

TRANSFORM COEFFICIENT CODING

RELATED APPLICATIONS

[0001] This application claims the benefit of:

U.S. Provisional Application No. 61/625,039, filed April 16, 2012, and

U.S. Provisional Application No. 61/667,382, filed July 2, 2012, the entire content each of which is incorporated by reference herein.

TECHNICAL FIELD

[0002] This disclosure relates to video coding and more particularly to techniques for coding syntax elements associated with transform coefficients, used in video coding.

BACKGROUND

[0003] Digital video capabilities can be incorporated into a wide range of devices, including digital televisions, digital direct broadcast systems, wireless broadcast systems, personal digital assistants (PDAs), laptop or desktop computers, tablet computers, e-book readers, digital cameras, digital recording devices, digital media players, video gaming devices, video game consoles, cellular or satellite radio telephones, so-called “smart phones,” video teleconferencing devices, video streaming devices, and the like. Digital video devices implement video compression techniques defined according to video coding standards. Digital video devices may transmit, receive, encode, decode, and/or store digital video information more efficiently by implementing such video compression techniques. Video coding standards include ITU-T H.261, ISO/IEC MPEG-1 Visual, ITU-T H.262 or ISO/IEC MPEG-2 Visual, ITU-T H.263, ISO/IEC MPEG-4 Visual and ITU-T H.264 (also known as ISO/IEC MPEG-4 AVC), including its Scalable Video Coding (SVC) and Multiview Video Coding (MVC) extensions. In addition, High-Efficiency Video Coding (HEVC) is a video coding standard being developed by the Joint Collaboration Team on Video Coding (JCT-VC) of ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Motion Picture Experts Group (MPEG).

[0004] Video compression techniques perform spatial (intra-picture) prediction and/or temporal (inter-picture) prediction to reduce or remove redundancy inherent in video sequences. For block-based video coding, a video slice (i.e., a video frame or a portion of a video frame) may be partitioned into video blocks, which may also be referred to as

treeblocks, coding units (CUs) and/or coding nodes. Video blocks in an intra-coded (I) slice of a picture are encoded using spatial prediction with respect to reference samples in neighboring blocks in the same picture. Video blocks in an inter-coded (P or B) slice of a picture may use spatial prediction with respect to reference samples in neighboring blocks in the same picture or temporal prediction with respect to reference samples in other reference pictures. Pictures may be referred to as frames, and reference pictures may be referred to as reference frames.

[0005] Spatial or temporal prediction results in a predictive block for a block to be coded. Residual data represents pixel differences between the original block to be coded and the predictive block. An inter-coded block is encoded according to a motion vector that points to a block of reference samples forming the predictive block, and the residual data indicating the difference between the coded block and the predictive block. An intra-coded block is encoded according to an intra-coding mode and the residual data. For further compression, the residual data may be transformed from the pixel domain to a transform domain, resulting in residual transform coefficients, which then may be quantized. The quantized transform coefficients, initially arranged in a two-dimensional array, may be scanned in order to produce a one-dimensional vector of transform coefficients, and entropy coding may be applied to achieve even more compression.

SUMMARY

[0006] In general, this disclosure describes techniques for encoding and decoding data representing syntax elements (e.g., significance flags) associated with transform coefficients of a block. In some techniques, a video encoder and a video decoder each determine contexts to be used for context adaptive binary arithmetic coding (CABAC). As described in more detail, the video encoder and the video decoder determine a scan order for the block, and determine the contexts based on the scan order. In some examples, the video decoder determines contexts that are the same for two or more scan orders, and different contexts for other scan orders. Similarly, in these examples, the video encoder determines contexts that are the same for the two or more scan orders, and different contexts for the other scan orders.

[0007] In one example, the disclosure describes a method for decoding video data. The method comprising receiving, from a coded bitstream, significance flags of transform coefficients of a block, determining a scan order for the transform coefficients of the

block, determining contexts for the significance flags of the transform coefficients of the block based on the determined scan order, and context adaptive binary arithmetic coding (CABAC) decoding the significance flags of the transform coefficients based at least on the determined contexts.

[0008] In another example, the disclosure describes a method for encoding video data. The method comprising determining a scan order for transform coefficients of a block, determining contexts for significance flags of the transform coefficients of the block based on the determined scan order, context adaptive binary arithmetic coding (CABAC) encoding the significance flags of the transform coefficients based at least on the determined contexts, and signaling the encoded significance flags in a coded bitstream.

[0009] In another example, the disclosure describes an apparatus for coding video data. The apparatus comprises a video coder configured to determine a scan order for transform coefficients of a block, determine contexts for significance flags of the transform coefficients of the block based on the determined scan order, and context adaptive binary arithmetic coding (CABAC) code the significance flags of the transform coefficients based at least on the determined contexts.

[0010] In another example, the disclosure describes an apparatus for coding video data. The apparatus comprises means for determining a scan order for transform coefficients of a block, means for determining contexts for significance flags of the transform coefficients of the block based on the determined scan order, and means for context adaptive binary arithmetic coding (CABAC) the significance flags of the transform coefficients based at least on the determined contexts.

[0011] In another example, the disclosure describes a computer-readable storage medium. The computer-readable storage medium having instructions stored thereon that when executed cause one or more processors of an apparatus for coding video data to determine a scan order for transform coefficients of a block, determine contexts for significance flags of the transform coefficients of the block based on the determined scan order, and context adaptive binary arithmetic coding (CABAC) code the significance flags of the transform coefficients based at least on the determined contexts.

[0012] The details of one or more examples are set forth in the accompanying drawings and the description below. Other features, objects, and advantages will be apparent from the description and drawings, and from the claims.

BRIEF DESCRIPTION OF DRAWINGS

[0013] FIGS. 1A–1C are conceptual diagrams illustrating examples of scan orders of a block that includes transform coefficients.

[0014] FIG. 2 is a conceptual diagram illustrating a mapping of transform coefficients to significance syntax elements.

[0015] FIG. 3 is a block diagram illustrating an example video encoding and decoding system that may utilize techniques described in this disclosure.

[0016] FIG. 4 is a block diagram illustrating an example video encoder that may implement techniques described in this disclosure.

[0017] FIG. 5 is a block diagram illustrating an example of an entropy encoder that may implement techniques for entropy encoding syntax elements in accordance with this disclosure.

[0018] FIG. 6 is a flowchart illustrating an example process for encoding video data according to this disclosure.

[0019] FIG. 7 is a block diagram illustrating an example video decoder that may implement techniques described in this disclosure.

[0020] FIG. 8 is a block diagram illustrating an example of an entropy decoder that may implement techniques for decoding syntax elements in accordance with this disclosure.

[0021] FIG. 9 is a flowchart illustrating an example process of decoding video data according to this disclosure.

[0022] FIG. 10 is a conceptual diagram illustrating positions of a last significant coefficient depending on the scan order.

[0023] FIG. 11 is a conceptual diagram illustrating use of a diagonal scan in place of an original horizontal scan.

[0024] FIG. 12 is a conceptual diagram illustrating a context neighborhood for a nominal horizontal scan.

DETAILED DESCRIPTION

[0025] A video encoder determines transform coefficients for a block, encodes syntax elements, that indicate the values of the transform coefficients, using context adaptive binary arithmetic coding (CABAC), and signals the encoded syntax elements in a bitstream. A video decoder receives the bitstream that includes the encoded syntax elements that indicate the values of the transform coefficients and CABAC decodes the syntax elements to determine the transform coefficients for the block.

[0026] The video encoder and video decoder determine which contexts are to be used to perform CABAC encoding and CABAC decoding, respectively. In the techniques described in this disclosure, the video encoder and the video decoder may determine which contexts to use to perform CABAC encoding or CABAC decoding based on a scan order of the block of the transform coefficients. In some examples, the video encoder and the video decoder may determine which contexts to use to perform CABAC encoding or CABAC decoding based on a size of the block, positions of the transform coefficients within the block, and the scan order.

[0027] In some examples, the video encoder and the video decoder may utilize different contexts for different scan orders (i.e., a first set of contexts for horizontal scan, a second set of contexts for vertical scan, and a third set of contexts for diagonal scan). As another example, if the block of transform coefficients is scanned vertically or horizontally, the video encoder and the video decoder may utilize the same contexts for both of these scan orders (e.g., for a particular position of a transform coefficient).

[0028] By determining which contexts to use for CABAC encoding or CABAC decoding, the techniques described in this disclosure may exploit the statistical behavior of the magnitudes of the transform coefficients in a way that achieves better video compression, as compared to other techniques. For instance, it may be possible for the video encoder and the video decoder to determine which contexts to use for CABAC encoding or CABAC decoding based on the position of the transform coefficient, irrespective of the scan order. However, the scan order may have an effect on the ordering of the transform coefficients.

[0029] For example, the block of transform coefficients may be a two-dimensional (2D) block of coefficients that the video encoder scans to construct a one-dimensional (1D) vector, and the video encoder entropy encodes (using CABAC) the values of the transform coefficients in the 1D vector. The order in which the video encoder places the values (e.g., magnitudes) of the transform coefficients in the 1D vector is a function of the scan order. The order in which the video encoder places the magnitudes of the transform coefficients for a diagonal scan may be different than the order in which the video encoder places the magnitudes of the transform coefficients for a vertical scan.

[0030] In other words, the position of the magnitudes of the transform coefficients may be different for different scan orders. The position of the magnitudes of the transform coefficients may have an effect on coding efficiency. For instance, the location of the last significant coefficient, in the block, may be different for different scan orders. In

this case, the magnitude of the last significant coefficient may be different for different scan orders.

[0031] Accordingly, these other techniques that determine contexts based on the position of the transform coefficient irrespective to the scan order fail to properly account for the potential that the significance statistics for a transform coefficient in a particular position may vary depending on the scan order. In the techniques described in this disclosure, the video encoder and video decoder may determine the scan order for the block, and determine contexts based on the determined scan order (and in some examples, also based on the positions of the transform coefficients and possibly the size of the block). This way, the video encoder and video decoder may better account for the significance statistics for determining which contexts to use as compared to techniques that do not rely on the scan order and rely only on the position for determining which contexts to use.

[0032] In some examples of video coding, the video encoder and the video decoder may use five coding passes to encode or decode transform coefficients of a block, namely, (1) a significance pass, (2) a greater than one pass, (3) a greater than two pass, (4) a sign pass, and (5) a coefficient level remaining pass. The techniques of this disclosure, however, are not necessarily limited to five pass scenarios. In general, significance coding refers to generating syntax elements to indicate whether any of the coefficients within the block have an absolute value of one or greater. That is, a coefficient with an absolute value of one or greater is considered “significant.” The other coding passes are described in more detail below.

[0033] During the significance pass, the video encoder determines syntax elements that indicate whether a transform coefficient is significant. Syntax elements that indicate whether a transform coefficient is significant are referred to herein as significance syntax elements. One example of a significance syntax element is a significance flag, where a value of 0 for the significance flag indicates that the coefficient is not significant (i.e., the value of the transform coefficient is 0) and a value of 1 for the significance flag indicates that the coefficient is significant (i.e., the value of the transform coefficient is non-zero).

[0034] To perform the significance pass, the video encoder scans the transform coefficients of a block or part of the block (if the position of the last significant position is previously determined and signaled to the decoder), and determines the significance syntax element for each transform coefficient. There are various examples of the scan

order, such as a horizontal scan, a vertical scan, and a diagonal scan. The video encoder CABAC encodes the significance syntax elements and signals the encoded significance syntax elements in a coded bitstream. Other types of scans, such as zig-zag scans, adaptive or partially adaptive scans may also be used in some examples.

[0035] To apply CABAC coding to a syntax element, binarization may be applied to a syntax element to form a series of one or more bits, which are referred to as “bins.” In addition, a coding context may be associated with a bin of the syntax element. The coding context may identify probabilities of coding bins having particular values. For instance, a coding context may indicate a 0.7 probability of coding a 0-valued bin (representing an example of a “most probable symbol,” in this instance) and a 0.3 probability of coding a 1-valued bin. After identifying the coding context, a bin may be arithmetically coded based on the context. In some cases, contexts associated with a particular syntax element or bins thereof may be dependent on other syntax elements or coding parameters.

[0036] In the techniques described in this disclosure, the video encoder may determine which contexts to use for the CABAC encoding based on the scan order. The video encoder may use one set of contexts per scan order type. For example, if the block is a 4x4 block, there are sixteen coefficients. In this example, the video encoder may utilize sixteen contexts for each scan resulting in a total of forty-eight contexts (i.e., sixteen contexts for horizontal scan, sixteen contexts for vertical scan, and sixteen contexts for diagonal scan for a total of forty-eight contexts). The same would hold for an 8x8 block, but with a total of 192 contexts (i.e., sixty-four contexts for horizontal scan, sixty-four contexts for vertical scan, and sixty-four contexts for diagonal scan for a total of 192 contexts). However, the example of forty-eight or 192 contexts is provided for purposes of illustration only. It may be possible that the number of contexts for each block is a function of block size.

[0037] The video decoder receives the coded bitstream (e.g., from the video encoder directly or via a storage medium that stores the coded bitstream) and performs a reciprocal function, as that of the video encoder, to determine the values of the transform coefficients. For example, the video decoder implements the significance pass to determine which transform coefficients are significant based on the significance syntax elements in the received bitstream.

[0038] In the techniques described in this disclosure, the video decoder may determine the scan order of the transform coefficients of the block (e.g., the scan order in which

the transform coefficients were scanned). The video decoder may determine which contexts to use for CABAC decoding the significance syntax elements based on the scan order (e.g., sixteen of the forty-eight contexts for a 4x4 block or sixty-four of the 192 contexts for an 8x8 block). In this manner, the video decoder may select the same contexts for CABAC decoding that video encoder selected for CABAC encoding. The video decoder CABAC decodes the significance syntax elements based on the determined contexts.

[0039] In the above examples, the video encoder and the video decoder determined contexts based on the scan order, where the contexts were different for different scan orders resulting in a total of forty-eight contexts for a 4x4 block and 192 contexts for an 8x8 block. However, the techniques described in this disclosure are not limited in this respect. Alternatively, in some examples, the contexts that the video encoder and the video decoder use may be the same contexts for multiple (i.e., two or more) scan orders to allow for context sharing depending on scan order type.

[0040] As one example, the video encoder and the video decoder may determine contexts that are the same if the scan order is a horizontal scan or if the scan order is a vertical scan. In other words, the contexts are the same if the scan order is the horizontal scan or if the scan order is the vertical scan for a particular position of the transform coefficient within the block. The video encoder and the video decoder may utilize different contexts for the diagonal scan. In this example, the number of contexts for the 4x4 block reduces from forty-eight contexts to thirty-two contexts and for the 8x8 block reduces from 192 contexts to 128 because the contexts for the horizontal scan and the vertical scan are the same, and there are different contexts for the diagonal scan.

[0041] As another example, it may be possible for the video encoder and the video decoder to use the same contexts for all scan order types, which reduces the contexts to sixteen for the 4x4 block and sixty-four for the 8x8 block. However, using the same contexts for all scan order types may be a function of the block size. For example, for certain block sizes, it may be possible to use the same contexts for all scan orders, and for certain other blocks sizes, the contexts may be different for the different scan orders, or two or more of the scan orders may share contexts.

[0042] For instance, for an 8x8 block, the contexts for the horizontal and vertical scans may be the same (e.g., for a particular position), and different for the diagonal scan. For the 4x4, 16x16, and 32x32 blocks, the contexts may be different for different scan orders. Moreover, in some other techniques that relied on position, the contexts for the

2D block and the 1D block may be different. In the techniques described in this disclosure, when contexts are shared for all scan orders, the contexts for the 2D block or the 1D block may be the same.

[0043] In some examples, in addition to utilizing the scan order to determine the contexts, the video encoder and the video decoder may account for the size of the block. For instance, in the above example, the size of the block indicated whether all scan orders share contexts. In some examples, the video encoder and the video decoder may determine which contexts to use based on the size of the block and the scan order. In these examples, the techniques described in this disclosure may allow for context sharing. For instance, for a block with a first size, the video encoder and the video decoder may determine contexts that are the same if the block of the first size is scanned horizontally or if the block of the first size is scanned vertically. For a block with a second size, the video encoder and the video decoder may determine contexts that are the same if the block of the second size is scanned horizontally or if the block of the second size is scanned vertically.

[0044] There may be other variations to these techniques. For example, for certain sized blocks (e.g., 16x16 or 32x32), the video encoder and the video decoder determine a first set of contexts that are used for CABAC encoding or CABAC decoding for all scan orders. For certain sized blocks (e.g., 8x8), the video encoder and the video decoder determines a second set of contexts that are used for CABAC encoding or CABAC decoding for a diagonal scan, and a third set of contexts that are used for CABAC encoding or CABAC decoding for both a horizontal scan and a vertical scan. For certain sized blocks (e.g., 4x4), the video encoder and the video decoder determine a fourth set of contexts that are used for CABAC encoding or CABAC decoding for a diagonal scan, a horizontal scan and a vertical scan.

[0045] In some cases, the examples of determining contexts based on the scan order may be directed to intra-coding modes. For example, the transform coefficients may be the result from intra-coding, and the techniques described in this disclosure may be applicable to such transform coefficients. However, the techniques described in this disclosure are not so limited and may be applicable for inter-coding or intra-coding.

[0046] FIGS. 1A–1C are conceptual diagrams illustrating examples of scan orders of a block that includes transform coefficients. A block that includes transform coefficients may be referred to as a transform block (TB). A transform block may be a block of a transform unit. For example, a transform unit includes three transform blocks and the

corresponding syntax elements. A transform unit may be a transform block of luma samples of size 8x8, 16x16, or 32x32 or four transform blocks of luma samples of size 4x4, two corresponding transform blocks of chroma samples of a picture that three sample arrays, or a transform block of luma samples of size 8x8, 16x16, or 32x32, or four transform blocks of luma samples of size 4x4 or a monochrome picture or a picture that is coded using separate color planes and syntax structures used to transform the transform block samples.

[0047] FIG. 1A illustrates a horizontal scan of 4x4 block 10 (e.g., TB 10) that includes transform coefficients 12A to 12P (collectively referred to as “transform coefficients 12”). For example, the horizontal scan starts from transform coefficient 12P and ends at transform coefficient 12A, and proceeds horizontally through the transform coefficients.

[0048] FIG. 1B illustrates a vertical scan of 4x4 block 14 (e.g., TB 14) that includes transform coefficients 16A to 16P (collectively referred to as “transform coefficients 16”). For example, the vertical scan starts from transform coefficient 16P and ends at transform coefficient 16A, and proceeds vertically through the transform coefficients.

[0049] FIG. 1C illustrates a diagonal scan of 4x4 block 18 (e.g., TB 18) that includes transform coefficients 20A to 20P (collectively referred to as “transform coefficients 20”). For example, the diagonal scan starts from transform coefficient 20P and ends at transform coefficient 20A, and proceeds diagonally through the transform coefficients.

[0050] It should be understood that although FIGS. 1A–1C illustrate starting from the last transform coefficient and ending on the first transform coefficient, the techniques of this disclosure are not so limited. In some examples, the video encoder may determine the location of the last significant coefficient (e.g., the last transform coefficient with a non-zero value) in the block. The video encoder may scan starting from the last significant coefficient and ending on the first transform coefficient. The video encoder may signal the location of the last significant coefficient in the coded bitstream (i.e., x and y coordinate of the last significant coefficient), and the video decoder may receive the location of the last significant coefficient from the coded bitstream. In this manner, the video decoder may determine that subsequent syntax elements for the transform coefficients (e.g., the significance syntax elements) are for transform coefficients starting from the last significant coefficient and ending on the first transform coefficient.

[0051] Although FIGS. 1A–1C are illustrated as 4x4 blocks, the techniques described in this disclosure are not so limited, and the techniques can be extended to other sized blocks. Moreover, in some cases, one or more of 4x4 blocks 10, 14, and 18 may be sub-

blocks of a larger block. For example, an 8x8 block can be divided into four 4x4 sub-blocks, a 16x16 can be divided into sixteen 4x4 sub-blocks, and so forth, and one or more of 4x4 blocks 10, 14, and 18 may be sub-blocks of the 8x8 block or 16x16 block. Examples of sub-block horizontal and vertical scans are described in: (1) Rosewarne, C., Maeda, M. "Non-CE11: Harmonisation of 8x8 TU residual scan" JCT-VC Contribution JCTVC-H0145; (2) Yu, Y., Panusopone, K., Lou, J., Wang, L. "Adaptive Scan for Large Blocks for HEVC; JCT-VC Contribution JCTVC-F569; and (3) U.S. Patent Application No. 13/551,458, filed July 17, 2012, each of which is hereby incorporated by reference.

[0052] Transform coefficients 12, 16, and 20 represent transformed residual values between a block that is being predicted and another block. The video encoder generates significance syntax elements that indicate whether the values of transform coefficients 12, 16, and 20 are zero or non-zero, encodes the significance syntax elements, and signals the encoded significance syntax elements in a coded bitstream. The video decoder receives the coded bitstream and decodes the significance syntax elements as part of the process of determining transform coefficients 12, 16, and 20.

[0053] For encoding and decoding, the video encoder and the video decoder determine contexts that are to be used for context adaptive binary arithmetic coding (CABAC) encoding and decoding. In the techniques described in this disclosure, to determine the contexts for the significance syntax elements for transform coefficients 12, 16, and 20, the video encoder and the video decoder account for the scan order.

[0054] For example, if the video encoder and the video decoder determine that the scan order is a horizontal scan, then the video encoder and the video decoder may determine a first set of contexts for the sixteen transform coefficients 12 of TU 10. If the video encoder and the video decoder determine that the scan order is a vertical scan, then the video encoder and the video decoder may determine a second set of contexts for the sixteen transform coefficients 16 of TU 14. If the video encoder and the video decoder determine that the scan order is a diagonal scan, then the video encoder and the video decoder may determine a third set of contexts for the sixteen transform coefficients 20 of TU 18.

[0055] In this example, assuming no context sharing, there are a total of forty-eight contexts for the 4x4 blocks 10, 14, and 18 (i.e., sixteen contexts for each of the three scan orders). If blocks 10, 14, and 18 were 8x8 sized blocks, assuming no context

sharing, then there would sixty-four contexts for each of the three 8x8 sized blocks, for a total of 192 contexts (i.e., sixty-four contexts for each of the three scan orders).

[0056] As described in more detail, in some examples, it may be possible for two or more scan orders to share contexts. For example, two or more of the first set of contexts, second set of contexts, and the third set of contexts may be the same set of contexts. For instance, the first set of contexts for the horizontal scan may be the same as the second set of contexts for the vertical scan. In some cases, the first, second, and third contexts may be the same set of contexts.

[0057] In the above examples, the video encoder and the video decoder determine from a first, second, and third set of contexts the contexts to use for CABAC encoding and decoding based on the scan order. In some examples, the video encoder and the video decoder determine which contexts to use for CABAC encoding and decoding based on the scan order and a size of the block.

[0058] For example, if the block is 8x8, then the video encoder and the video decoder determine contexts from a fourth, fifth, and sixth set of contexts (one for each scan order) based on the scan order. If the block is 16x16, then the video encoder and the video decoder determine contexts from a seventh, eighth, and ninth set of contexts (one for each scan order) based on the scan order, and so forth. Similar to above, in some examples, there may be context sharing for the different sized blocks.

[0059] There may be variants of the above example techniques. For example, in one case, for a particular sized block (e.g., 4x4), the video encoder and video decoder determine contexts that are the same for all scan orders, but for an 8x8 sized block, the video encoder and the video determine the contexts that are the same for a horizontal scan and a vertical scan (e.g., for transform coefficients in particular positions), and different contexts for the diagonal scan. As another example, for larger sized blocks (e.g., 16x16 and 32x32), the video encoder and the video decoder may determine contexts that are the same for all scan orders and for both sizes. In some examples, for the 16x16 and 32x32 blocks, horizontal and vertical scans may not be applied. Other such permutations and combinations are possible, and are contemplated by this disclosure.

[0060] Determining which contexts to use for CABAC encoding and decoding based on the scan order may better account for the magnitudes of the transform coefficients. For example, the scan order defines the arrangement of the transform coefficients. As one example, the magnitude of the first transform coefficient (referred to as the DC

coefficient) is generally the highest. The magnitude of the second transform coefficient is the next highest (on average, but not necessarily), and so forth. However, the location of the second transform coefficient is based on the scan order. For example, in FIG. 1A, the second transform coefficient is the transform coefficient immediately to the right of the first transform coefficient (i.e., immediately right of transform coefficient 12A). However, in FIGS. 1B and 1C, the second transform coefficient is the transform coefficient immediately below the first transform coefficient (i.e., immediately below transform coefficient 16A in FIG. 1B and immediately below transform coefficient 20A in FIG. 1C).

[0061] In this way, the significance statistics for a transform coefficient in a particular scan position may vary depending on the scan order. For example, in FIG. 1A, for the horizontal scan, the last transform coefficient in the first row may have much higher magnitude (on average) compared to the same transform coefficient in the vertical scan of FIG. 1B or the diagonal scan of FIG. 1C.

[0062] By determining which contexts to use based on the scan order, the video encoder and the video decoder may be configured to better CABAC encode or CABAC decode as compared to other techniques that do not account for the scan order. For example, it may be possible that the encoding and decoding of the significance syntax elements (e.g., significance flags) for 4x4 and 8x8 blocks is position based. For instance, there is a separate context for each position in a 4x4 block and a separate context for each 2x2 sub-block of an 8x8 block.

[0063] However, in this case, the context is based on the location of the transform coefficient, irrespective of the actual scan order (i.e., position based contexts for 4x4 and 8x8 blocks do not distinguish between the various scans). For example, the context for a transform coefficient located at (i, j) in the block is the same for the horizontal, vertical, and diagonal scans. As described above, the scan order may have an effect on the significance statistics for the transform coefficients, and the techniques described in this disclosure may determine contexts based on the scan order to account for the significance statistics.

[0064] As described above, in some examples, the video encoder and the video decoder may determine contexts that are the same for two or more scan orders. There may be various ways in which the video encoder and the video decoder may determine contexts that are the same for two or more scan orders for particular locations of transform coefficients. As one example, the horizontal and the vertical scan orders share the

contexts for a particular block size by sharing contexts between the horizontal scan and a transpose of the block of the vertical scan. For instance, the video encoder and the video decoder may determine the same context for a transform coefficient (i, j) for the horizontal scan and a transform coefficient (j, i) for a vertical scan for a particular block size.

[0065] This instance is one example of where transform coefficients at a particular position share contexts for different scan orders. For example, the context for the transform coefficient at position (i, j) for a horizontal scan and the context for the transform coefficient at position (j, i) for a vertical scan may be the same context. In some examples, the sharing of the contexts may be applicable for 8x8 sized blocks of transform coefficients. Also, in some examples, if the scan order is not horizontal or vertical (e.g., diagonal), the context for position (i, j) and/or (j, i) may be different than for the shared context for horizontal and vertical scan.

[0066] However, the techniques described in this disclosure are not so limited, and should not be considered limited to examples where the contexts for a transform coefficient (i, j) for the horizontal scan and a transform coefficient (j, i) for a vertical scan for a particular block size are the same. The following is another example manner in which the contexts for transform coefficients at particular positions are shared for different scan orders.

[0067] For instance, the contexts for the fourth (last) row of the block, for the horizontal scan, may be same as the contexts for the fourth (last) column of the block, for the vertical scan, the contexts for the third row of the block, for the horizontal scan, may be the same the contexts for the third column of the block, for the vertical scan, the contexts for the second row of the block, for the horizontal scan, may be the same the contexts for the second column of the block, for the vertical scan, and the contexts for the first row of the block, for the horizontal scan, may be the same the contexts for the first column of the block, for the vertical scan. The same may be applied to 8x8 blocks. There may be other example ways for the video encoder and the video decoder to determine contexts that are the same for two or more of the scan orders.

[0068] In some examples, it may be possible for contexts to be shared between different block sizes (e.g., shared between a 4x4 block and an 8x8 block). As an example, the context for transform coefficient $(1, 1)$ in a 4x4 block and the context for transform coefficients $(2, 2)$, $(2, 3)$, $(3, 2)$, and $(3, 3)$ in an 8x8 block may be the same, and in some examples, may be the same for a particular scan order.

[0069] FIG. 2 is a conceptual diagram illustrating a mapping of transform coefficients to significance syntax elements. For example, the left side of FIG. 2 illustrates transform coefficients values and the right side of FIG. 2 illustrates corresponding significance syntax elements. For all transform coefficients whose values are non-zero, there is a corresponding significance syntax element (e.g., significance flag) with a value of 1. For all transform coefficients whose values are 0, there is a corresponding significance syntax element (e.g., significance flag) with a value of 0. In the examples described in this disclosure, the video encoder and the video decoder are configured to CABAC encode and CABAC decode the example significance syntax elements illustrated in FIG. 2 by determining contexts based on the scan order, and in some examples, also based on positions of the transform coefficients and the size of the block.

[0070] FIG. 3 is a block diagram illustrating an example video encoding and decoding system 22 that may be configured to assign contexts utilizing the techniques described in this disclosure. As shown in FIG. 3, system 22 includes a source device 24 that generates encoded video data to be decoded at a later time by a destination device 26. Source device 24 and destination device 26 may comprise any of a wide range of devices, including desktop computers, notebook (i.e., laptop) computers, tablet computers, set-top boxes, telephone handsets such as so-called “smart” phones, so-called “smart” pads, televisions, cameras, display devices, digital media players, video gaming consoles, video streaming device, or the like. In some cases, source device 24 and destination device 26 may be equipped for wireless communication.

[0071] Destination device 26 may receive the encoded video data to be decoded via a link 28. Link 28 may comprise any type of medium or device capable of moving the encoded video data from source device 24 to destination device 26. In one example, link 28 may comprise a communication medium to enable source device 24 to transmit encoded video data directly to destination device 26 in real-time. The encoded video data may be modulated according to a communication standard, such as a wireless communication protocol, and transmitted to destination device 26. The communication medium may comprise any wireless or wired communication medium, such as a radio frequency (RF) spectrum or one or more physical transmission lines. The communication medium may form part of a packet-based network, such as a local area network, a wide-area network, or a global network such as the Internet. The communication medium may include routers, switches, base stations, or any other

equipment that may be useful to facilitate communication from source device 24 to destination device 26.

[0072] Alternatively, encoded data may be output from output interface 34 to a storage device 38. Similarly, encoded data may be accessed from storage device 38 by input interface 40. Storage device 38 may include any of a variety of distributed or locally accessed data storage media such as a hard drive, Blu-ray discs, DVDs, CD-ROMs, flash memory, volatile or non-volatile memory, or any other suitable digital storage media for storing encoded video data. In a further example, storage device 38 may correspond to a file server or another intermediate storage device that may hold the encoded video generated by source device 24. Destination device 26 may access stored video data from storage device 38 via streaming or download. The file server may be any type of server capable of storing encoded video data and transmitting that encoded video data to the destination device 26. Example file servers include a web server (e.g., for a website), an FTP server, network attached storage (NAS) devices, or a local disk drive. Destination device 26 may access the encoded video data through any standard data connection, including an Internet connection. This may include a wireless channel (e.g., a Wi-Fi connection), a wired connection (e.g., DSL, cable modem, etc.), or a combination of both that is suitable for accessing encoded video data stored on a file server. The transmission of encoded video data from storage device 38 may be a streaming transmission, a download transmission, or a combination of both.

[0073] The techniques of this disclosure are not necessarily limited to wireless applications or settings. The techniques may be applied to video coding in support of any of a variety of multimedia applications, such as over-the-air television broadcasts, cable television transmissions, satellite television transmissions, streaming video transmissions, e.g., via the Internet, encoding of digital video for storage on a data storage medium, decoding of digital video stored on a data storage medium, or other applications. In some examples, system 22 may be configured to support one-way or two-way video transmission to support applications such as video streaming, video playback, video broadcasting, and/or video telephony.

[0074] In the example of FIG. 3, source device 24 includes a video source 30, video encoder 32 and an output interface 34. In some cases, output interface 34 may include a modulator/demodulator (modem) and/or a transmitter. In source device 24, video source 30 may include a source such as a video capture device, e.g., a video camera, a video archive containing previously captured video, a video feed interface to receive

video from a video content provider, and/or a computer graphics system for generating computer graphics data as the source video, or a combination of such sources. As one example, if video source 30 is a video camera, source device 24 and destination device 26 may form so-called camera phones or video phones. However, the techniques described in this disclosure may be applicable to video coding in general, and may be applied to wireless and/or wired applications.

[0075] The captured, pre-captured, or computer-generated video may be encoded by video encoder 32. The encoded video data may be transmitted directly to destination device 26 via output interface 34 of source device 24. The encoded video data may also (or alternatively) be stored onto storage device 38 for later access by destination device 26 or other devices, for decoding and/or playback.

[0076] Destination device 26 includes an input interface 40, a video decoder 42, and a display device 44. In some cases, input interface 40 may include a receiver and/or a modem. Input interface 40 of destination device 26 receives the encoded video data over link 28. The encoded video data communicated over link 28, or provided on storage device 38, may include a variety of syntax elements generated by video encoder 32 for use by a video decoder, such as video decoder 42, in decoding the video data. Such syntax elements may be included with the encoded video data transmitted on a communication medium, stored on a storage medium, or stored a file server.

[0077] Display device 44 may be integrated with, or external to, destination device 26. In some examples, destination device 26 may include an integrated display device and also be configured to interface with an external display device. In other examples, destination device 26 may be a display device. In general, display device 44 displays the decoded video data to a user, and may comprise any of a variety of display devices such as a liquid crystal display (LCD), a plasma display, an organic light emitting diode (OLED) display, or another type of display device.

[0078] Video encoder 32 and video decoder 42 may operate according to a video compression standard, such as the ITU-T H.264 standard, alternatively referred to as MPEG-4, Part 10, Advanced Video Coding (AVC), or extensions of such standards. Alternatively, video encoder 32 and video decoder 42 may operate according to other proprietary or industry standards, such as the High Efficiency Video Coding (HEVC) standard, and may conform to the HEVC Test Model (HM). The techniques of this disclosure, however, are not limited to any particular coding standard. Other examples of video compression standards include MPEG-2 and ITU-T H.263.

[0079] Although not shown in FIG. 3, in some aspects, video encoder 32 and video decoder 42 may each be integrated with an audio encoder and decoder, and may include appropriate MUX-DEMUX units, or other hardware and software, to handle encoding of both audio and video in a common data stream or separate data streams. If applicable, in some examples, MUX-DEMUX units may conform to the ITU H.223 multiplexer protocol, or other protocols such as the user datagram protocol (UDP).

[0080] Video encoder 32 and video decoder 42 each may be implemented as any of a variety of suitable encoder circuitry, such as one or more microprocessors, digital signal processors (DSPs), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), discrete logic, software, hardware, firmware or any combinations thereof. When the techniques are implemented partially in software, a device may store instructions for the software in a suitable, computer-readable storage medium and execute the instructions in hardware using one or more processors to perform the techniques of this disclosure. Each of video encoder 32 and video decoder 42 may be included in one or more encoders or decoders, either of which may be integrated as part of a combined encoder/decoder (CODEC) in a respective device. For example, the device that includes video decoder 42 may be microprocessor, an integrated circuit (IC), or a wireless communication device that includes video decoder 42.

[0081] The JCT-VC is working on development of the HEVC standard. The HEVC standardization efforts are based on an evolving model of a video coding device referred to as the HEVC Test Model (HM). The HM presumes several additional capabilities of video coding devices relative to existing devices according to, e.g., ITU-T H.264/AVC. For example, whereas H.264 provides nine intra-prediction encoding modes, the HM may provide as many as thirty-five intra-prediction encoding modes.

[0082] In general, the working model of the HM describes that a video frame or picture may be divided into a sequence of treeblocks or largest coding units (LCU) that include both luma and chroma samples. A treeblock has a similar purpose as a macroblock of the H.264 standard. A slice includes a number of consecutive treeblocks in coding order. A video frame or picture may be partitioned into one or more slices. Each treeblock may be split into coding units (CUs) according to a quadtree. For example, a treeblock, as a root node of the quadtree, may be split into four child nodes, and each child node may in turn be a parent node and be split into another four child nodes. A final, unsplit child node, as a leaf node of the quadtree, comprises a coding node, i.e., a coded video block. Syntax data associated with a coded bitstream may define a

maximum number of times a treeblock may be split, and may also define a minimum size of the coding nodes.

[0083] A CU includes a coding node and prediction units (PUs) and transform units (TUs) associated with the coding node. As described above, a transform unit includes one or more transform blocks, and the techniques described in this disclosure are related to determining contexts for the significance syntax elements for the transform coefficients of a transform block based on a scan order and, in some examples, based on a scan order and size of the transform block. A size of the CU corresponds to a size of the coding node and must be square in shape. The size of the CU may range from 8x8 pixels up to the size of the treeblock with a maximum of 64x64 pixels or greater. Each CU may contain one or more PUs and one or more TUs. Syntax data associated with a CU may describe, for example, partitioning of the CU into one or more PUs.

Partitioning modes may differ between whether the CU is skip or direct mode encoded, intra-prediction mode encoded, or inter-prediction mode encoded. PUs may be partitioned to be non-square in shape. Syntax data associated with a CU may also describe, for example, partitioning of the CU into one or more TUs according to a quadtree.

[0084] A TU can be square or non-square in shape. Again, a TU includes one or more transform blocks (TBs) (e.g., one TB for the luma samples, one TB for the first chroma samples, and one TB for the second chroma samples). In this sense, a TU can be considered conceptually as including these TBs, and these TBs can be square or non-square in shape. For example, in this disclosure, the term TU is used to generically refer to the TBs, and the example techniques described in this disclosure are described with respect to a TB.

[0085] The HEVC standard allows for transformations according to TUs, which may be different for different CUs. The TUs are typically sized based on the size of PUs within a given CU defined for a partitioned LCU, although this may not always be the case. The TUs are typically the same size or smaller than the PUs. In some examples, residual samples corresponding to a CU may be subdivided into smaller units using a quadtree structure known as "residual quad tree" (RQT). The leaf nodes of the RQT may be referred to as transform units (TUs). Pixel difference values associated with the TUs may be transformed to produce transform coefficients, which may be quantized.

[0086] In general, a PU includes data related to the prediction process. For example, when the PU is intra-mode encoded (intra-prediction encoded), the PU may include data

describing an intra-prediction mode for the PU. As another example, when the PU is inter-mode encoded (inter-prediction encoded), the PU may include data defining a motion vector for the PU. The data defining the motion vector for a PU may describe, for example, a horizontal component of the motion vector, a vertical component of the motion vector, a resolution for the motion vector (e.g., one-quarter pixel precision or one-eighth pixel precision), a reference picture to which the motion vector points, and/or a reference picture list (e.g., List 0 (L0) or List 1 (L1)) for the motion vector.

[0087] In general, a TU is used for the transform and quantization processes. A given CU having one or more PUs may also include one or more transform units (TUs). The TUs include one or more transform blocks (TBs). Blocks 10, 14, and 18 of FIGS. 1A–1C, respectively, are examples of TBs. Following prediction, video encoder 32 may calculate residual values corresponding to the PU. The residual values comprise pixel difference values that may be transformed into transform coefficients, quantized, and scanned using the TBs to produce serialized transform coefficients for entropy coding. This disclosure typically uses the term “video block” to refer to a coding node of a CU. In some specific cases, this disclosure may also use the term “video block” to refer to a treeblock, i.e., LCU, or a CU, which includes a coding node and PUs. The term “video block” may also refer to transform blocks of a TU.

[0088] For example, for video coding according to the high efficiency video coding (HEVC) standard currently under development, a video picture may be partitioned into coding units (CUs), prediction units (PUs), and transform units (TUs). A CU generally refers to an image region that serves as a basic unit to which various coding tools are applied for video compression. A CU typically has a square geometry, and may be considered to be similar to a so-called “macroblock” under other video coding standards, such as, for example, ITU-T H.264.

[0089] To achieve better coding efficiency, a CU may have a variable size depending on the video data it contains. That is, a CU may be partitioned, or “split” into smaller blocks, or sub-CUs, each of which may also be referred to as a CU. In addition, each CU that is not split into sub-CUs may be further partitioned into one or more PUs and TUs for purposes of prediction and transform of the CU, respectively.

[0090] PUs may be considered to be similar to so-called partitions of a block under other video coding standards, such as H.264. PUs are the basis on which prediction for the block is performed to produce “residual” coefficients. Residual coefficients of a CU represent a difference between video data of the CU and predicted data for the CU

determined using one or more PUs of the CU. Specifically, the one or more PUs specify how the CU is partitioned for the purpose of prediction, and which prediction mode is used to predict the video data contained within each partition of the CU.

[0091] One or more TUs of a CU specify partitions of a block of residual coefficients of the CU on the basis of which a transform is applied to the block to produce a block of residual transform coefficients for the CU. The one or more TUs may also be associated with the type of transform that is applied. The transform converts the residual coefficients from a pixel, or spatial domain to a transform domain, such as a frequency domain. In addition, the one or more TUs may specify parameters on the basis of which quantization is applied to the resulting block of residual transform coefficients to produce a block of quantized residual transform coefficients. The residual transform coefficients may be quantized to possibly reduce the amount of data used to represent the coefficients.

[0092] A CU generally includes one luminance component, denoted as Y, and two chrominance components, denoted as U and V. In other words, a given CU that is not further split into sub-CUs may include Y, U, and V components, each of which may be further partitioned into one or more PUs and TUs for purposes of prediction and transform of the CU, as previously described. For example, depending on the video sampling format, the size of the U and V components, in terms of a number of samples, may be the same as or different than the size of the Y component. As such, the techniques described above with reference to prediction, transform, and quantization may be performed for each of the Y, U, and V components of a given CU.

[0093] To encode a CU, one or more predictors for the CU are first derived based on one or more PUs of the CU. A predictor is a reference block that contains predicted data for the CU, and is derived on the basis of a corresponding PU for the CU, as previously described. For example, the PU indicates a partition of the CU for which predicted data is to be determined, and a prediction mode used to determine the predicted data. The predictor can be derived either through intra- (I) prediction (i.e., spatial prediction) or inter- (P or B) prediction (i.e., temporal prediction) modes. Hence, some CUs may be intra-coded (I) using spatial prediction with respect to neighboring reference blocks, or CUs, in the same frame, while other CUs may be inter-coded (P or B) with respect to reference blocks, or CUs, in other frames.

[0094] Upon identification of the one or more predictors based on the one or more PUs of the CU, a difference between the original video data of the CU corresponding to the

one or more PUs and the predicted data for the CU contained in the one or more predictors is calculated. This difference, also referred to as a prediction residual, comprises residual coefficients, and refers to pixel differences between portions of the CU specified by the one or more PUs and the one or more predictors, as previously described. The residual coefficients are generally arranged in a two-dimensional (2-D) array that corresponds to the one or more PUs of the CU.

[0095] To achieve further compression, the prediction residual is generally transformed, e.g., using a discrete cosine transform (DCT), integer transform, Karhunen-Loeve (K-L) transform, or another transform. The transform converts the prediction residual, i.e., the residual coefficients, in the spatial domain to residual transform coefficients in the transform domain, e.g., a frequency domain, as also previously described. In some occasions the transform is skipped, i.e., no transform is applied to the prediction residual. Transform skipped coefficients are also referred to as transform coefficients. The transform coefficients (including transform skip coefficients) are also generally arranged in a 2-D array that corresponds to the one or more TUs of the CU. For further compression, the residual transform coefficients may be quantized to possibly reduce the amount of data used to represent the coefficients, as also previously described.

[0096] To achieve still further compression, an entropy coder subsequently encodes the resulting residual transform coefficients, using Context Adaptive Variable Length Coding (CAVLC), Context Adaptive Binary Arithmetic Coding (CABAC), Probability Interval Partitioning Entropy Coding (PIPE), or another entropy coding methodology. Entropy coding may achieve this further compression by reducing or removing statistical redundancy inherent in the video data of the CU, represented by the coefficients, relative to other CUs.

[0097] A video sequence typically includes a series of video frames or pictures. A group of pictures (GOP) generally comprises a series of one or more of the video pictures. A GOP may include syntax data in a header of the GOP, a header of one or more of the pictures, or elsewhere, that describes a number of pictures included in the GOP. Each slice of a picture may include slice syntax data that describes an encoding mode for the respective slice. Video encoder 32 typically operates on video blocks within individual video slices in order to encode the video data. A video block may correspond to a coding node within a CU (e.g., a transform block of transform coefficients). The video blocks may have fixed or varying sizes, and may differ in size according to a specified coding standard.

[0098] As an example, the HM supports prediction in various PU sizes. Assuming that the size of a particular CU is $2N \times 2N$, the HM supports intra-prediction in PU sizes of $2N \times 2N$ or $N \times N$, and inter-prediction in symmetric PU sizes of $2N \times 2N$, $2N \times N$, $N \times 2N$, or $N \times N$. The HM also supports asymmetric partitioning for inter-prediction in PU sizes of $2N \times nU$, $2N \times nD$, $nL \times 2N$, and $nR \times 2N$. In asymmetric partitioning, one direction of a CU is not partitioned, while the other direction is partitioned into 25% and 75%. The portion of the CU corresponding to the 25% partition is indicated by an “n” followed by an indication of “Up”, “Down,” “Left,” or “Right.” Thus, for example, “ $2N \times nU$ ” refers to a $2N \times 2N$ CU that is partitioned horizontally with a $2N \times 0.5N$ PU on top and a $2N \times 1.5N$ PU on bottom.

[0099] In this disclosure, “ $N \times N$ ” and “N by N” may be used interchangeably to refer to the pixel dimensions of a video block in terms of vertical and horizontal dimensions, e.g., 16×16 pixels or 16 by 16 pixels. In general, a 16×16 block will have 16 pixels in a vertical direction ($y = 16$) and 16 pixels in a horizontal direction ($x = 16$). Likewise, an $N \times N$ block generally has N pixels in a vertical direction and N pixels in a horizontal direction, where N represents a nonnegative integer value. The pixels in a block may be arranged in rows and columns. Moreover, blocks need not necessarily have the same number of pixels in the horizontal direction as in the vertical direction. For example, blocks may comprise $N \times M$ pixels, where M is not necessarily equal to N.

[0100] Following intra-predictive or inter-predictive encoding using the PUs of a CU, video encoder 32 may calculate residual data for the TUs of the CU. The PUs may comprise pixel data in the spatial domain (also referred to as the pixel domain) and the TUs may comprise coefficients in the transform domain following application of a transform, e.g., a discrete cosine transform (DCT), an integer transform, a wavelet transform, skip transform, or a conceptually similar transform to residual video data. The residual data may correspond to pixel differences between pixels of the unencoded picture and prediction values corresponding to the PUs. Video encoder 32 may form the TUs including the residual data for the CU, and then transform the TUs to produce transform coefficients for the CU.

[0101] Following any transforms to produce transform coefficients, video encoder 32 may perform quantization of the transform coefficients. Quantization generally refers to a process in which transform coefficients are quantized to possibly reduce the amount of data used to represent the coefficients, providing further compression. The quantization process may reduce the bit depth associated with some or all of the coefficients. For

example, an n -bit value may be rounded down to an m -bit value during quantization, where n is greater than m .

[0102] In some examples, video encoder 32 may utilize a predefined scan order (e.g., horizontal, vertical, or diagonal) to scan the quantized transform coefficients to produce a serialized vector that can be entropy encoded. In some examples, video encoder 32 may perform an adaptive scan. After scanning the quantized transform coefficients to form a one-dimensional vector, video encoder 32 may entropy encode the one-dimensional vector, e.g., according to context adaptive variable length coding (CAVLC), context adaptive binary arithmetic coding (CABAC), syntax-based context-adaptive binary arithmetic coding (SBAC), Probability Interval Partitioning Entropy (PIPE) coding or another entropy encoding methodology. Video encoder 32 may also entropy encode syntax elements associated with the encoded video data for use by video decoder 42 in decoding the video data.

[0103] To perform CABAC, video encoder 32 may assign a context within a context model to a symbol to be transmitted. The context may relate to, for example, whether neighboring values of the symbol are non-zero or not. To perform CAVLC, video encoder 32 may select a variable length code for a symbol to be transmitted.

Codewords in VLC may be constructed such that relatively shorter codes correspond to more probable symbols, while longer codes correspond to less probable symbols. In this way, the use of VLC may achieve a bit savings over, for example, using equal-length codewords for each symbol to be transmitted. The probability determination may be based on a context assigned to the symbol.

[0104] Video decoder 42 may be configured to implement the reciprocal of the encoding techniques implemented by video encoder 32. For example, for the encoded significance syntax elements, video decoder 42 may decode the significance syntax elements by determining which contexts to use based on the determined scan order.

[0105] For instance, video encoder 32 signals syntax elements that indicate the values of the transform coefficients. Video encoder 32 generates these syntax elements in five passes, as one example, and using five passes is not necessary in every example. Video encoder 32 determines the location of the last significant coefficient and begins the first pass from the last significant coefficient. After the first pass, video encoder 32 implements the remaining four passes only on those transform coefficients remaining from the previous pass. In the first pass, video encoder 32 scans the transform coefficients using one of the scan orders illustrated in FIGS. 1A–1C and determines a

significance syntax element for each transform coefficient that indicates whether the value for the transform coefficient is zero or non-zero (i.e., insignificant or significant).

[0106] In the second pass, referred to as a greater than one pass, video encoder 32 generates syntax elements to indicate whether the absolute value of a significant coefficient is larger than one. In a similar manner, in the third pass, referred to as the greater than two pass, video encoder 32 generates syntax elements to indicate whether the absolute value of a greater than one coefficient is larger than two.

[0107] In the fourth pass, referred to as a sign pass, video encoder 32 generates syntax elements to indicate the sign information for significant coefficients. In the fifth pass, referred to as a coefficient level remaining pass, video encoder 32 generates syntax elements that indicate the remaining absolute value of a transform coefficient level (e.g., the remainder value). The remainder value may be coded as the absolute value of the coefficient minus 3. It should be noted that the five pass approach is just one example technique that may be used for coding transform coefficient and the techniques described herein may be equally applicable to other techniques.

[0108] In the techniques described in this disclosure, video encoder 32 encodes the significance syntax elements using context adaptive binary arithmetic coding (CABAC). In accordance with the techniques described in this disclosure, video encoder 32 may determine a scan order for the transform coefficients of the block, and determine contexts for the significance syntax elements of the transform coefficients of the block based on the determined scan order. Video encoder 32 may CABAC encode the significance syntax elements based on the determined contexts, and signal the encoded significance syntax elements in the coded bitstream.

[0109] Video decoder 42 may be configured to perform similar functions. For example, video decoder 42 receives from the coded bitstream significance syntax elements of transform coefficients of a block. Video decoder 42 may determine a scan order for the transform coefficients of the block (e.g., an order in which video encoder 32 scanned the transform coefficients). Video decoder 42 may then CABAC decode the significance syntax elements of the transform coefficients based at least one the determined contexts.

[0110] In some examples, video encoder 32 and video decoder 42 each determines the contexts that are the same if the determined scan order is a horizontal scan and if the determined scan order is a vertical scan, and determines the contexts, which are different than the contexts for the horizontal scan and vertical scan, if the determined scan order is a diagonal scan. In general, video encoder 32 and video decoder 42 may

each determine a first set of contexts for the significance syntax elements if the scan order is a first scan order, and determine a second set of contexts for the significance syntax elements if the scan order is a second scan order. The first set of contexts and the second set of contexts may be same in some cases (e.g., where the first scan order is a horizontal scan and the second scan order is a vertical scan, or vice-versa). The first set of contexts and the second set of contexts may be different in some cases (e.g., where the first scan order is either a horizontal or a vertical scan and the second scan order is not a horizontal or a vertical scan).

[0111] In some examples, video encoder 32 and video decoder 42 also determine a size of the block. In some of these examples, video encoder 32 and video decoder 42 determine the contexts for the significance syntax elements based on the determined scan order and based on the determined size of the block. For example, to determine the contexts, video encoder 32 and video decoder 42 may determine, based on the size of the block, that the contexts for the significance syntax elements of the transform coefficients that are the same for all scan orders. In other words, for certain sized blocks, video encoder 32 and video decoder 42 may determine contexts that are the same for all scan orders.

[0112] In some examples, the techniques described in this disclosure may build upon the concepts of sub-block horizontal and vertical scans, such as those described in: (1) Rosewarne, C., Maeda, M. "Non-CE11: Harmonisation of 8x8 TU residual scan" JCT-VC Contribution JCTVC-H0145; (2) Yu, Y., Panusopone, K., Lou, J., Wang, L. "Adaptive Scan for Large Blocks for HEVC; JCT-VC Contribution JCTVC-F569; and (3) U.S. Patent Application No. 13/551,458, filed July 17, 2012. For instance, the techniques described in this disclosure provide for improvement in the coding of significance syntax elements and harmonization across different scan orders and block (e.g., TU) sizes.

[0113] For example, as described above, a 4x4 block may be a sub-block of a larger block. In the techniques described in this disclosure, relatively large sized blocks (e.g., 16x16 or 32x32) may be divided into 4x4 sub-blocks, and video encoder 32 and video decoder 42 may be configured to determine the contexts for the 4x4 sub-blocks based on the scan order. In some examples, such techniques may be extendable to 8x8 sized blocks as well as for all scan orders (i.e., the 4x4 sub-blocks of the 8x8 block can be scanned horizontally, vertically, or diagonally). Such techniques may also allow for context sharing between the different scan orders.

[0114] In some examples, video encoder 32 and video decoder 42 determine contexts that are the same for all block sizes if the scan order is a diagonal scan (i.e., the contexts are shared for all of the TUs when using the diagonal scan). In this example, video encoder 32 and video decoder 42 may determine another set of contexts that are the same for the horizontal and vertical scan, which allows for context sharing depending on the scan order.

[0115] In some examples, there may be three sets of contexts: one for relatively large blocks, one for the diagonal scan of the 8x8 block or the 4x4 block, and one for both horizontal and vertical scans of the 8x8 block or the 4x4 block, where the contexts for the 8x8 block and the 4x4 block are different. Other combinations and permutations of the sizes and the scan orders may be possible, and video encoder 32 and video decoder 42 may be configured to determine contexts that are the same for these various combinations and permutations of sizes and scan orders.

[0116] FIG. 4 is a block diagram illustrating an example video encoder 32 that may implement the techniques described in this disclosure. In the example of FIG. 4, video encoder 32 includes a mode select unit 46, prediction processing unit 48, reference picture memory 70, summer 56, transform processing unit 58, quantization processing unit 60, and entropy encoding unit 62. Prediction processing unit 48 includes motion estimation unit 50, motion compensation unit 52, and intra prediction unit 54. For video block reconstruction, video encoder 32 also includes inverse quantization processing unit 64, inverse transform processing unit 66, and summer 68. A deblocking filter (not shown in FIG. 4) may also be included to filter block boundaries to remove blockiness artifacts from reconstructed video. If desired, the deblocking filter would typically filter the output of summer 68. Additional loop filters (in loop or post loop) may also be used in addition to the deblocking filter. It should be noted that prediction processing unit 48 and transform processing unit 58 should not be confused with PUs and TUs as described above.

[0117] As shown in FIG. 4, video encoder 32 receives video data, and mode select unit 46 partitions the data into video blocks. This partitioning may also include partitioning into slices, tiles, or other larger units, as well as video block partitioning, e.g., according to a quadtree structure of LCUs and CUs. Video encoder 32 generally illustrates the components that encode video blocks within a video slice to be encoded. A slice may be divided into multiple video blocks (and possibly into sets of video blocks referred to as tiles). Prediction processing unit 48 may select one of a plurality of possible coding

modes, such as one of a plurality of intra coding modes or one of a plurality of inter coding modes, for the current video block based on error results (e.g., coding rate and the level of distortion). Prediction processing unit 48 may provide the resulting intra- or inter-coded block to summer 56 to generate residual block data and to summer 68 to reconstruct the encoded block for use as a reference picture.

[0118] Intra prediction unit 54 within prediction processing unit 48 may perform intra-predictive coding of the current video block relative to one or more neighboring blocks in the same frame or slice as the current block to be coded to provide spatial compression. Motion estimation unit 50 and motion compensation unit 52 within prediction processing unit 48 perform inter-predictive coding of the current video block relative to one or more predictive blocks in one or more reference pictures to provide temporal compression.

[0119] Motion estimation unit 50 may be configured to determine the inter-prediction mode for a video slice according to a predetermined pattern for a video sequence. The predetermined pattern may designate video slices in the sequence as P slices or B slices. Motion estimation unit 50 and motion compensation unit 52 may be highly integrated, but are illustrated separately for conceptual purposes. Motion estimation, performed by motion estimation unit 50, is the process of generating motion vectors, which estimate motion for video blocks. A motion vector, for example, may indicate the displacement of a PU of a video block within a current video frame or picture relative to a predictive block within a reference picture.

[0120] A predictive block is a block that is found to closely match the PU of the video block to be coded in terms of pixel difference, which may be determined by sum of absolute difference (SAD), sum of square difference (SSD), or other difference metrics. In some examples, video encoder 32 may calculate values for sub-integer pixel positions of reference pictures stored in reference picture memory 70. For example, video encoder 32 may interpolate values of one-quarter pixel positions, one-eighth pixel positions, or other fractional pixel positions of the reference picture. Therefore, motion estimation unit 50 may perform a motion search relative to the full pixel positions and fractional pixel positions and output a motion vector with fractional pixel precision.

[0121] Motion estimation unit 50 calculates a motion vector for a PU of a video block in an inter-coded slice by comparing the position of the PU to the position of a predictive block of a reference picture. The reference picture may be selected from a first reference picture list (List 0) or a second reference picture list (List 1), each of

which identify one or more reference pictures stored in reference picture memory 70. Motion estimation unit 50 sends the calculated motion vector to entropy encoding unit 62 and motion compensation unit 52.

[0122] Motion compensation, performed by motion compensation unit 52, may involve fetching or generating the predictive block based on the motion vector determined by motion estimation, possibly performing interpolations to sub-pixel precision. Upon receiving the motion vector for the PU of the current video block, motion compensation unit 52 may locate the predictive block to which the motion vector points in one of the reference picture lists. Video encoder 32 forms a residual video block by subtracting pixel values of the predictive block from the pixel values of the current video block being coded, forming pixel difference values. The pixel difference values form residual data for the block, and may include both luma and chroma difference components. Summer 56 represents the component or components that perform this subtraction operation. Motion compensation unit 52 may also generate syntax elements associated with the video blocks and the video slice for use by video decoder 42 in decoding the video blocks of the video slice.

[0123] Intra-prediction unit 54 may intra-predict a current block, as an alternative to the inter-prediction performed by motion estimation unit 50 and motion compensation unit 52, as described above. In particular, intra-prediction unit 54 may determine an intra-prediction mode to use to encode a current block. In some examples, intra-prediction unit 54 may encode a current block using various intra-prediction modes, e.g., during separate encoding passes, and intra-prediction unit 54 (or mode select unit 46, in some examples) may select an appropriate intra-prediction mode to use from the tested modes. For example, intra-prediction unit 54 may calculate rate-distortion values using a rate-distortion analysis for the various tested intra-prediction modes, and select the intra-prediction mode having the best rate-distortion characteristics among the tested modes. Rate-distortion analysis generally determines an amount of distortion (or error) between an encoded block and an original, unencoded block that was encoded to produce the encoded block, as well as a bit rate (that is, a number of bits) used to produce the encoded block. Intra-prediction unit 54 may calculate ratios from the distortions and rates for the various encoded blocks to determine which intra-prediction mode exhibits the best rate-distortion value for the block.

[0124] In any case, after selecting an intra-prediction mode for a block, intra-prediction unit 54 may provide information indicative of the selected intra-prediction mode for the

block to entropy encoding unit 62. Entropy encoding unit 62 may encode the information indicating the selected intra-prediction mode in accordance with the entropy techniques described herein.

[0125] After prediction processing unit 48 generates the predictive block for the current video block via either inter-prediction or intra-prediction, video encoder 32 forms a residual video block by subtracting the predictive block from the current video block. The residual video data in the residual block may be included in one or more TBs and applied to transform processing unit 58. Transform processing unit 58 may transform the residual video data into residual transform coefficients using a transform, such as a discrete cosine transform (DCT) or a conceptually similar transform. Transform processing unit 58 may convert the residual video data from a pixel domain to a transform domain, such as a frequency domain. In some cases, transform processing unit 58 may apply a 2-dimensional (2-D) transform (in both the horizontal and vertical direction) to the residual data in the TBs. In some examples, transform processing unit 58 may instead apply a horizontal 1-D transform, a vertical 1-D transform, or no transform to the residual data in each of the TBs.

[0126] Transform processing unit 58 may send the resulting transform coefficients to quantization processing unit 60. Quantization processing unit 60 quantizes the transform coefficients to further reduce the bit rate. The quantization process may reduce the bit depth associated with some or all of the coefficients. The degree of quantization may be modified by adjusting a quantization parameter. In some examples, quantization processing unit 60 may then perform a scan of the matrix including the quantized transform coefficients. Alternatively, entropy encoding unit 62 may perform the scan.

[0127] As described above, the scan performed on a transform block may be based on the size of the transform block. Quantization processing unit 60 and/or entropy encoding unit 62 may scan 8x8, 16x16, and 32x32 transform blocks using any combination of the sub-block scans described above with respect to FIGS. 1A–1C. When more one than one scan is available for a transform block, entropy encoding unit 62 may determine a scan order based on a coding parameter associated with the transform block, such as a prediction mode associated with a prediction unit corresponding to the transform block. Further details with respect to entropy encoding unit 62 are described below with respect to FIG. 5.

[0128] Inverse quantization processing unit 64 and inverse transform processing unit 66 apply inverse quantization and inverse transformation, respectively, to reconstruct the residual block in the pixel domain for later use as a reference block of a reference picture. Motion compensation unit 52 may calculate a reference block by adding the residual block to a predictive block of one of the reference pictures within one of the reference picture lists. Motion compensation unit 52 may also apply one or more interpolation filters to the reconstructed residual block to calculate sub-integer pixel values for use in motion estimation. Summer 68 adds the reconstructed residual block to the motion compensated prediction block produced by motion compensation unit 52 to produce a reference block for storage in reference picture memory 70. The reference block may be used by motion estimation unit 50 and motion compensation unit 52 as a reference block to inter-predict a block in a subsequent video frame or picture.

[0129] Following quantization, entropy encoding unit 62 entropy encodes the quantized transform coefficients. For example, entropy encoding unit 62 may perform context adaptive variable length coding (CAVLC), context adaptive binary arithmetic coding (CABAC), syntax-based context-adaptive binary arithmetic coding (SBAC), probability interval partitioning entropy (PIPE) coding or another entropy encoding methodology or technique. Following the entropy encoding by entropy encoding unit 62, the encoded bitstream may be transmitted to video decoder 42, or archived for later transmission or retrieval by video decoder 42. Entropy encoding unit 62 may also entropy encode the motion vectors and the other syntax elements for the current video slice being coded. Entropy encoding unit 62 may entropy encode syntax elements such as the significance syntax elements and the other syntax elements for the transform coefficients described above using CABAC.

[0130] In some examples, entropy encoding unit 62 may be configured to implement the techniques described in this disclosure of determining contexts based on a determined scan order. In some examples, entropy encoding unit 62 in conjunction with one or more units within video encoder 32 may be configured to implement the techniques described in this disclosure. In some examples, a processor or processing unit (not shown) of video encoder 32 may be configured to implement the techniques described in this disclosure.

[0131] FIG. 5 is a block diagram that illustrates an example entropy encoding unit 62 that may implement the techniques described in this disclosure. The entropy encoding unit 62 illustrated in FIG. 5 may be a CABAC encoder. The example entropy encoding

unit 62 may include a binarization unit 72, an arithmetic encoding unit 80, which includes a bypass encoding engine 74 and a regular encoding engine 78, and a context modeling unit 76.

[0132] Entropy encoding unit 62 may receive one or more syntax elements, such as the significance syntax element, referred to as a `significant_coefficient_flag` in HEVC, the greater than 1 flag, referred to as a `coeff_abs_level_greater1_flag` in HEVC, the greater than 2 flag, referred to as `coeff_abs_level_greater2_flag` in HEVC, the sign flag referred to as `coeff_sign_flag` in HEVC, and the level syntax element referred to as `coeff_abs_level_remain`. Binarization unit 72 receives a syntax element and produces a bin string (i.e., binary string). Binarization unit 72 may use, for example, any one or combination of the following techniques to produce a bin string: fixed length coding, unary coding, truncated unary coding, truncated Rice coding, Golomb coding, exponential Golomb coding, and Golomb-Rice coding. Further, in some cases, binarization unit 72 may receive a syntax element as a binary string and simply pass-through the bin values. In one example, binarization unit 72 receives the significance syntax element and produces a bin string.

[0133] Arithmetic encoding unit 80 is configured to receive a bin string from binarization unit 72 and perform arithmetic encoding on the bin string. As shown in FIG. 5, arithmetic encoding unit 80 may receive bin values from a bypass path or the regular coding path. Bin values that follow the bypass path may be bins values identified as bypass coded and bin values that follow the regular encoding path may be identified as CABAC-coded. Consistent with the CABAC process described above, in the case where arithmetic encoding unit 80 receives bin values from a bypass path, bypass encoding engine 74 may perform arithmetic encoding on bin values without utilizing an adaptive context assigned to a bin value. In one example, bypass encoding engine 74 may assume equal probabilities for possible values of a bin.

[0134] In the case where arithmetic encoding unit 80 receives bin values through the regular path, context modeling unit 76 may provide a context variable (e.g., a context state), such that regular encoding engine 78 may perform arithmetic encoding based on the context assignments provided by context modeling unit 76. The context assignments may be defined according to a video coding standard, such as the HEVC standard. Further, in one example context modeling unit 76 and/or entropy encoding unit 62 may be configured to determine contexts for bins of the significance syntax elements based on techniques described herein. The techniques may be incorporated

into HEVC or another video coding standard. The context models may be stored in memory. Context modeling unit 76 may include a series of indexed tables and/or utilize mapping functions to determine a context and a context variable for a particular bin. After encoding a bin value, regular encoding engine 78 may update a context based on the actual bin values.

[0135] FIG. 6 is a flowchart illustrating an example process for encoding video data according to this disclosure. Although the process in FIG. 6 is described below as generally being performed by video encoder 32, the process may be performed by any combination of video encoder 32, entropy encoding unit 62, and/or context modeling unit 76.

[0136] As illustrated, video encoder 32 may determine a scan order for transform coefficients of a block (82). Video encoder 32 may determine contexts for the transform coefficients based on the scan order (84). In some examples, video encoder 32 determines the contexts based on the determined scan order, positions of the transform coefficients with the block, and a size of the block. For example, for a particular block size (e.g., an 8x8 block of transform coefficients) and a particular position (e.g., transform coefficient position), video encoder 32 may determine the same context if the scan order is either horizontal scan or vertical scan, and determine a different context if the scan order is not the horizontal scan or the vertical scan.

[0137] Video encoder 32 may CABAC encode significance syntax elements (e.g., significance flags) for the transform coefficients based on the determined contexts (86). Video encoder 32 may signal the encoded significance syntax elements (e.g., significance flags) (88).

[0138] FIG. 7 is a block diagram illustrating an example video decoder 42 that may implement the techniques described in this disclosure. In the example of FIG. 7, video decoder 42 includes an entropy decoding unit 90, prediction processing unit 92, inverse quantization processing unit 98, inverse transform processing unit 100, summer 102, and reference picture memory 104. Prediction processing unit 92 includes motion compensation unit 94 and intra prediction unit 96. Video decoder 42 may, in some examples, perform a decoding pass generally reciprocal to the encoding pass described with respect to video encoder 32 from FIG. 4.

[0139] During the decoding process, video decoder 42 receives an encoded video bitstream that represents video blocks of an encoded video slice and associated syntax elements from video encoder 32. Entropy decoding unit 90 of video decoder 42 entropy

decodes the bitstream to generate quantized coefficients, motion vectors, and other syntax elements. Entropy decoding unit 90 forwards the motion vectors and other syntax elements to prediction processing unit 92. Video decoder 42 may receive the syntax elements at the video slice level and/or the video block level.

[0140] In some examples, entropy decoding unit 90 may be configured to implement the techniques described in this disclosure of determining contexts based on a determined scan order. In some examples, entropy decoding unit 90 in conjunction with one or more units within video decoder 42 may be configured to implement the techniques described in this disclosure. In some examples, a processor or processing unit (not shown) of video decoder 42 may be configured to implement the techniques described in this disclosure.

[0141] FIG. 8 is a block diagram that illustrates an example entropy decoding unit 90 that may implement the techniques described in this disclosure. Entropy decoding unit 90 receives an entropy encoded bitstream and decodes syntax elements from the bitstream. Syntax elements may include the syntax elements such as `significant_coefficient_flag`, `coeff_abs_level_remain`, `coeff_abs_level_greater1_flag`, `coeff_abs_level_greater2_flag`, and `coeff_sign_flag`, syntax elements described above for transform coefficients of a block. The example entropy decoding unit 90 in FIG. 8 includes an arithmetic decoding unit 106, which may include a bypass decoding engine 108 and a regular decoding engine 110. The example entropy decoding unit 90 also includes context modeling unit 112 and inverse binarization unit 114. The example entropy decoding unit 90 may perform the reciprocal functions of the example entropy encoding unit 62 described with respect to FIG. 5. In this manner, entropy decoding unit 90 may perform entropy decoding based on the techniques described in this disclosure.

[0142] Arithmetic decoding unit 106 receives an encoded bit stream. As shown in FIG. 8, arithmetic decoding unit 106 may process encoded bin values according to a bypass path or the regular coding path. An indication whether an encoded bin value should be processed according to a bypass path or a regular pass may be signaled in the bitstream with higher level syntax. Consistent with the CABAC process described above, in the case where arithmetic decoding unit 106 receives bin values from a bypass path, bypass decoding engine 108 may perform arithmetic encoding on bin values without utilizing a context assigned to a bin value. In one example, bypass decoding engine 108 may assume equal probabilities for possible values of a bin.

[0143] In the case where arithmetic decoding unit 106 receives bin values through the regular path, context modeling unit 112 may provide a context variable, such that regular decoding engine 110 may perform arithmetic encoding based on the context assignments provided by context modeling unit 112. The context assignments may be defined according to a video coding standard, such as HEVC. The context models may be stored in memory. Context modeling unit 112 may include a series of indexed tables and/or utilize mapping functions to determine a context and a context variable portion of an encoded bitstream. Further, in one example context modeling unit 112 and/or entropy decoding unit 90 may be configured to assign contexts to bins of the significance syntax elements based on techniques described herein. After decoding a bin value, regular decoding engine 110, may update a context based on the decoded bin values. Further, inverse binarization unit 114 may perform an inverse binarization on a bin value and use a bin matching function to determine if a bin value is valid. The inverse binarization unit 114 may also update the context modeling unit based on the matching determination. Thus, the inverse binarization unit 114 outputs syntax elements according to a context adaptive decoding technique.

[0144] Referring back to FIG. 7, when the video slice is coded as an intra-coded (I) slice, intra prediction unit 96 of prediction processing unit 92 may generate prediction data for a video block of the current video slice based on a signaled intra prediction mode and data from previously decoded blocks of the current frame or picture. When the video frame is coded as an inter-coded (i.e., B or P) slice, motion compensation unit 94 of prediction processing unit 92 produces predictive blocks for a video block of the current video slice based on the motion vectors and other syntax elements received from entropy decoding unit 90. The predictive blocks may be produced from one of the reference pictures within one of the reference picture lists. Video decoder 42 may construct the reference picture lists, List 0 and List 1, using default construction techniques based on reference pictures stored in reference picture memory 104.

[0145] Motion compensation unit 94 determines prediction information for a video block of the current video slice by parsing the motion vectors and other syntax elements, and uses the prediction information to produce the predictive blocks for the current video block being decoded. For example, motion compensation unit 94 uses some of the received syntax elements to determine a prediction mode (e.g., intra- or inter-prediction) used to code the video blocks of the video slice, an inter-prediction slice type (e.g., B slice or P slice), construction information for one or more of the reference

picture lists for the slice, motion vectors for each inter-encoded video block of the slice, inter-prediction status for each inter-coded video block of the slice, and other information to decode the video blocks in the current video slice.

[0146] Motion compensation unit 94 may also perform interpolation based on interpolation filters. Motion compensation unit 94 may use interpolation filters as used by video encoder 32 during encoding of the video blocks to calculate interpolated values for sub-integer pixels of reference blocks. In this case, motion compensation unit 94 may determine the interpolation filters used by video encoder 32 from the received syntax elements and use the interpolation filters to produce predictive blocks.

[0147] Inverse quantization processing unit 98 inverse quantizes, i.e., de-quantizes, the quantized transform coefficients provided in the bitstream and decoded by entropy decoding unit 90. The inverse quantization process may include use of a quantization parameter calculated by video encoder 32 for each video block in the video slice to determine a degree of quantization and, likewise, a degree of inverse quantization that should be applied. Inverse transform processing unit 100 applies an inverse transform, e.g., an inverse DCT, an inverse integer transform, or a conceptually similar inverse transform process, to the transform coefficients in order to produce residual blocks in the pixel domain.

[0148] In some cases, inverse transform processing unit 100 may apply a 2-dimensional (2-D) inverse transform (in both the horizontal and vertical direction) to the coefficients. In some examples, inverse transform processing unit 88 may instead apply a horizontal 1-D inverse transform, a vertical 1-D inverse transform, or no transform to the residual data in each of the TUs. The type of transform applied to the residual data at video encoder 32 may be signaled to video decoder 42 to apply an appropriate type of inverse transform to the transform coefficients.

[0149] After motion compensation unit 94 generates the predictive block for the current video block based on the motion vectors and other syntax elements, video decoder 42 forms a decoded video block by summing the residual blocks from inverse transform processing unit 100 with the corresponding predictive blocks generated by motion compensation unit 94. Summer 102 represents the component or components that perform this summation operation. If desired, a deblocking filter may also be applied to filter the decoded blocks in order to remove blockiness artifacts. Other loop filters (either in the coding loop or after the coding loop) may also be used to smooth pixel transitions, or otherwise improve the video quality. The decoded video blocks in a

given frame or picture are then stored in reference picture memory 104, which stores reference pictures used for subsequent motion compensation. Reference picture memory 104 also stores decoded video for later presentation on a display device, such as display device 44 of FIG. 3.

[0150] FIG. 9 is a flowchart illustrating an example process for decoding video data according to this disclosure. Although the process in FIG. 9 is described below as generally being performed by video decoder 42, the process may be performed by any combination of video decoder 42, entropy decoding unit 90, and/or context modeling unit 112.

[0151] As illustrated in FIG. 9, video decoder 42 receives, from a coded bitstream, significance syntax elements (e.g., significance flags) for transform coefficients of a block (116). Video decoder 42 determines a scan order for the transform coefficients (118). Video decoder 42 determines contexts for the transform coefficients based on the determined scan order (120). In some examples, video decoder 42 also determines the block size and determines the contexts based on the determined scan order and block size. In some examples, video decoder 42 determines the contexts based on the determined scan order, positions of the transform coefficients with the block, and a size of the block. For example, for a particular block size (e.g., an 8x8 block of transform coefficients) and a particular position (e.g., transform coefficient position), video decoder 42 may determine the same context if the scan order is either horizontal scan or vertical scan, and determine a different context if the scan order is not the horizontal scan or the vertical scan. Video decoder 42 CABAC decodes the significance syntax elements (e.g., significance flags) based on the determined contexts (122).

[0152] Video encoder 32, as described in the flowchart of FIG. 6, and video decoder 42, as described in the flowchart of FIG. 9, may be configured to implement various other example techniques described in this disclosure. For example, to determine the contexts, video encoder 32 and video decoder 42 may be configured to determine the contexts that are the same if the determined scan order is a horizontal scan and if the determined scan order is a vertical scan, and determine the contexts, which are different than the contexts if the determined scan order is the horizontal scan and if the determined scan order is the vertical scan, if the determined scan order is not the horizontal scan or the vertical scan (e.g., diagonal scan).

[0153] In some examples, to determine the contexts, video encoder 32 and video decoder 42 may be configured to determine a first set of contexts for the significance

syntax elements if the scan order is a first scan order, and determine a second set of contexts for the significance syntax elements if the scan order is a second scan order. In some these examples, the first set of contexts is the same as the second set of contexts if the first scan order is a horizontal scan and the second scan order is a vertical scan. In some of these examples, the first set of contexts is different than the second set of contexts if the first scan order is one of a horizontal scan or a vertical scan and the second scan order is not the horizontal scan or the vertical scan.

[0154] In some examples, video encoder 32 and video decoder 42 may determine a size of the block. In some of these examples, video encoder 32 and video decoder 42 may determine the contexts based on the scan order and the determined size of the block. As one example, video encoder 32 and video decoder 42 may determine, based on the determined size of the block, the contexts for the significance syntax elements of the transform coefficients that are the same for all scan orders (i.e., for some block sizes, the contexts are the same for all scan orders).

[0155] For example, video encoder 32 and video decoder 42 may determine whether the size of the block is a first size or a second size. One example of the first size is the 4x4 block, and one example of the second size is the 8x8 block. If the size of the block is the first size (e.g., the 4x4 block), video encoder 32 and video decoder 42 may determine the contexts that are the same for all scan orders (e.g., the contexts that are the same for the diagonal, horizontal, and vertical scans for the 4x4 block). If the size of the block is the second size (e.g., the 8x8 block), video encoder 32 and video decoder 42 may determine the contexts that are different for at least two different scan orders (e.g., the contexts for the diagonal scan of the 8x8 block is different than the contexts for the horizontal or vertical scan of the 8x8 block, but the contexts for the horizontal and vertical scan of the 8x8 block may be the same).

[0156] The following describes various additional techniques for improving the manner in which transform coefficients are coded, such as transform coefficients resulting from intra-coding, as one example. However, the techniques may be applicable to other examples as well, such as for inter-coding. The following techniques can be used individually or in conjunction with any of the other techniques described in this disclosure. Moreover, the techniques described above may be used in conjunction with any of the following techniques, or may be implemented separately from any of the following techniques.

[0157] In some examples, video encoder 32 and video decoder 42 may utilize one scan order to determine the location of last significant coefficient. Video encoder 32 and video decoder 42 may utilize a different scan order to determine neighborhood contexts for the transform coefficients. Video encoder 32 and video decoder 42 may then code significance flags, level information, and sign information based on the determined neighborhood contexts. For example, video encoder 32 and video decoder 42 may utilize a horizontal or vertical scan (referred to as the nominal scan) to identify the last significant transform coefficient, and then utilize a diagonal scan on the 4x4 blocks or 4x4 sub-blocks (if 8x8 block) to determine the neighborhood contexts.

[0158] In some examples, for 16x16 and 32x32 blocks, a neighborhood (in the transform domain) of the current coefficient being processed is used for derivation of the context used to code the significance flag for the coefficient. Similarly, in JCTVC-H0228, a neighborhood is used for coding significance as well as level information for all block sizes. Using neighborhood-based contexts for 4x4 and 8x8 blocks may improve the coding efficiency of HEVC. But if the existing significance neighborhoods for significance maps from some other techniques are used with horizontal or vertical scans, the ability to derive contexts in parallel may be affected. Hence, in some examples, a scheme is described which uses certain aspects of horizontal and vertical scans with the neighborhood used for significance coding from some other techniques.

[0159] This is accomplished as follows. In some examples, first the position of the last significant coefficient in the scan order is coded in the bit-stream. This is followed by the significance map for a subset of 16 coefficients (a 4x4 sub-block in case of a 4x4 sub-block based diagonal scan) in backwards scan order, followed by coding passes for level information and sign. It should be noted that the position of the last significant coefficient depends directly on the specific scan that is used. An example of this is shown in FIG. 10.

[0160] FIG. 10 is a conceptual diagram illustrating positions of a last significant coefficient depending on the scan order. FIG. 10 illustrates block 124. The pixels shown with solid circles are significant. For a horizontal scan, the position of the last significant position is (1, 2) in (row, column) format (transform coefficient 128). For a 4x4 subblock based diagonal scan (up-right), the position of the last significant position is (0, 3) (transform coefficient 126).

[0161] In this example, for horizontal or vertical scans, the last significant coefficient position is still determined and coded based on the nominal scan. But then, for coding

significance, level and sign information, the block is scanned using a 4x4 sub-block based diagonal scan starting with the bottom-right coefficient and proceeding backwards to the DC coefficient. If it can be derived from the position of the last significant coefficient that a particular coefficient is not significant, no significance, level or sign information is coded for that coefficient.

[0162] Example of this approach is shown in FIG. 11 for a horizontal scan. FIG. 11 is a conceptual diagram illustrating use of a diagonal scan in place of an original horizontal scan. FIG. 11 illustrates block 130. The coefficients with solid fill are significant. The position of the last significant position, assuming a horizontal scan, is (1, 1) (transform coefficient 132). All coefficients with row indices greater than 1 can be inferred to be not significant. Similarly, all coefficients with row index 1 and column index greater than 1 can be inferred to be not significant. Similarly, the coefficient (1, 1) can be inferred to be significant. Its level and sign information cannot be inferred. For coding of significance, level and sign information, a backward 4x4 sub-block based diagonal scan is used. Starting with the bottom right coefficient, the significance flags are encoded. The significance flags that can be inferred are not explicitly coded. A neighborhood based context is used for coding of significance flags. The neighborhood may be the same as that used for 16x16 and 32x32 blocks or a different neighborhood may be used. It should be noted that, similar to above, separate sets of neighborhood-based contexts may be used for the different scans (horizontal, vertical, and 4x4 sub-block). Also, the contexts may be shared between different block sizes.

[0163] In another example, any of a various techniques, such as those of JCTVC-H0228, may be used for coding significance, level and sign information for 4x4 and 8x8 blocks after the position of the last significant position is coded assuming the nominal scan. For coding of significance, level and sign information, a 4x4 sub-block based diagonal scan may be used.

[0164] It should be noted that the method is not restricted to horizontal, vertical and 4x4 sub-block based diagonal scans. The basic principle is to send the last significant coefficient position assuming the nominal scan and then code the significance (and possibly level and sign) information using another scan which uses neighborhood based contexts. Similarly, although the techniques have been described for 4x4 and 8x8 blocks, it can be extended to any block size where horizontal and/or vertical scans may be used.

[0165] In one example, rather than utilizing separate contexts for each transform coefficient based on its position in the transform block, the video coder (e.g., video encoder 32 or video decoder 42) may determine which context to use for coding a transform coefficient based on row index or the column index of the transform coefficient. For example, for a horizontal scan, all transform coefficients in the same row may share the same context, and the video coder may utilize different contexts for transform coefficients in the different rows. For a vertical scan, all transform coefficients in the same column may share the same context, and the video coder may utilize different contexts for transform coefficients in the different columns.

[0166] Some other techniques may use multiple context sets based on coefficient position for coding of significance maps for block sizes of 16x16 and higher. Similarly, JCTVC-H0228 (and also HM5.0) uses the sum of row and column indices to determine the context set. In the case of JCTVC-H0228, this is done even for horizontal and vertical scans.

[0167] In some example techniques of this disclosure, the context set used to code the significance or level for a particular coefficient for horizontal scan may depend only on the row index of the coefficient. Similarly, the context set to code the significance or level for a coefficient in case of vertical scan may depend only on the column index of the coefficient.

[0168] In some example techniques of this disclosure, the context set may depend only on the absolute index of the coefficient in the scan. Different scans may use different functions to derive the context set.

[0169] Furthermore, as described above, horizontal, vertical and 4x4 sub-block-based diagonal scans may use separate context sets or the horizontal and vertical scans may share context sets. In some examples, not only the context set but also the context itself depends only on the absolute index of the coefficient in the scanning order.

[0170] In some examples, the video coder (e.g., video encoder 32 or video decoder 42) may be configured to implement only one type of scan (e.g., a diagonal scan).

However, the neighboring regions that the video coder evaluates may be based on the nominal scan. The nominal scan is the scan the video coder would have performed had the video coder been able to perform other scans. For instance, video encoder 32 may signal that the horizontal scan is to be used. However, video decoder 42 may implement the diagonal scan instead, but the neighboring regions that the video coder evaluates

may be based on the signaling that the horizontal scan is to be used. The same would apply for the vertical scan.

[0171] In some examples, if the nominal scan is the horizontal scan, then the video coder may stretch the neighboring region that is evaluated in the horizontal direction relative to the regions that are currently used. The same would apply when the nominal scan is the vertical scan, but in the vertical direction. The stretching of the neighboring region may be referred to as varying the region. For example, if the nominal scan is horizontal, then rather than evaluating a transform coefficient that is two rows down from where the current transform coefficient being coded is located, the video coder may evaluate the transform coefficient that is three columns apart from where the current transform coefficient is located. The same would apply when the nominal scan is the vertical scan, but the transform coefficient would be located three rows apart from where the current transform coefficient (e.g., the one being coded) is located

[0172] FIG. 12 is a conceptual diagram illustrating a context neighborhood for a nominal horizontal scan. FIG. 12 illustrates 8x8 block 134 that includes 4x4 sub-blocks 136A–136D. Compared to the context neighborhood in some other techniques, the coefficient two rows down has been replaced by the coefficient that is in the same row but three columns apart (X_4). Similarly, if the nominal scan is vertical, a context neighborhood that is stretched in the vertical direction may be used.

[0173] In one or more examples, the functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on or transmitted over, as one or more instructions or code, a computer-readable medium and executed by a hardware-based processing unit.

Computer-readable media may include computer-readable storage media, which corresponds to a tangible medium such as data storage media, or communication media including any medium that facilitates transfer of a computer program from one place to another, e.g., according to a communication protocol. In this manner, computer-readable media generally may correspond to (1) tangible computer-readable storage media which is non-transitory or (2) a communication medium such as a signal or carrier wave. Data storage media may be any available media that can be accessed by one or more computers or one or more processors to retrieve instructions, code and/or data structures for implementation of the techniques described in this disclosure. A computer program product may include a computer-readable medium.

[0174] By way of example, and not limitation, such computer-readable storage media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage, or other magnetic storage devices, flash memory, or any other medium that can be used to store desired program code in the form of instructions or data structures and that can be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if instructions are transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. It should be understood, however, that computer-readable storage media and data storage media do not include connections, carrier waves, signals, or other transient media, but are instead directed to non-transient, tangible storage media. Disk and disc, as used herein, includes compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk and Blu-ray disc, where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media.

[0175] Instructions may be executed by one or more processors, such as one or more digital signal processors (DSPs), general purpose microprocessors, application specific integrated circuits (ASICs), field programmable logic arrays (FPGAs), or other equivalent integrated or discrete logic circuitry. Accordingly, the term “processor,” as used herein may refer to any of the foregoing structure or any other structure suitable for implementation of the techniques described herein. In addition, in some aspects, the functionality described herein may be provided within dedicated hardware and/or software modules configured for encoding and decoding, or incorporated in a combined codec. Also, the techniques could be fully implemented in one or more circuits or logic elements.

[0176] The techniques of this disclosure may be implemented in a wide variety of devices or apparatuses, including a wireless handset, an integrated circuit (IC) or a set of ICs (e.g., a chip set). Various components, modules, or units are described in this disclosure to emphasize functional aspects of devices configured to perform the disclosed techniques, but do not necessarily require realization by different hardware units. Rather, as described above, various units may be combined in a codec hardware

unit or provided by a collection of interoperative hardware units, including one or more processors as described above, in conjunction with suitable software and/or firmware.

[0177] Various examples have been described. These and other examples are within the scope of the following claims.

WHAT IS CLAIMED IS:

1. A method for decoding video data, the method comprising:
receiving, from a coded bitstream, significance flags of transform coefficients of a block;
determining a scan order for the transform coefficients of the block;
determining contexts for the significance flags of the transform coefficients of the block based on the determined scan order; and
context adaptive binary arithmetic coding (CABAC) decoding the significance flags of the transform coefficients based at least on the determined contexts.
2. The method of claim 1, wherein determining the contexts comprises determining the contexts based on size of the block, positions of the transform coefficients within the block, and the scan order.
3. The method of claim 1, wherein determining the contexts comprises:
determining the contexts that are the same if the determined scan order is a horizontal scan and if the determined scan order is a vertical scan; and
determining the contexts, which are different than the contexts if the determined scan order is the horizontal scan and if the determined scan order is the vertical scan, if the determined scan order is not the horizontal scan or the vertical scan.
4. The method of claim 1, wherein determining contexts for the significance flags of the transform coefficients of the block based on the determined scan order comprises determining the same contexts if the scan order is horizontal scan order or vertical scan order.
5. The method of claim 1, wherein determining the contexts comprises:
determining a first set of contexts for the significance flags if the scan order is a first scan order; and
determining a second set of contexts for the significance flags if the scan order is a second scan order.

6. The method of claim 5, wherein the first set of contexts is the same as the second set of contexts if the first scan order is a horizontal scan and the second scan order is a vertical scan.
7. The method of claim 5, wherein the first set of context is different than the second set of contexts if the first scan order is one of a horizontal scan or a vertical scan and the second scan order is not the horizontal scan or the vertical scan.
8. The method of claim 1, wherein determining the contexts comprises determining the contexts for the significance flags of the transform coefficients of the block based on the determined scan order and based on size of the block.
9. The method of claim 1, further comprising:
 - determining whether size of the block is a first size or a second size,
 - wherein, if the size of the block is the first size, determining the contexts comprises determining the contexts that are the same for all scan orders, and
 - wherein, if the size of the block is the second size, determining the contexts comprises determining the contexts that are different for at least two different scan orders.
10. The method of claim 1, wherein the block comprises an 8x8 block of transform coefficients.
11. A method for encoding video data, the method comprising:
 - determining a scan order for transform coefficients of a block;
 - determining contexts for significance flags of the transform coefficients of the block based on the determined scan order;
 - context adaptive binary arithmetic coding (CABAC) encoding the significance flags of the transform coefficients based at least on the determined contexts; and
 - signaling the encoded significance flags in a coded bitstream.
12. The method of claim 11, wherein determining the contexts comprises determining the contexts based on size of the block, positions of the transform coefficients within the block, and the scan order.

13. The method of claim 11, wherein determining the contexts comprises:
determining the contexts that are the same if the determined scan order is a horizontal scan and if the determined scan order is a vertical scan; and
determining the contexts, which are different than the contexts if the determined scan order is the horizontal scan and if the determined scan order is the vertical scan, if the determined scan order is not the horizontal scan or the vertical scan.
14. The method of claim 11, wherein determining contexts for the significance flags of the transform coefficients of the block based on the determined scan order comprises determining the same contexts if the scan order is horizontal scan order or vertical scan order.
15. The method of claim 11, wherein determining the contexts comprises:
determining a first set of contexts for the significance flags if the scan order is a first scan order; and
determining a second set of contexts for the significance flags if the scan order is a second scan order.
16. The method of claim 15, wherein the first set of contexts is the same as the second set of contexts if the first scan order is a horizontal scan and the second scan order is a vertical scan.
17. The method of claim 15, wherein the first set of context is different than the second set of contexts if the first scan order is one of a horizontal scan or a vertical scan and the second scan order is not the horizontal scan or the vertical scan.
18. The method of claim 11, wherein determining the contexts comprises determining the contexts for the significance flags of the transform coefficients of the block based on the determined scan order and based on size of the block.
19. The method of claim 11, wherein the block comprises an 8x8 block of transform coefficients.

20. An apparatus for coding video data, the apparatus comprising a video coder configured to:

- determine a scan order for transform coefficients of a block;
- determine contexts for significance flags of the transform coefficients of the block based on the determined scan order; and
- context adaptive binary arithmetic coding (CABAC) code the significance flags of the transform coefficients based at least on the determined contexts.

21. The apparatus of claim 20, wherein the video coder comprises a video decoder, and wherein the video decoder is configured to:

- receive, from a coded bitstream, the significance flags of the transform coefficients of the block; and
- CABAC decode the significance flags of the transform coefficients based on the determined contexts.

22. The apparatus of claim 20, wherein the video coder comprises a video encoder, and wherein the video encoder is configured to:

- CABAC encode the significance flags of the transform coefficients based on the determined contexts; and
- signal, in a coded bitstream, the significance flags of the transform coefficients.

23. The apparatus of claim 20, wherein, to determine the contexts, the video coder is configured to determine the contexts based on size of the block, positions of the transform coefficients within the block, and the scan order.

24. The apparatus of claim 20, wherein, to determine the contexts, the video coder is configured to:

- determine the contexts that are the same if the determined scan order is a horizontal scan and if the determined scan order is a vertical scan; and
- determine the contexts, which are different than the contexts if the determined scan order is the horizontal scan and if the determined scan order is the vertical scan, if the determined scan order is not the horizontal scan or the vertical scan.

25. The apparatus of claim 20, wherein, to determine contexts for the significance flags of the transform coefficients of the block based on the determined scan order, the video coder is configured to determine the same contexts if the scan order is horizontal scan order or vertical scan order.

26. The apparatus of claim 20, wherein, to determine the contexts, the video coder is configured to:

determine a first set of contexts for the significance flags if the scan order is a first scan order; and

determine a second set of contexts for the significance flags if the scan order is a second scan order.

27. The apparatus of claim 26, wherein the first set of contexts is the same as the second set of contexts if the first scan order is a horizontal scan and the second scan order is a vertical scan.

28. The apparatus of claim 26, wherein the first set of context is different than the second set of contexts if the first scan order is one of a horizontal scan or a vertical scan and the second scan order is not the horizontal scan or the vertical scan.

29. The apparatus of claim 20, wherein, to determine the contexts, the video coder is configured to determine the contexts for the significance flags of the transform coefficients of the block based on the determined scan order and based on size of the block.

30. The apparatus of claim 20, wherein the video coder is configured to:

determine whether size of the block is a first size or a second size,

wherein, if the size of the block is the first size, the video coder is configured to determine the contexts that are the same for all scan orders, and

wherein, if the size of the block is the second size, the video coder is configured to determine the contexts that are different for at least two different scan orders.

31. The apparatus of claim 20, wherein the block comprises an 8x8 block of transform coefficients.

32. The apparatus of claim 20, wherein the apparatus comprises one of:
a microprocessor;
an integrated circuit (IC); and
a wireless communication device that includes the video coder.
33. An apparatus for coding video data, the apparatus comprising:
means for determining a scan order for transform coefficients of a block;
means for determining contexts for significance flags of the transform coefficients of the block based on the determined scan order; and
means for context adaptive binary arithmetic coding (CABAC) the significance flags of the transform coefficients based at least on the determined contexts.
34. The apparatus of claim 33, wherein the means for determining the contexts comprises means for determining the contexts based on size of the block, positions of the transform coefficients within the block, and the scan order.
35. A computer-readable storage medium having instructions stored thereon that when executed cause one or more processors of an apparatus for coding video data to:
determine a scan order for transform coefficients of a block;
determine contexts for significance flags of the transform coefficients of the block based on the determined scan order; and
context adaptive binary arithmetic coding (CABAC) code the significance flags of the transform coefficients based at least on the determined contexts.
36. The computer-readable storage medium of claim 35, wherein the instructions that cause the one or more processors to determine the contexts comprise instructions that cause the one or more processors to determine the contexts based on size of the block, positions of the transform coefficients within the block, and the scan order.