined in any manner. Hereinafter, temporal filter is used to represent filters that require samples in other pictures. $\text{Max}(x, y)$ returns the larger one of x and y . $\text{Min}(x, y)$ returns the smaller one of x and y .

1. The position (named position RB) at which a temporal MV predictor is fetched in a picture to generate affine motion candidates (e.g. a constructed affine merge candidate) must be in a required sub-picture, assuming the top-left corner coordinate of the required sub-picture is (x_{TL}, y_{TL}) and bottom-right coordinate of the required sub-picture is (x_{BR}, y_{BR}) .
 - a. In one example, the required sub-picture is the sub-picture covering the current block.
 - b. In one example, if position RB with a coordinate (x, y) is out of the required sub-picture, the temporal MV predictor is treated as unavailable.
 - i. In one example, position RB is out of the required sub-picture if $x > x_{BR}$.
 - ii. In one example, position RB is out of the required sub-picture if $y > y_{BR}$.
 - iii. In one example, position RB is out of the required sub-picture if $x < x_{TL}$.
 - iv. In one example, position RB is out of the required sub-picture if $y < y_{TL}$.
 - c. In one example, position RB, if outside of the required sub-picture, a replacement of RB is utilized.
 - i. Alternatively, furthermore, the replacement position shall be in the required sub-picture.
 - d. In one example, position RB is clipped to be in the required sub-picture.
 - i. In one example, x is clipped as $x = \text{Min}(x, x_{BR})$.
 - ii. In one example, y is clipped as $y = \text{Min}(y, y_{BR})$.
 - iii. In one example, x is clipped as $x = \text{Max}(x, x_{TL})$.

- iv. In one example, y is clipped as $y = \text{Max}(y, y_{\text{TL}})$.
 - e. In one example, the position RB may be the bottom right position inside the corresponding block of current block in the collocated picture.
 - f. The proposed method may be utilized in other coding tools which require to access motion information from a picture different than the current picture.
 - g. In one example, whether the above methods are applied (e.g., position RB must be in a required sub-picture (e.g. to do as claimed in 1.a and/or 1.b)) may depend on one or more syntax elements signaled in VPS/DPS/SPS/PPS/APS/slice header/tile group header. For example, the syntax element may be `subpic_treated_as_pic_flag[ SubPicIdx ]`, where `SubPicIdx` is the sub-picture index of sub-picture covering the current block.
2. The position (named position S) at which an integer sample is fetched in a reference not used in the interpolation process must be in a required sub-picture, assuming the top-left corner coordinate of the required sub-picture is $(x_{\text{TL}}, y_{\text{TL}})$ and the bottom-right coordinate of the required sub-picture is $(x_{\text{BR}}, y_{\text{BR}})$.
- a. In one example, the required sub-picture is the sub-picture covering the current block.
 - b. In one example, if position S with a coordinate (x, y) is out of the required sub-picture, the reference sample is treated as unavailable.
 - i. In one example, position S is out of the required sub-picture if $x > x_{\text{BR}}$.
 - ii. In one example, position S is out of the required sub-picture if $y > y_{\text{BR}}$.
 - iii. In one example, position S is out of the required sub-picture if $x < x_{\text{TL}}$.
 - iv. In one example, position S is out of the required sub-picture if $y < y_{\text{TL}}$.
 - c. In one example, position S is clipped to be in the required sub-picture.
 - i. In one example, x is clipped as $x = \text{Min}(x, x_{\text{BR}})$.
 - ii. In one example, y is clipped as $y = \text{Min}(y, y_{\text{BR}})$.
 - iii. In one example, x is clipped as $x = \text{Max}(x, x_{\text{TL}})$.
 - iv. In one example, y is clipped as $y = \text{Max}(y, y_{\text{TL}})$.
 - d. In one example, whether position S must be in a required sub-picture (e.g. to do as claimed in 2.a and/or 2.b) may depend on one or more syntax elements signaled in VPS/DPS/SPS/PPS/APS/slice header/tile group header. For example,

the syntax element may be `subpic_treated_as_pic_flag[ SubPicIdx ]`, where `SubPicIdx` is the sub-picture index of sub-picture covering the current block.

- e. In one example, the fetched integer sample is used to generate gradients in BDOF and/or PORF.
3. The position (named position R) at which the reconstructed luma sample value is fetched may be in a required sub-picture, assuming the top-left corner coordinate of the required sub-picture is (xTL, yTL) and the bottom-right coordinate of the required sub-picture is (xBR, yBR).
 - a. In one example, the required sub-picture is the sub-picture covering the current block.
 - b. In one example, if position R with a coordinate (x, y) is out of the required sub-picture, the reference sample is treated as unavailable.
 - i. In one example, position R is out of the required sub-picture if $x > xBR$.
 - ii. In one example, position R is out of the required sub-picture if $y > yBR$.
 - iii. In one example, position R is out of the required sub-picture if $x < xTL$.
 - iv. In one example, position R is out of the required sub-picture if $y < yTL$.
 - c. In one example, position R is clipped to be in the required sub-picture.
 - i. In one example, x is clipped as $x = \text{Min}(x, xBR)$.
 - ii. In one example, y is clipped as $y = \text{Min}(y, yBR)$.
 - iii. In one example, x is clipped as $x = \text{Max}(x, xTL)$.
 - iv. In one example, y is clipped as $y = \text{Max}(y, yTL)$.
 - d. In one example, whether position R must be in a required sub-picture (e.g. to do as claimed in 4.a and/or 4.b) may depend on one or more syntax elements signaled in VPS/DPS/SPS/PPS/APS/slice header/tile group header. For example, the syntax element may be `subpic_treated_as_pic_flag[ SubPicIdx ]`, where `SubPicIdx` is the sub-picture index of sub-picture covering the current block.
 - e. In one example, the fetched luma sample is used to derive the scaling factor for the chroma component(s) in LMCS.
 4. The position (named position N) at which the picture boundary check for BT/TT/QT splitting, BT/TT/QT depth derivation, and/or the signaling of CU split flag must be in a

required sub-picture, assuming the top-left corner coordinate of the required sub-picture is (xTL, yTL) and the bottom-right coordinate of the required sub-picture is (xBR, yBR).

- a. In one example, the required sub-picture is the sub-picture covering the current block.
 - b. In one example, if position N with a coordinate (x, y) is out of the required sub-picture, the reference sample is treated as unavailable.
 - i. In one example, position N is out of the required sub-picture if $x > xBR$.
 - ii. In one example, position N is out of the required sub-picture if $y > yBR$.
 - iii. In one example, position N is out of the required sub-picture if $x < xTL$.
 - iv. In one example, position N is out of the required sub-picture if $y < yTL$.
 - c. In one example, position N is clipped to be in the required sub-picture.
 - i. In one example, x is clipped as $x = \text{Min}(x, xBR)$.
 - ii. In one example, y is clipped as $y = \text{Min}(y, yBR)$.
 - iii. In one example, x is clipped as $x = \text{Max}(x, xTL)$.
 - d. In one example, y is clipped as $y = \text{Max}(y, yTL)$. In one example, whether position N must be in a required sub-picture (e.g. to do as claimed in 5.a and/or 5.b) may depend on one or more syntax elements signaled in VPS/DPS/SPS/PPS/APS/slice header/tile group header. For example, the syntax element may be `subpic_treated_as_pic_flag[ SubPicIdx ]`, where SubPicIdx is the sub-picture index of sub-picture covering the current block.
5. History -based Motion Vector Prediction (HMVP) table may be reset before decoding a new sub-picture in one picture.
- a. In one example, the HMVP table used for IBC coding may be reset
 - b. In one example, the HMVP table used for inter coding may be reset
 - c. In one example, the HMVP table used for intra coding may be reset
6. The subpicture syntax elements may be defined in units of N (such as N = 8, 32, and etc) samples.
- a. In one example, the width of each element of the subpicture identifier grid in units of N samples.
 - b. In one example, the height of each element of the subpicture identifier grid in units of N samples.

- c. In one example, N is set to the width and/or height of CTU.
- 7. The syntax element of picture width and picture height may be restricted to be no smaller than K ($K \geq 8$).
 - a. In one example, the picture width may need to be restricted to be no smaller than 8.
 - b. In one example, the picture height may need to be restricted to be no smaller than 8.
- 8. A conformance bitstream shall satisfy that sub-picture coding and Adaptive resolution conversion (ARC)/ Dynamic resolution conversion (DRC)/ Reference picture resampling (RPR) are disallowed to be enabled for one video unit (e.g., sequence).
 - a. In one example, signaling of enabling sub-picture coding may be under the conditions of disallowing ARC/DRC/RPR.
 - i. In one example, when subpicture is enabled, such as **subpics_present_flag** equal to 1, **pic_width_in_luma_samples** for all pictures for which this SPS is active is equal to **max_width_in_luma_samples**.
 - b. Alternatively, sub-picture coding and ARC/DRC/RPR may be both enabled for one video unit (e.g., sequence).
 - i. In one example, a conformance bitstream shall satisfy that the donwsampled sub-picture due to ARC/DRC/RPR shall still be in the form of K CTUs in width and M CTUs in height wherein K and M are both integers.
 - ii. In one example, a conformance bitstream shall satisfy that for sub-pictures not located at picture boundaries (e.g., right boundary and/or bottom boundary), the donwsampled sub-picture due to ARC/DRC/RPR shall still be in the form of K CTUs in width and M CTUs in height wherein K and M are both integers.
 - iii. In one example, CTU sizes may be adaptively changed based on the picture resolution.

- 1) In one example, a max CTU size may be signaled in SPS. For each picture with less resolution, the CTU size may be changed accordingly based on the reduced resolution.
 - 2) In one example, CTU size may be signaled in SPS and PPS, and/or sub-picture level.
9. The syntax element `subpic_grid_col_width_minus1` and `subpic_grid_row_height_minus1` may be constrained.
- a. In one example, `subpic_grid_col_width_minus1` must be no larger (or must be smaller) than T1.
 - b. In one example, `subpic_grid_row_height_minus1` must be no larger (or must be smaller) than T2.
 - c. In one example, in a conformance bit-stream, `subpic_grid_col_width_minus1` and/or `subpic_grid_row_height_minus1` must follow the constraint such as bullet 3.a or 3.b.
 - d. In one example, T1 in 3.a and/or T2 in 3.b may depend on profiles/levels/tiers of a video coding standard.
 - e. In one example, T1 in 3.a may depend on the picture width.
 - i. For example, T1 is equal to $\text{pic_width_max_in_luma_samples}/4$ or $\text{pic_width_max_in_luma_samples}/4 + \text{Off}$. Off may be 1, 2, -1, -2, etc.
 - f. In one example, T2 in 3.b may depend on the picture width.
 - i. For example, T2 is equal to $\text{pic_height_max_in_luma_samples}/4$ or $\text{pic_height_max_in_luma_samples}/4 - 1 + \text{Off}$. Off may be 1, 2, -1, -2, etc.
10. It is constrained that a boundary between two sub-pictures must be a boundary between two CTUs.
- a. In other words, a CTU cannot be covered by more than one sub-picture.
 - b. In one example, the unit of `subpic_grid_col_width_minus1` may be the CTU width (such as 32, 64, 128), instead of 4 as in VVC. The sub-picture grid width should be $(\text{subpic_grid_col_width_minus1} + 1) * \text{CTU width}$.
 - c. In one example, the unit of `subpic_grid_col_height_minus1` may be the CTU height (such as 32, 64, 128), instead of 4 as in VVC. The sub-picture grid height should be $(\text{subpic_grid_col_height_minus1} + 1) * \text{CTU height}$.

- d. In one example, in a conformance bit-stream, the constraint must be satisfied if the sub-picture approach is applied.
11. It is constrained that the shape of a sub-picture must be rectangular.
- a. In one example, in a conformance bit-stream, the constraint must be satisfied if the sub-picture approach is applied.
 - b. Sub-picture may only contain rectangular slices. For example, in a conformance bit-stream, the constraint must be satisfied if the sub-picture approach is applied.
12. It is constrained that two sub-pictures cannot be overlapped.
- a. In one example, in a conformance bit-stream, the constraint must be satisfied if the sub-picture approach is applied.
 - b. Alternatively, two sub-pictures may be overlapped with each other.
13. It is constrained that any position in the picture must be covered by one and only one sub-picture.
- a. In one example, in a conformance bit-stream, the constraint must be satisfied if the sub-picture approach is applied.
 - b. Alternatively, one sample may not belong to any sub-picture.
 - c. Alternatively, one sample may belong to more than one sub-pictures.
14. It may be constrained that sub-pictures defined in a SPS mapped to every resolution presented in the same sequence should obey the location and/or size constrained mentioned above.
- a. In one example, the width and height of a sub-picture defined in the SPS mapped to a resolution presented in the same sequence, should be integer multiple times of N (such as 8, 16, 32) luma samples.
 - b. In one example, sub-pictures may be defined for certain layer and may be mapped to other layers.
 - i. For example, sub-pictures may be defined for the layer with the highest resolution in the sequence.
 - ii. For example, sub-pictures may be defined for the layer with the lowest resolution in the sequence.

- iii. Which layer the sub-pictures are defined for may be signaled in SPS/VPS/PPS/slice header.
 - c. In one example, when sub-pictures and different resolutions are both applied, all resolutions (e.g., width or/and height) may be integer multiple of a given resolution.
 - d. In one example, the width and/or height of a sub-picture defined in the SPS may be integer multiple times (e.g., M) of the CTU size.
 - e. Alternatively, sub-pictures and different resolutions in a sequence may not be allowed simultaneously.
15. Sub-pictures may only apply to a certain layer(s)
- a. In one example, sub-pictures defined in a SPS may only apply to the layer with the highest resolution in a sequence.
 - b. In one example, sub-pictures defined in a SPS may only apply to the layer with the lowest temporal id in a sequence.
 - c. Which layer(s) that sub-pictures may be applied to may be indicated by one or multiple syntax elements in SPS/VPS/PPS.
 - d. Which layer(s) that sub-picture cannot be applied to may be indicated by one or multiple syntax elements in SPS/VPS/PPS.
16. In one example, the position and/or dimensions of a sub-picture may be signaled without using subpic_grid_idx.
- a. In one example, the top-left position of a sub-picture may be signaled.
 - b. In one example, the bottom-right position of a sub-picture may be signaled.
 - c. In one example, the width of sub-picture may be signaled.
 - d. In one example, the height of a sub-picture may be signaled.
17. For temporal filter, when performing the temporal filtering of a sample, only samples within the same sub-picture that the current sample belongs to may be used. The required samples may be in the same picture that the current sample belongs to or in other pictures.
18. In one example, whether to and/or how to apply a partitioning method (such as QT, horizontal BT, vertical BT, horizontal TT, vertical TT, or not split, etc.) may depend on whether the current block (or partition) crosses one or multiple boundary of a sub-picture.

- a. In one example, the picture boundary handling method for partitioning in VVC may also be applied when a picture boundary is replaced by a sub-picture boundary.
 - b. In one example, whether to parse a syntax element (e.g. a flag) which represents a partitioning method (such as QT, horizontal BT, vertical BT, horizontal TT, vertical TT, or not split, etc.) may depend on whether the current block (or partition) crosses one or multiple boundary of a sub-picture.
19. Instead of splitting one picture into multiple sub-pictures with independent coding of each sub-picture, it is proposed to split a picture into at least two sets of sub-regions, with the first set including multiple sub-pictures and the second set including all the remaining samples.
- a. In one example, a sample in the second set is not in any sub-pictures.
 - b. Alternatively, furthermore, the second set may be encoded/decoded based on the information of the first set.
 - c. In one example, a default value may be utilized to mark whether a sample/MxK sub-region belonging to the second set.
 - i. In one example, the default value may be set equal to $(\text{max_subpics_minus1} + K)$ wherein K is an integer greater than 1.
 - ii. The default value may be assigned to `subpic_grid_idx[i][j]` to indicate that grid belongs to the second set.
20. It is proposed that the syntax element `subpic_grid_idx[i][j]` cannot be larger than `max_subpics_minus1`.
- a. For example, it is constrained that in a conformance bit-stream, `subpic_grid_idx[i][j]` cannot be larger than `max_subpics_minus1`.
 - b. For example, the codeword to code `subpic_grid_idx[i][j]` cannot be larger than `max_subpics_minus1`.
21. It is proposed that, any integer number from 0 to `max_subpics_minus1` must be equal to at least one `subpic_grid_idx[i][j]`.
22. IBC virtual buffer may be reset before decoding a new sub-picture in one picture.
- a. In one example, all the samples in the IBC virtual buffer may be reset to -1.

23. Palette entry list may be reset before decoding a new sub-picture in one picture.

- a. In one example, PredictorPaletteSize may be set equal to 0 before decoding a new sub-picture in one picture.

[00114] 5. Embodiments

[00115] In the following embodiments, the newly added texts are bold italicized and the deleted texts are marked by “[]”.

[00116] 5.1 Embodiment 1: sub-picture constraint on affine constructed merge candidates

8.5.5.6 Derivation process for constructed affine control point motion vector merging candidates

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- two variables cbWidth and cbHeight specifying the width and the height of the current luma coding block,
- the availability flags availableA₀, availableA₁, availableA₂, availableB₀, availableB₁, availableB₂, availableB₃,
- the sample locations (xNbA₀, yNbA₀), (xNbA₁, yNbA₁), (xNbA₂, yNbA₂), (xNbB₀, yNbB₀), (xNbB₁, yNbB₁), (xNbB₂, yNbB₂) and (xNbB₃, yNbB₃).

Output of this process are:

- the availability flag of the constructed affine control point motion vector merging candidates availableFlagConstK, with K = 1..6,
 - the reference indices refIdxLXConstK, with K = 1..6, X being 0 or 1,
 - the prediction list utilization flags predFlagLXConstK, with K = 1..6, X being 0 or 1,
 - the affine motion model indices motionModelIdxConstK, with K = 1..6,
 - the bi-prediction weight indices bcwIdxConstK, with K = 1..6,
 - the constructed affine control point motion vectors cpMvLXConstK[cpIdx] with cpIdx = 0..2, K = 1..6 and X being 0 or 1.
- ...

The fourth (collocated bottom-right) control point motion vector cpMvLXCorner[3], reference index refIdxLXCorner[3], prediction list utilization flag predFlagLXCorner[3] and the availability flag availableFlagCorner[3] with X being 0 and 1 are derived as follows:

- The reference indices for the temporal merging candidate, refIdxLXCorner[3], with X being 0 or 1, are set equal to 0.
- The variables mvLXCol and availableFlagLXCol, with X being 0 or 1, are derived as follows:
 - If slice_temporal_mvp_enabled_flag is equal to 0, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
 - Otherwise (slice_temporal_mvp_enabled_flag is equal to 1), the following applies:

$$xColBr = xCb + cbWidth \quad (8-601)$$

$$yColBr = yCb + cbHeight \quad (8-602)$$

$rightBoundaryPos = subpic \text{ treated as pic flag}[SubPicIdx] ?$

$SubPicRightBoundaryPos : pic \text{ width in luma samples} - 1$

$botBoundaryPos = subpic \text{ treated as pic flag}[SubPicIdx] ?$

$SubPicBotBoundaryPos : pic \text{ height in luma samples} - 1$

- If $yCb \gg CtbLog2SizeY$ is equal to $yColBr \gg CtbLog2SizeY$, $yColBr$ is less than or equal to $botBoundaryPos$ and $xColBr$ is less than or equal to $rightBoundaryPos$, the following applies:
 - The variable $colCb$ specifies the luma coding block covering the modified location given by $((xColBr \gg 3) \ll 3, (yColBr \gg 3) \ll 3)$ inside the collocated picture specified by $ColPic$.
 - The luma location $(xColCb, yColCb)$ is set equal to the top-left sample of the collocated luma coding block specified by $colCb$ relative to the top-left luma sample of the collocated picture specified by $ColPic$.
 - The derivation process for collocated motion vectors as specified in clause 8.5.2.12 is invoked with $currCb, colCb, (xColCb, yColCb)$, $refIdxLXCorner[3]$ and $sbFlag$ set equal to 0 as inputs, and the output is assigned to $mvLXCol$ and $availableFlagLXCol$.
- Otherwise, both components of $mvLXCol$ are set equal to 0 and $availableFlagLXCol$ is set equal to 0.

...

[00117] 5.2 Embodiment 2: sub-picture constraint on affine constructed merge candidates

8.5.5.6 Derivation process for constructed affine control point motion vector merging candidates

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- two variables $cbWidth$ and $cbHeight$ specifying the width and the height of the current luma coding block,
- the availability flags $availableA_0, availableA_1, availableA_2, availableB_0, availableB_1, availableB_2, availableB_3$,
- the sample locations $(xNbA_0, yNbA_0), (xNbA_1, yNbA_1), (xNbA_2, yNbA_2), (xNbB_0, yNbB_0), (xNbB_1, yNbB_1), (xNbB_2, yNbB_2)$ and $(xNbB_3, yNbB_3)$.

Output of this process are:

- the availability flag of the constructed affine control point motion vector merging candidates $availableFlagConstK$, with $K = 1..6$,
- the reference indices $refIdxLXConstK$, with $K = 1..6$, X being 0 or 1,
- the prediction list utilization flags $predFlagLXConstK$, with $K = 1..6$, X being 0 or 1,
- the affine motion model indices $motionModelIdxConstK$, with $K = 1..6$,
- the bi-prediction weight indices $bcwIdxConstK$, with $K = 1..6$,
- the constructed affine control point motion vectors $cpMvLXConstK[cpIdx]$ with $cpIdx = 0..2$, $K = 1..6$ and X being 0 or 1.

...

The fourth (collocated bottom-right) control point motion vector $cpMvLXCorner[3]$, reference index $refIdxLXCorner[3]$, prediction list utilization flag $predFlagLXCorner[3]$ and the availability flag $availableFlagLXCorner[3]$ with X being 0 and 1 are derived as follows:

- The reference indices for the temporal merging candidate, $refIdxLXCorner[3]$, with X being 0 or 1, are set equal to 0.
- The variables $mvLXCol$ and $availableFlagLXCol$, with X being 0 or 1, are derived as follows:
 - If $slice_temporal_mvp_enabled_flag$ is equal to 0, both components of $mvLXCol$ are set equal to 0 and $availableFlagLXCol$ is set equal to 0.
 - Otherwise ($slice_temporal_mvp_enabled_flag$ is equal to 1), the following applies:

$$xColBr = xCb + cbWidth \quad (8-601)$$

$$yColBr = yCb + cbHeight \quad (8-602)$$

$rightBoundaryPos = subpic_treated_as_pic_flag[SubPicIdx] ?$

$SubPicRightBoundaryPos : pic_width_in_luma_samples - 1$

$botBoundaryPos = subpic_treated_as_pic_flag[SubPicIdx] ?$

$SubPicBotBoundaryPos : pic_height_in_luma_samples - 1$

$xColBr = Min(rightBoundaryPos, xColBr)$

$yColBr = Min(botBoundaryPos, yColBr)$

- If $yCb \gg CtbLog2SizeY$ is equal to $yColBr \gg CtbLog2SizeY$, [$yColBr$ is less than $pic_height_in_luma_samples$ and $xColBr$ is less than $pic_width_in_luma_samples$, the following applies]]:
 - The variable $colCb$ specifies the luma coding block covering the modified location given by $((xColBr \gg 3) \ll 3, (yColBr \gg 3) \ll 3)$ inside the collocated picture specified by $ColPic$.
 - The luma location $(xColCb, yColCb)$ is set equal to the top-left sample of the collocated luma coding block specified by $colCb$ relative to the top-left luma sample of the collocated picture specified by $ColPic$.
 - The derivation process for collocated motion vectors as specified in clause 8.5.2.12 is invoked with $currCb, colCb, (xColCb, yColCb), refIdxLXCorner[3]$ and $sbFlag$ set equal to 0 as inputs, and the output is assigned to $mvLXCol$ and $availableFlagLXCol$.
- Otherwise, both components of $mvLXCol$ are set equal to 0 and $availableFlagLXCol$ is set equal to 0.

...

[00118] 5.3 Embodiment 3: Fetching integer samples under the sub-picture constraint

8.5.6.3.3 Luma integer sample fetching process

Inputs to this process are:

- a luma location in full-sample units $(xInt_L, yInt_L)$,
- the luma reference sample array $refPicLX_L$.

Output of this process is a predicted luma sample value $predSampleLX_L$.

The variable shift is set equal to $\text{Max}(2, 14 - \text{BitDepth}_Y)$.

The variable picW is set equal to pic_width_in_luma_samples and the variable picH is set equal to pic_height_in_luma_samples.

The luma locations in full-sample units (xInt, yInt) are derived as follows:

– If subpic treated as pic flag[SubPicIdx] is equal to 1, the following applies:

$xInt = \text{Clip3}(\text{SubPicLeftBoundaryPos}, \text{SubPicRightBoundaryPos}, xInt)$

$yInt = \text{Clip3}(\text{SubPicTopBoundaryPos}, \text{SubPicBotBoundaryPos}, yInt)$

– Otherwise:

$xInt = \text{Clip3}(0, picW - 1, \text{sps_ref_wraparound_enabled_flag} ?$ (8-782)

$\text{ClipH}((\text{sps_ref_wraparound_offset_minus1} + 1) * \text{MinCbSizeY}, picW, xInt_L) : xInt_L)$

$yInt = \text{Clip3}(0, picH - 1, yInt_L)$ (8-783)

The predicted luma sample value predSampleLX_L is derived as follows:

$\text{predSampleLX}_L = \text{refPicLX}_L[xInt][yInt] \ll \text{shift3}$ (8-784)

[00119] 5.4 Embodiment 4: deriving the variable invAvgLuma in chroma residual scaling of LMCS

8.7.5.3 Picture reconstruction with luma dependent chroma residual scaling process for chroma samples

Inputs to this process are:

- a chroma location (xCurr, yCurr) of the top-left chroma sample of the current chroma transform block relative to the top-left chroma sample of the current picture,
- a variable nCurrSw specifying the chroma transform block width,
- a variable nCurrSh specifying the chroma transform block height,
- a variable tuCbfChroma specifying the coded block flag of the current chroma transform block,
- an (nCurrSw)x(nCurrSh) array predSamples specifying the chroma prediction samples of the current block,
- an (nCurrSw)x(nCurrSh) array resSamples specifying the chroma residual samples of the current block,

Output of this process is a reconstructed chroma picture sample array recSamples.

The variable sizeY is set equal to $\text{Min}(\text{CtbSizeY}, 64)$.

The reconstructed chroma picture sample recSamples is derived as follows for $i = 0..nCurrSw - 1$,

$j = 0..nCurrSh - 1$:

- ...
- Otherwise, the following applies:
 - ...
 - The variable currPic specifies the array of reconstructed luma samples in the current picture.
 - For the derivation of the variable varScale the following ordered steps apply:
 1. The variable invAvgLuma is derived as follows:
 - The array recLuma[i] with $i=0..(2 * \text{sizeY} - 1)$ and the variable cnt are derived as follows:
 - The variable cnt is set equal to 0.
 - *The variable rightBoundaryPos and botBoundaryPos are derived as follows:*

$rightBoundaryPos = subpic_treated_as_pic_flag[SubPicIdx] ?$

$SubPicRightBoundaryPos : pic_width_in_luma_samples - 1$

$botBoundaryPos = subpic_treated_as_pic_flag[SubPicIdx] ?$

$SubPicBotBoundaryPos : pic_height_in_luma_samples - 1$

- When availL is equal to TRUE, the array recLuma[i] with $i = 0..sizeY - 1$ is set equal to $currPic[xCuCb - 1][Min(yCuCb + i, [[pic_height_in_luma_samples - 1]]$
 $botBoundaryPos)]$ with $i = 0..sizeY - 1$, and cnt is set equal to sizeY
- When availT is equal to TRUE, the array recLuma[cnt + i] with $i = 0..sizeY - 1$ is set equal to
 $currPic[Min(xCuCb + i, [[pic_width_in_luma_samples - 1]] rightBoundaryPos)][yCuCb - 1]$ with $i = 0..sizeY - 1$, and cnt is set equal to (cnt + sizeY)
- The variable invAvgLuma is derived as follows:
 - If cnt is greater than 0, the following applies:

$$invAvgLuma = Clip1_Y((\sum_{k=0}^{cnt-1} recLuma[k] + (cnt \gg 1)) \gg \text{Log2}(cnt)) \quad (8-1013)$$
 - Otherwise (cnt is equal to 0), the following applies:

$$invAvgLuma = 1 \ll (BitDepth_Y - 1) \quad (8-1014)$$

[00120] 5.5 Embodiment 5: an example of defining the subpicture element in unit of N (such as N=8 or 32) other than 4 samples

7.4.3.3 Sequence parameter set RBSP semantics

subpic_grid_col_width_minus1 plus 1 specifies the width of each element of the subpicture identifier grid in units of $\frac{4N}{4}$ samples. The length of the syntax element is $\text{Ceil}(\text{Log2}(pic_width_max_in_luma_samples / \frac{4N}{4}))$ bits.

The variable NumSubPicGridCols is derived as follows:

$$\begin{aligned} \text{NumSubPicGridCols} = & (pic_width_max_in_luma_samples + subpic_grid_col_width_minus1 * [[4+ 3]] \frac{N}{4} + N - 1) / \\ & (subpic_grid_col_width_minus1 * [[4+ 3]] \frac{N + N - 1}{4}) \end{aligned} \quad (7-5)$$

subpic_grid_row_height_minus1 plus 1 specifies the height of each element of the subpicture identifier grid in units of 4 samples. The length of the syntax element is $\text{Ceil}(\text{Log2}(pic_height_max_in_luma_samples / \frac{4N}{4}))$ bits.

The variable NumSubPicGridRows is derived as follows:

$$\begin{aligned} \text{NumSubPicGridRows} = & (pic_height_max_in_luma_samples + subpic_grid_row_height_minus1 * \frac{4N + N - 1}{4} / \\ & (subpic_grid_row_height_minus1 * [[4+ 3]] \frac{N + N - 1}{4}) \end{aligned}$$

7.4.7.1 General slice header semantics

The variables SubPicIdx, SubPicLeftBoundaryPos, SubPicTopBoundaryPos, SubPicRightBoundaryPos, and SubPicBotBoundaryPos are derived as follows:

$$\begin{aligned} \text{SubPicIdx} &= \text{CtbToSubPicIdx}[\text{CtbAddrBsToRs}[\text{FirstCtbAddrBs}[\text{SliceBrickIdx}[0]]]] \\ \text{if}(& \quad \quad \quad \text{subpic_treated_as_pic_flag}[\text{SubPicIdx}] \quad \quad \quad) \quad \quad \quad \{ \\ \text{SubPicLeftBoundaryPos} &= \text{SubPicLeft}[\text{SubPicIdx}] * (subpic_grid_col_width_minus1 + 1) * \frac{4N}{4} \end{aligned}$$

$$\begin{aligned}
\text{SubPicRightBoundaryPos} &= (\text{SubPicLeft}[\text{SubPicIdx}] + \text{SubPicWidth}[\text{SubPicIdx}]) * \\
&\quad (\text{subpic_grid_col_width_minus1} + 1) * \underline{4N} \quad (7-93) \\
\text{SubPicTopBoundaryPos} &= \text{SubPicTop}[\text{SubPicIdx}] * (\text{subpic_grid_row_height_minus1} + 1) * \underline{4N} \\
\text{SubPicBotBoundaryPos} &= (\text{SubPicTop}[\text{SubPicIdx}] + \text{SubPicHeight}[\text{SubPicIdx}]) * \\
&\quad (\text{subpic_grid_row_height_minus1} + 1) * \underline{4N} \\
&\}
\end{aligned}$$

[00121] 5.6 Embodiment 6: restrict the picture width and the picture height to be equal or larger than 8

7.4.3.3 Sequence parameter set RBSP semantics

pic_width_max_in_luma_samples specifies the maximum width, in units of luma samples, of each decoded picture referring to the SPS. **pic_width_max_in_luma_samples** shall not be equal to 0 and shall be an integer multiple of $[[\text{MinCbSizeY}]] \underline{\text{Max}(8, \text{MinCbSizeY})}$.

pic_height_max_in_luma_samples specifies the maximum height, in units of luma samples, of each decoded picture referring to the SPS. **pic_height_max_in_luma_samples** shall not be equal to 0 and shall be an integer multiple of $[[\text{MinCbSizeY}]] \underline{\text{Max}(8, \text{MinCbSizeY})}$.

[00122] 5.7 Embodiment 7: subpicture boundary check for BT/TT/QT splitting, BT/TT/QT depth derivation, and/or the signaling of CU split flag

6.4.2 Allowed binary split process

The variable **allowBtSplit** is derived as follows:

- ...
- Otherwise, if all of the following conditions are true, **allowBtSplit** is set equal to FALSE
 - **btSplit** is equal to **SPLIT_BT_VER**
 - **y0 + cbHeight** is greater than $[[\text{pic_height_in_luma_samples}]] \underline{\text{subpic treated as pic flag}[\text{SubPicIdx}] ? \text{SubPicBotBoundaryPos} + 1 : \text{pic height in luma samples} }$.
- Otherwise, if all of the following conditions are true, **allowBtSplit** is set equal to FALSE
 - **btSplit** is equal to **SPLIT_BT_VER**
 - **cbHeight** is greater than **MaxTbSizeY**
 - **x0 + cbWidth** is greater than $[[\text{pic_width_in_luma_samples}]] \underline{\text{subpic treated as pic flag}[\text{SubPicIdx}] ? \text{SubPicRightBoundaryPos} + 1 : \text{pic width in luma samples} }$
- Otherwise, if all of the following conditions are true, **allowBtSplit** is set equal to FALSE
 - **btSplit** is equal to **SPLIT_BT_HOR**
 - **cbWidth** is greater than **MaxTbSizeY**
 - **y0 + cbHeight** is greater than $[[\text{pic_height_in_luma_samples}]] \underline{\text{subpic treated as pic flag}[\text{SubPicIdx}] ? \text{SubPicBotBoundaryPos} + 1 : \text{pic height in luma samples} }$.
- Otherwise, if all of the following conditions are true, **allowBtSplit** is set equal to FALSE

- $x0 + cbWidth$ is greater than $[[pic_width_in_luma_samples]]$ *subpic treated as pic flag[SubPicIdx] ? SubPicRightBoundaryPos + 1 : pic width in luma samples*
- $y0 + cbHeight$ is greater than $[[pic_height_in_luma_samples]]$ *subpic treated as pic flag[SubPicIdx] ? SubPicBotBoundaryPos + 1 : pic height in luma samples*.
- $cbWidth$ is greater than $minQtSize$
- Otherwise, if all of the following conditions are true, $allowBtSplit$ is set equal to FALSE
 - $btSplit$ is equal to SPLIT_BT_HOR
 - $x0 + cbWidth$ is greater than $[[pic_width_in_luma_samples]]$ *subpic treated as pic flag[SubPicIdx] ? SubPicRightBoundaryPos + 1 : pic width in luma samples*
 - $y0 + cbHeight$ is less than or equal to $[[pic_height_in_luma_samples]]$ *subpic treated as pic flag[SubPicIdx] ? SubPicBotBoundaryPos + 1 : pic height in luma samples*.

6.4.2 Allowed ternary split process

The variable $allowTtSplit$ is derived as follows:

- If one or more of the following conditions are true, $allowTtSplit$ is set equal to FALSE:
 - $cbSize$ is less than or equal to $2 * MinTtSizeY$
 - $cbWidth$ is greater than $Min(MaxTbSizeY, maxTtSize)$
 - $cbHeight$ is greater than $Min(MaxTbSizeY, maxTtSize)$
 - $mttDepth$ is greater than or equal to $maxMttDepth$
 - $x0 + cbWidth$ is greater than $[[pic_width_in_luma_samples]]$ *subpic treated as pic flag[SubPicIdx] ? SubPicRightBoundaryPos + 1 : pic width in luma samples*
 - $y0 + cbHeight$ is greater than $[[pic_height_in_luma_samples]]$ *subpic treated as pic flag[SubPicIdx] ? SubPicBotBoundaryPos + 1 : pic height in luma samples*.
 - $treeType$ is equal to DUAL_TREE_CHROMA and $(cbWidth / SubWidthC) * (cbHeight / SubHeightC)$ is less than or equal to 32
 - $treeType$ is equal to DUAL_TREE_CHROMA and $modeType$ is equal to MODE_TYPE_INTRA
- Otherwise, $allowTtSplit$ is set equal to TRUE.

7.3.8.2 Coding tree unit syntax

	Descriptor
dual_tree_implicit_qt_split(x0, y0, cbSize, cqtDepth) {	
...	
if(x1 < [[pic_width_in_luma_samples]] <u>(subpic treated as pic flag[SubPicIdx] ? SubPicRightBoundaryPos + 1 : pic width in luma samples)</u>)	
dual_tree_implicit_qt_split(x1, y0, cbSize / 2, cqtDepth + 1)	
if(y1 < [[pic_height_in_luma_samples]] <u>(subpic treated as pic flag[SubPicIdx] ? SubPicBotBoundaryPos + 1 : pic height in luma samples)</u>)	
dual_tree_implicit_qt_split(x0, y1, cbSize / 2, cqtDepth + 1)	
if(x1 < [[pic_width_in_luma_samples]] <u>(subpic treated as pic flag[SubPicIdx] ? SubPicRightBoundaryPos + 1 : pic width in luma samples)</u> && y1 < [[pic_height_in_luma_samples]] <u>(subpic treated as pic flag[SubPicIdx] ? SubPicBotBoundaryPos + 1 : pic height in luma samples)</u>)	
dual_tree_implicit_qt_split(x1, y1, cbSize / 2, cqtDepth + 1)	
} else {	
...	
}	
}	

7.3.8.4 Coding tree syntax

coding_tree(x0, y0, cbWidth, cbHeight, qgOnY, qgOnC, cbSubdiv, cqtDepth, mttDepth, depthOffset, partIdx, treeTypeCurr, modeTypeCurr) {	Descriptor
if((allowSplitBtVer allowSplitBtHor allowSplitTtVer allowSplitTtHor allowSplitQT) &&(x0 + cbWidth <= [[pic_width_in_luma_samples]] <u>(subpic treated as pic flag[SubPicIdx] ? SubPicRightBoundaryPos + 1 : pic_width_in_luma_samples)</u>) &&(y0 + cbHeight <= [[pic_height_in_luma_samples]] <u>(subpic treated as pic flag[SubPicIdx] ? SubPicBotBoundaryPos + 1 : pic_height_in_luma_samples)</u>))	
split_cu_flag	ae(v)
if(cu_qp_delta_enabled_flag && qgOnY && cbSubdiv <= cu_qp_delta_subdiv) {	
...	
depthOffset += (y0 + cbHeight > [[pic_height_in_luma_samples]] <u>(subpic treated as pic flag[SubPicIdx] ? SubPicBotBoundaryPos + 1 : pic_height_in_luma_samples)</u>) ? 1 : 0	
y1 = y0 + (cbHeight / 2)	
coding_tree(x0, y0, cbWidth, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, modeType)	
if(y1 < [[pic_height_in_luma_samples]] <u>(subpic treated as pic flag[SubPicIdx] ? SubPicBotBoundaryPos + 1 : pic_height_in_luma_samples)</u>)	
coding_tree(x0, y1, cbWidth, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 1, treeType, modeType)	
...	
if(x1 < [[pic_width_in_luma_samples]] <u>(subpic treated as pic flag[SubPicIdx] ? SubPicRightBoundaryPos + 1 : pic_width_in_luma_samples)</u>)	
coding_tree(x1, y0, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 1, treeType, modeType)	
if(y1 < [[pic_height_in_luma_samples]] <u>(subpic treated as pic flag[SubPicIdx] ? SubPicBotBoundaryPos + 1 : pic_height_in_luma_samples)</u>)	
coding_tree(x0, y1, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 2, treeType, modeType)	
if(y1 < [[pic_height_in_luma_samples]] <u>(subpic treated as pic flag[SubPicIdx] ? SubPicBotBoundaryPos + 1 : pic_height_in_luma_samples)</u>) && x1 < [[pic_width_in_luma_samples]] <u>(subpic treated as pic flag[SubPicIdx] ? SubPicRightBoundaryPos + 1 : pic_width_in_luma_samples)</u>)	
coding_tree(x1, y1, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 3, treeType, modeType)	

[00123] FIG. 3 is a block diagram of a video processing apparatus 300. The apparatus 300 may be used to implement one or more of the methods described herein. The apparatus 300 may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 300 may include one or more processors 302, one or more memories 304 and video processing hardware 306. The processor(s) 302 may be configured to implement one or more methods described in the present document. The memory (memories) 304 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 306 may be used to implement, in hardware circuitry, some techniques described in the present document.

[00124] FIG. 4 is a flowchart for a method 400 of processing a video. The method 1800 includes determining (402), for a video block in a first video region of a video, whether a position at which a temporal motion vector predictor is determined for a conversion between the video block and a bitstream representation of the current video block using an affine mode is within a second video region, and performing (404) the conversion based on the determining.

[00125] The following solutions may be implemented as preferred solutions in some embodiments.

[00126] The following solutions may be implemented together with additional techniques described in items listed in the previous section (e.g., item 1).

[00127] 1. A method of video processing, comprising: determining, for a video block in a first video region of a video, whether a position at which a temporal motion vector predictor is determined for a conversion between the video block and a bitstream representation of the current video block using an affine mode is within a second video region; and performing the conversion based on the determining.

[00128] 2. The method of solution 1, wherein the video block is covered by the first region and the second region.

[00129] 3. The method of any of solutions 1-2, wherein, in case that the position of the temporal motion vector predictor is outside of the second video region, then the temporal motion vector predictor is marked as unavailable and is unused in the conversion.

[00130] The following solutions may be implemented together with additional techniques described in items listed in the previous section (e.g., item 2).

[00131] 4. A method of video processing, comprising: determining, for a video block in a first video region of a video, whether a position at which an integer sample in a reference picture is fetched for a conversion between the video block and a bitstream representation of the current video block is within a second video region, wherein the reference picture is not used in an interpolation process during the conversion; and performing the conversion based on the determining.

[00132] 5. The method of solution 4, wherein the video block is covered by the first region and the second region.

[00133] 6. The method of any of solutions 4-5, wherein, in case that the position of the sample is outside of the second video region, then the sample is marked as unavailable and is unused in the conversion.

[00134] The following solutions may be implemented together with additional techniques described in items listed in the previous section (e.g., item 3).

[00135] 7. A method of video processing, comprising: determining, for a video block in a first video region of a video, whether a position at which a reconstructed luma sample value is fetched for a conversion between the video block and a bitstream representation of the current video block is within a second video region; and performing the conversion based on the determining.

[00136] 8. The method of solution 7, wherein the luma sample is covered by the first region and the second region.

[00137] 9. The method of any of solutions 7-8, wherein, in case that the position of the luma sample is outside of the second video region, then the luma sample is marked as unavailable and is unused in the conversion.

[00138] The following solutions may be implemented together with additional techniques described in items listed in the previous section (e.g., item 4).

[00139] 10. A method of video processing, comprising: determining, for a video block in a first video region of a video, whether a position at which a check regarding splitting, depth derivation or split flag signaling for the video block is performed during a conversion between the video block and a bitstream representation of the current video block is within a second video region; and performing the conversion based on the determining.

[00140] 11. The method of solution 10, wherein the position is covered by the first region and the second region.

[00141] 12. The method of any of solutions 10-11, wherein, in case that the position is outside of the second video region, then the luma sample is marked as unavailable and is unused in the conversion.

[00142] The following solutions may be implemented together with additional techniques described in items listed in the previous section (e.g., item 8).

[00143] 13. A method of video processing, comprising: performing a conversion between a video comprising one or more video pictures comprising one or more video blocks, and a coded representation of the video, wherein the coded representation complies with a coding syntax requirement that the conversion is not to use sub-picture coding/decoding and a dynamic resolution conversion coding/decoding tool or a reference picture resampling tool within a video unit.

[00144] 14. The method of solution 13, wherein the video unit corresponds to a sequence of the one or more video pictures.

[00145] 15. The method of any of solutions 13-14, wherein the dynamic resolution conversion coding/decoding tool comprises an adaptive resolution conversion coding/decoding tool.

[00146] 16. The method of any of solutions 13-14, wherein the dynamic resolution conversion coding/decoding tool comprises a dynamic resolution conversion coding/decoding tool.

[00147] 17. The method of any of solutions 13-16, wherein the coded representation indicates that the video unit complies with the coding syntax requirement.

[00148] 18. The method of solution 17, wherein the coded representation indicates that the video unit uses sub-picture coding.

[00149] 19. The method of solution 17, wherein the coded representation indicates that the video unit uses the dynamic resolution conversion coding/decoding tool or the reference picture resampling tool.

[00150] The following solutions may be implemented together with additional techniques described in items listed in the previous section (e.g., item 10).

[00151] 20. The method of any of solutions 1-19, wherein the second video region comprises a video sub-picture and wherein boundaries of the second video region and another video region is also a boundary between two coding tree units.

[00152] 21. The method of any of solutions 1-19, wherein the second video region comprises a video sub-picture and wherein boundaries of the second video region and another video region is also a boundary between two coding tree units.

[00153] The following solutions may be implemented together with additional techniques described in items listed in the previous section (e.g., item 11).

[00154] 22. The method of any of solutions 1-21, wherein the first video region and the second video region have rectangular shapes.

[00155] The following solutions may be implemented together with additional techniques described in items listed in the previous section (e.g., item 12).

[00156] 23. The method of any of solutions 1-22, wherein the first video region and the second video region are non-overlapping.

[00157] The following solutions may be implemented together with additional techniques described in items listed in the previous section (e.g., item 13).

[00158] 24. The method of any of solutions 1-23, wherein the video picture is divided into video regions such that a pixel in the video picture is covered by one and only one video region.

[00159] The following solutions may be implemented together with additional techniques described in items listed in the previous section (e.g., item 15).

[00160] 25. The method of any of solutions 1-24, wherein the video picture is split into the first video region and the second video region due to the video picture being in a specific layer of the video sequence.

[00161] The following solutions may be implemented together with additional techniques described in items listed in the previous section (e.g., item 10).

[00162] 26. A method of video processing, comprising: performing a conversion between a video comprising one or more video pictures comprising one or more video blocks, and a coded representation of the video, wherein the coded representation complies with a coding syntax requirement that a first syntax element `subpic_grid_idx[i][j]` is not larger than a second syntax element `max_subpics_minus1`.

[00163] 27. The method of solution 26, wherein a codeword representing the first syntax element is not larger than a codeword representing the second syntax element.

[00164] 28. The method of any of solutions 1-27, wherein the first video region comprises a video sub-picture.

[00165] 29. The method of any of solutions 1-28, wherein the second video region comprises a video sub-picture.

[00166] 30. The method of any of solutions 1 to 29, wherein the conversion comprises encoding the video into the coded representation.

[00167] 31. The method of any of solutions 1 to 29, wherein the conversion comprises decoding the coded representation to generate pixel values of the video.

[00168] 32. A video decoding apparatus comprising a processor configured to implement a method recited in one or more of solutions 1 to 31.

[00169] 33. A video encoding apparatus comprising a processor configured to implement a method recited in one or more of solutions 1 to 31.

[00170] 34. A computer program product having computer code stored thereon, the code, when executed by a processor, causes the processor to implement a method recited in any of solutions 1 to 31.

[00171] 35. A method, apparatus or system described in the present document.

[00172] FIG. 5 is a block diagram showing an example video processing system 500 in which various techniques disclosed herein may be implemented. Various implementations may include some or all of the components of the system 500. The system 500 may include input 502 for receiving video content. The video content may be received in a raw or uncompressed format, e.g., 8 or 10 bit multi-component pixel values, or may be in a compressed or encoded format. The input 502 may represent a network interface, a peripheral bus interface, or a storage interface. Examples of network interface include wired interfaces such as Ethernet, passive optical network (PON), etc. and wireless interfaces such as Wi-Fi or cellular interfaces.

[00173] The system 500 may include a coding component 504 that may implement the various coding or encoding methods described in the present document. The coding component 504 may reduce the average bitrate of video from the input 502 to the output of the coding component 504 to produce a coded representation of the video. The coding techniques are therefore sometimes called video compression or video transcoding techniques. The output of the coding component 504 may be either stored, or transmitted via a communication connected, as represented by the component 506. The stored or communicated bitstream (or coded) representation of the video received at the input 502 may be used by the component 508 for generating pixel values or displayable video that is sent to a display interface 510. The process of generating user-viewable video from the bitstream representation is sometimes called video decompression. Furthermore, while certain video processing operations are referred to as “coding” operations or tools, it will

be appreciated that the coding tools or operations are used at an encoder and corresponding decoding tools or operations that reverse the results of the coding will be performed by a decoder.

[00174] Examples of a peripheral bus interface or a display interface may include universal serial bus (USB) or high definition multimedia interface (HDMI) or Displayport, and so on. Examples of storage interfaces include SATA (serial advanced technology attachment), PCI, IDE interface, and the like. The techniques described in the present document may be embodied in various electronic devices such as mobile phones, laptops, smartphones or other devices that are capable of performing digital data processing and/or video display.

[00175] FIG. 6 is a flowchart representation of a method 600 for video processing in accordance with the present technology. The method 600 includes, at operation 610, determining, for a conversion between a current block of a first picture of a video and a bitstream representation of the video, a motion candidate based on motion information from a second picture according to a rule. The rule specifies that a position from which the motion information is accessed is constrained to be within a specific subpicture of the second picture. The method 600 also includes at operation 620, performing the conversion based on the determining.

[00176] In some embodiments, the motion information comprises a motion vector and the motion candidate comprises an affine motion candidate. In some embodiments, the rule specifies that the position is used as a replacement for a first position in case the first position is outside of the specific subpicture. In some embodiments, the position is at a bottom-right corner of a block in the video picture that corresponds to the current block.

[00177] FIG. 7 is a flowchart representation of a method 700 for video processing in accordance with the present technology. The method 700 includes, at operation 710, determining, for a conversion of a current block of a first picture of a video and a bitstream representation of the video, an integer sample from a second picture according to a rule. The second picture comprises a reference picture that is not used in an interpolation process. The rule specifies that a position from which the integer sample is accessed is constrained to be within a specific subpicture of the second picture. The method 700 also includes, at operation 720, performing the conversion based on the determining. In some embodiments, the integer sample is used to generate one or more gradients in a Bi-Directional Optical Flow or a Prediction Refinement Optical Flow process.

[00178] FIG. 8 is a flowchart representation of a method 800 for video processing in accordance

with the present technology. The method 800 includes, at operation 810, determining, for a conversion of a current block of a video and a bitstream representation of the video, a position at which a reconstructed luma sample is accessed according to a rule. The rule specifies that the position is constrained to be within a specific subpicture of a video picture. The method 800 also includes, at operation 820, performing the conversion based on the determining. In some embodiments, the reconstructed luma sample is accessed to derive a scaling factor for a chroma component in a luma mapping chroma scaling process.

[00179] FIG. 9 is a flowchart representation of a method 900 for video processing in accordance with the present technology. The method 900 includes, at operation 910, determining, for a conversion of a current block of a video and a bitstream representation of the video, a position at which a picture boundary check is performed according to a rule. The rule specifies that the position is constrained to be within a specific subpicture of a video picture. The method 900 also includes, at operation 920, performing the conversion based on the determining.

[00180] In some embodiments, the picture boundary check is performed for at least one of: (1) a splitting of a binary tree, a tertiary tree, or a quad tree, (2) a depth derivation for a binary tree, a tertiary tree, or a quad tree, (3) or a signaling of a split flag for the current block. In some embodiments, the specific subpicture is a collocated subpicture that covers the current block.

[00181] In some embodiments, the rule specifies that information at the position treated as unavailable in case the position is outside of a specific subpicture. The position is represented as (x, y) , a top-left corner of the specific subpicture is represented as (x_{TL}, y_{TL}) , and a bottom-right corner of the specific subpicture is represented as (x_{BR}, y_{BR}) . In some embodiments, the position is outside of the specific subpicture in case $x > x_{BR}$, $y > y_{BR}$, $x < x_{TL}$, or $y < y_{TL}$. In some embodiments, the rule specifies that the position is clipped to be within a specific subpicture of the video picture. The position is represented as (x, y) , a top-left corner of the specific subpicture is represented as (x_{TL}, y_{TL}) , and a bottom-right corner of the specific subpicture is represented as (x_{BR}, y_{BR}) . In some embodiments, x is clipped to be a smaller value of x and x_{BR} . In some embodiments, y is clipped to be a smaller value of y and y_{BR} . In some embodiments, x is clipped to be a larger value of x and x_{TL} . In some embodiments, y is clipped to be a larger value of y and y_{TL} .

[00182] In some embodiments, whether the rule is applicable is based on a syntax element in the bitstream representation. In some embodiments, the syntax element is signaled in a video

parameter set, a dependency parameter set, a slice parameter set, a picture parameter set, an active parameter set, a slice header, or a tile group header. In some embodiments, the syntax element comprises `subpic_treated_as_pic_flag[SubPicIdx]`, where `SubPicIdx` is a subpicture index of the specific subpicture that covers the current block.

[00183] In some embodiments, the conversion generates the current block from the bitstream representation. In some embodiments, the conversion generates the bitstream representation from the current block.

[00184] FIG. 10 is a flowchart representation of a method 1000 for video processing in accordance with the present technology. The method 1000 includes, at operation 1010, resetting, after a conversion of a sub-picture of a video picture of a video and a bitstream representation of the video, a table of motion candidates derived based on past conversions. The method 1000 also includes, at operation 1020, performing a conversion of a subsequent sub-picture of the video picture and the bitstream representation using the table after the resetting.

[00185] In some embodiments, the table of motion candidates comprises motion candidates for an intra-block coding mode. In some embodiments, the table of motion candidates comprises motion candidates for an inter coding mode. In some embodiments, the table of motion candidates comprises motion candidates for an intra coding mode.

[00186] In some embodiments, the conversion generates the sub-picture or the subsequent sub-picture from the bitstream representation. In some embodiments, the conversion generates the bitstream representation from the sub-picture or the subsequent sub-picture.

[00187] FIG. 11 is a flowchart representation of a method 1100 for video processing in accordance with the present technology. The method 1100 includes, at operation 1110, performing a conversion between a video comprising a video picture that includes multiple sub-pictures and multiple video blocks and a coded representation of the video according to a rule. The rule specifies that a boundary between any two sub-pictures is also a boundary between two video blocks. A video block in the video picture is covered by a single subpicture of the video picture.

[00188] In some embodiments, any position in the video picture is covered by at most one subpicture of the video picture. In some embodiments, a position in the video picture is not covered by any subpicture of the video picture. In some embodiments, the two sub-pictures of the video picture have no overlapping area.

[00189] In some embodiments, a dimension of a subpicture of the video picture is determined based on a dimension of a video block. In some embodiments, a sub-picture comprises multiple elements. A syntax element indicating a width of an element in the sub-picture is represented as N samples, and a width of the sub-picture is determined based on the N samples. In some embodiments, a width of the video block comprises N samples.

[00190] In some embodiments, a sub-picture comprises multiple elements. A syntax element indicating a height of an element in the sub-picture is represented as N samples, and a height of the sub-picture is determined based on the N samples. In some embodiments, a height of the video block comprises N samples.

[00191] In some embodiments, the video block is a coding tree block (CTB) or a coding tree unit (CTU). In some embodiments, a sub-picture has a rectangular shape. In some embodiments, the sub-picture comprises rectangular slices. In some embodiments, a sub-picture is applicable to only selected one or more layers of the video. In some embodiments, the sub-picture is defined in a sequence parameter set in the bitstream representation, and the sub-picture is applicable to a layer with a highest resolution in a corresponding sequence. In some embodiments, the sub-picture is defined in a sequence parameter set in the bitstream representation, and the sub-picture is applicable to a layer with a lowest resolution in a corresponding sequence. In some embodiments, the selected one or more layers to which the sub-picture is applicable are signaled in one or more syntax elements in the bitstream representation. In some embodiments, one or more layers to which the sub-picture is inapplicable are signaled in one or more syntax elements in the bitstream representation. In some embodiments, the one or more syntax elements are signaled in a sequence parameter set, a video parameter set, or a picture parameter set in the bitstream representation.

[00192] FIG. 12 is a flowchart representation of a method 1200 for video processing in accordance with the present technology. The method 1200 includes, at operation 1210, performing a conversion between a video unit of a video and a coded representation of the video using at least a video picture, where only one of a sub-picture coding mode or a resolution-changing coding mode is enabled for the video unit. The sub-picture coding mode is a mode in which the video picture is divided into multiple sub-pictures, and the resolution-changing coding mode is a mode in which a resolution of the video picture is adjusted during the conversion.

[00193] In some embodiments, the video picture comprises a current picture or a reference picture.

In some embodiments, the resolution-changing coding mode comprises a Reference Picture Resampling (PRP) mode. In some embodiments, the resolution-changing coding mode comprises a Dynamic Resolution Conversion (DRC) mode. In some embodiments, the resolution-changing coding mode comprises an Adaptive Resolution Conversion (ARC) mode.

[00194] In some embodiments, the video unit comprises a video sequence. In some embodiments, a syntax element is included in the coded representation to indicate that the sub-picture coding mode is enabled for the coding unit in case the resolution-changing coding mode is disallowed. In some embodiments, the resolution-changing coding mode is disallowed in case a syntax element is included in the coded representation to indicate that the sub-picture coding mode is enabled. In some embodiments, the syntax element comprises `subpics_present_flag`. In some embodiments, a width of the video picture is set to be equal to a maximum width allowed for video pictures in the video unit in case the syntax element indicates that the sub-picture coding mode is enabled.

[00195] FIG. 13 is a flowchart representation of a method 1300 for video processing in accordance with the present technology. The method 1300 includes, at operation 1310, performing a conversion between a video unit of a video and a coded representation of the video using at least a video picture, where both a sub-picture coding mode and a resolution-changing coding mode are enabled for the video unit. The sub-picture coding mode is a mode in which the video picture is divided into multiple sub-pictures, and the resolution-changing coding mode is a mode in which a resolution of the video picture is adjusted during the conversion.

[00196] In some embodiments, the video unit comprises a video sequence. In some embodiments, the resolution-changing coding mode comprises an Adaptive Resolution Conversion (ARC) mode, a Dynamic Resolution Conversion (DRC) mode, a Reference Picture Resampling (PRP) mode.

[00197] In some embodiments, the video picture includes multiple video blocks, each having a dimension of $W \times H$. A sub-picture adjusted according to the resolution-changing coding mode has a width of $K \times W$ and a height of $M \times H$, K and M being integers. In some embodiments, the sub-picture is not located at a boundary of the video picture. In some embodiments, the boundary comprises a right boundary or a bottom boundary.

[00198] In some embodiments, the video picture includes multiple video blocks, and a dimension of individual video blocks is adjusted based on a resolution of the video picture. In some embodiments, the coded representation comprises a syntax element indicating a maximum dimension for a video block, and the dimension of an individual video block is adjusted based on

the maximum dimension and the resolution of the video picture. In some embodiments, the dimension of the individual video blocks is signaled in a sequence parameter set, a picture parameter set, or at a sub-picture level in the coded representation.

[00199] FIG. 14 is a flowchart representation of a method 1400 for video processing in accordance with the present technology. The method 1400 includes, at operation 1410, performing a conversion between a video comprising one or more video pictures and a coded representation of the video, where a dimension of an individual video picture is constrained to be greater than or equal to 8. In some embodiments, the dimension is a width of the individual video picture. In some embodiments, the dimension is a height of the individual video picture.

[00200] FIG. 15 is a flowchart representation of a method 1500 for video processing in accordance with the present technology. The method 1500 includes, at operation 1510, performing a conversion between a video picture of a video and a coded representation of the video according to a rule. The video picture comprises at least one sub-picture, and the rule specifies that a characteristic of a sub-picture is represented as at least one syntax element in the coded representation, the at least one syntax element being different than an index value of the sub-picture grid in the video picture.

[00201] In some embodiments, the characteristic comprises a top-left position of the sub-picture. In some embodiments, the characteristic comprises a bottom-right position of the sub-picture. In some embodiments, the characteristic comprises a width of the sub-picture. In some embodiments, the characteristic comprises a height of the sub-picture. In some embodiments, the index value of the sub-picture in the video picture is smaller than a maximum number of subpictures in the video picture. In some embodiments, an integer value in a range of $[0, \text{the maximum number of subpictures}-1]$ has a one-to-one correspondence with indices values of subpictures in the video picture.

[00202] FIG. 16 is a flowchart representation of a method 1600 for video processing in accordance with the present technology. The method 1600 includes, at operation 1610, performing a conversion between a video picture of a video and a coded representation of the video according to a rule. The video picture comprises multiple sub-pictures, each sub-picture comprising multiple elements. The rule specifies that a dimension of individual elements in a sub-picture satisfies a constraint.

[00203] In some embodiments, the constraint specifies that a width of an individual element is

smaller than T1. In some embodiments, the constraint specifies that a height of an individual element is smaller than T2. In some embodiments, the video picture comprises multiple video blocks, and the constraint specifies that the sub-picture determined based on the dimension of the individual elements is covered by a current video block being processed during the conversion. In some embodiments, a sample outside of the sub-picture covered by the current video block is considered as unavailable for the conversion. In some embodiments, the constraint specifies that the dimension of the individual elements is determined based on a profile, a level, or a tier of a video coding standard. In some embodiments, the constraint specifies that the dimension of the individual elements is determined based on a width of the video picture. In some embodiments, a width of the individual elements is equal to $\text{pic_width_max_in_luma_samples}/4 + \text{offset}$, where $\text{pic_width_max_in_luma_samples}$ represents a maximum picture width in luma samples and offset is zero or a non-zero integer. In some embodiments, the constraint specifies that the dimension of the individual elements is determined based on a height of the video picture. In some embodiments, a height of the individual elements is equal to $\text{pic_height_max_in_luma_samples}/4 + \text{offset}$, where $\text{pic_height_max_in_luma_samples}$ represents a maximum picture height in luma samples and offset is zero or a non-zero integer.

[00204] In some embodiments, the conversion generates video picture from the bitstream representation. In some embodiments, the conversion generates the bitstream representation from the video picture.

[00205] FIG. 17 is a flowchart representation of a method 1700 for video processing in accordance with the present technology. The method 1700 includes, at operation 1710, performing a conversion between a video comprising a picture that includes multiple sub-pictures and a coded representation of the video using a coding mode according to a rule. The rule specifies that certain stored information about a previous sub-picture is reset prior to processing each next sub-picture of the multiple sub-pictures.

[00206] In some embodiments, the certain stored information comprises a virtual buffer used for an intra-block copy coding mode in which a current block in the video picture is coded with samples from the video picture. In some embodiments, the certain stored information comprises a list of entries used for a palette coding mode in which a current block in the video picture is coded using a palette of representative sample values.

[00207] FIG. 18 is a flowchart representation of a method 1800 for video processing in accordance

with the present technology. The method 1800 includes, at operation 1810, performing a temporal filtering operation in a conversion between a video and a coded representation of the video according to a rule. The video comprises multiple video pictures, each comprising multiple sub-pictures. The rule specifies that, for a temporal filtering a current sample in a current sub-picture of a video picture, only samples within the same current sub-picture or a sub-picture in a different video picture corresponding to the current sub-picture are available.

[00208] FIG. 19 is a flowchart representation of a method 1900 for video processing in accordance with the present technology. The method 1900 includes, at operation 1910, determining, for a conversion between a block in a video picture of a video and a coded representation of the video, a manner of applying a partitioning method to the block based on whether the block crosses one or more sub-picture boundaries of the video picture. The method 1900 also includes, at operation 1920, performing the conversion based on the determining.

[00209] In some embodiments, the partitioning method comprising at least one of: a quad-tree partitioning method, a horizontal binary tree partitioning method, a vertical binary tree partitioning method, a horizontal tertiary tree partitioning method, a vertical tertiary tree partitioning method, or a no-splitting method. In some embodiments, the manner further specifies whether a boundary handling method is applicable to the block. In some embodiments, the method includes determining a manner of processing a syntax element in the coded representation indicating the partitioning method based on whether the block is located across the one or more boundaries of the sub-picture of the video picture.

[00210] FIG. 20 is a flowchart representation of a method 2000 for video processing in accordance with the present technology. The method 2000 includes, at operation 2010, determining, for a conversion between a video picture of a video and a coded representation of the video, two sub-regions of the video picture. A first sub-region comprises multiple sub-pictures of the video picture and a second sub-region comprises remaining samples in the video picture. The method 2000 also includes, at operation 2020, performing the conversion based on the determining.

[00211] In some embodiments, a sample in the second sub-region is not located in any sub-picture of the video picture. In some embodiments, the second sub-region is processed for the conversion based on information about the first sub-region. In some embodiments, a default value is used in the conversion to indicate whether a sample or an area of the video picture is located within the second sub-region. In some embodiments, the default value is set to

(max_subpics_minus1 + K), wherein max_subpics_minus1 indicates a maximum number of sub-pictures in the video picture, and K is an integer greater than 1. In some embodiments, the default value is assigned to each element in an array of index values representing the sub-pictures in the video picture.

[00212] In some embodiments, the conversion generates the video from the coded representation. In some embodiments, the conversion generates the coded representation from the video.

[00213] Some embodiments of the disclosed technology include making a decision or determination to enable a video processing tool or mode. In an example, when the video processing tool or mode is enabled, the encoder will use or implement the tool or mode in the processing of a block of video, but may not necessarily modify the resulting bitstream based on the usage of the tool or mode. That is, a conversion from the block of video to the bitstream representation of the video will use the video processing tool or mode when it is enabled based on the decision or determination. In another example, when the video processing tool or mode is enabled, the decoder will process the bitstream with the knowledge that the bitstream has been modified based on the video processing tool or mode. That is, a conversion from the bitstream representation of the video to the block of video will be performed using the video processing tool or mode that was enabled based on the decision or determination.

[00214] Some embodiments of the disclosed technology include making a decision or determination to disable a video processing tool or mode. In an example, when the video processing tool or mode is disabled, the encoder will not use the tool or mode in the conversion of the block of video to the bitstream representation of the video. In another example, when the video processing tool or mode is disabled, the decoder will process the bitstream with the knowledge that the bitstream has not been modified using the video processing tool or mode that was enabled based on the decision or determination.

[00215] The disclosed and other solutions, examples, embodiments, modules and the functional operations described in this document can be implemented in digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this document and their structural equivalents, or in combinations of one or more of them. The disclosed and other embodiments can be implemented as one or more computer program products, e.g., one or more modules of computer program instructions encoded on a computer readable medium for

execution by, or to control the operation of, data processing apparatus. The computer readable medium can be a machine-readable storage device, a machine-readable storage substrate, a memory device, a composition of matter effecting a machine-readable propagated signal, or a combination of one or more them. The term “data processing apparatus” encompasses all apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them. A propagated signal is an artificially generated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal, that is generated to encode information for transmission to suitable receiver apparatus.

[00216] A computer program (also known as a program, software, software application, script, or code) can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program does not necessarily correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data (e.g., one or more scripts stored in a markup language document), in a single file dedicated to the program in question, or in multiple coordinated files (e.g., files that store one or more modules, sub programs, or portions of code). A computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a communication network.

[00217] The processes and logic flows described in this document can be performed by one or more programmable processors executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit).

[00218] Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a

read only memory or a random-access memory or both. The essential elements of a computer are a processor for performing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Computer readable media suitable for storing computer program instructions and data include all forms of non-volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks; and CD ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[00219] While this patent document contains many specifics, these should not be construed as limitations on the scope of any subject matter or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular techniques. Certain features that are described in this patent document in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

[00220] Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. Moreover, the separation of various system components in the embodiments described in this patent document should not be understood as requiring such separation in all embodiments.

[00221] Only a few implementations and examples are described and other implementations, enhancements and variations can be made based on what is described and illustrated in this patent document.