

**SHAPE ADAPTIVE DISCRETE COSINE TRANSFORM FOR GEOMETRIC  
PARTITIONING WITH AN ADAPTIVE NUMBER OF REGIONS**

**CROSS-REFERENCE TO RELATED APPLICATIONS**

This application claims the benefit of priority of U.S. Provisional Patent Application  
5 Serial No. 62/797,799, filed on January 28, 2019, and titled “SHAPE ADAPTIVE DISCRETE  
COSINE TRANSFORM FOR GEOMETRIC PARTITIONING WITH AN ADAPTIVE  
NUMBER OF REGIONS,” which is incorporated by reference herein in its entirety.

**FIELD OF THE INVENTION**

The present invention generally relates to the field of video compression. In particular,  
10 the present invention is directed to a shape adaptive discrete cosine transform for geometric  
partitioning with an adaptive number of regions.

**BACKGROUND**

A video codec can include an electronic circuit or software that compresses or  
decompresses digital video. It can convert uncompressed video to a compressed format or vice  
15 versa. In the context of video compression, a device that compresses video (and/or performs  
some function thereof) can typically be called an encoder, and a device that decompresses video  
(and/or performs some function thereof) can be called a decoder.

A format of the compressed data can conform to a standard video compression  
specification. The compression can be lossy in that the compressed video lacks some information  
20 present in the original video. A consequence of this can include that decompressed video can  
have lower quality than the original uncompressed video because there is insufficient  
information to accurately reconstruct the original video.

There can be complex relationships between the video quality, the amount of data used to  
represent the video (e.g., determined by the bit rate), the complexity of the encoding and  
25 decoding algorithms, sensitivity to data losses and errors, ease of editing, random access, end-to-  
end delay (e.g., latency), and the like.

**SUMMARY OF THE DISCLOSURE**

In an aspect, a decoder includes circuitry configured to receive a bitstream, determine a  
first region, a second region, and a third region of a current block according to a geometric

partitioning mode, and decode the current block using an inverse discrete cosine transformation for each of the first region, the second region, and the third region.

In another aspect, a decoder includes circuitry configured to receive a bitstream determine a first region, a second region, and a third region of a current block and according to a  
5 geometric partitioning mode, determine, from a signal contained in the bitstream, a coding transformation type to decode each of the first region, the second region, and/or the third region, the coding transformation type characterizing at least an inverse block discrete cosine transformation and an inverse shape adaptive discrete cosine transformation and decode the current block, the decoding of the current block including using the determined transformation  
10 type for inverse transformation for each of the first region, the second region and/or the third region.

In another aspect, a method includes receiving, by a decoder, a bitstream, determining a first region, a second region, and a third region of a current block and according to a geometric partitioning mode, determining, from a signal contained in the bitstream, a coding transformation  
15 type to decode the first region, the second region, and/or the third region, the coding transformation type characterizing at least an inverse block discrete cosine transformation or an inverse shape adaptive discrete cosine transformation, and decoding the current block, the decoding of the current block including using the determined transformation type for inverse transformation for each of the first region, the second region, and/or the third region.

20 The details of one or more variations of the subject matter described herein are set forth in the accompanying drawings and the description below. Other features and advantages of the subject matter described herein will be apparent from the description and drawings, and from the claims.

## DESCRIPTION OF DRAWINGS

25 For the purpose of illustrating the invention, the drawings show aspects of one or more embodiments of the invention. However, it should be understood that the present invention is not limited to the precise arrangements and instrumentalities shown in the drawings, wherein: FIG. 1 is an illustration showing an example of a residual block (e.g., current block) with exponential partitioning where there are three segments with different prediction errors;

FIG. 2 is a system block diagram illustrating an example video encoder capable of shape adaptive discrete cosine transformation (SA-DCT) for geometric partitioning with an adaptive number of regions that can improve complexity and processing performance for video encoding and decoding;

5 FIG. 3 is a process flow diagram illustrating an example process of encoding a video with SA-DCT for geometric partitioning with an adaptive number of regions;

FIG. 4 is a system block diagram illustrating an example decoder capable of decoding a bitstream using SA-DCT for geometric partitioning with an adaptive number of regions;

FIG. 5 is a process flow diagram illustrating an example process of decoding a bitstream using  
10 SA-DCT for geometric partitioning with an adaptive number of regions; and

FIG. 6 is a block diagram of a computing system that can be used to implement any one or more of the methodologies disclosed herein and any one or more portions thereof.

The drawings are not necessarily to scale and may be illustrated by phantom lines, diagrammatic representations and fragmentary views. In certain instances, details that are not  
15 necessary for an understanding of the embodiments or that render other details difficult to perceive may have been omitted. Like reference symbols in the various drawings indicate like elements.

### DETAILED DESCRIPTION

Embodiments presented in this disclosure pertain to encoding and decoding blocks in  
20 geometric partitioning, where not all blocks are necessarily rectangular. Embodiments may include and/or be configured to perform encoding and/or decoding using discrete cosine transformations (DCT) and/or inverse DCTs. In some embodiments presented herein, a choice of DCT is made as a function of information content in geometrically partitioned blocks. In some existing video encoding and decoding approaches, all blocks are rectangular, and the  
25 residual is encoded using a regular Block DCT (B-DCT) for the entire rectangular block. However, in geometric partitioning where a block can be partitioned into multiple non-rectangular regions, use of regular B-DCT can inefficiently represent the underlying pixel information for some blocks and can require unnecessary computing resources to perform. In some implementations of the current subject matter, when using a geometric partitioning mode,  
30 an encoder may use Shape Adaptive DCT (SA-DCT) alternatively or additionally to B-DCT. In

some embodiments, an encoder may select between B-DCT and SA-DCT for each region of a block such as a geometrically partitioned block, based on a level of prediction error of that region; selection may be signaled in the bitstream for use in decoding. By encoding and/or decoding a non-rectangular region using either B-DCT or SA-DCT and signaling such selection, a bitrate of transmission in a bitstream may be reduced because a residual may be represented more efficiently, and computational resources required to perform the processing may be reduced as a result. The current subject matter may be applicable to relatively larger blocks, such as blocks with a size of 128 x 128 or 64 x 64, for example. In some implementations, geometric partitioning may involve partitioning a current block into an adaptive number of regions, such as three or more regions for a given current block; a DCT transform type (e.g., B-DCT or SA-DCT) may be signaled for each region.

In an embodiment, a B-DCT may be a DCT performed using an NXN invertible matrix on an NXN block of numerical values such as without limitation chroma and/or luma values of a corresponding NXN array of pixels. For instance, and as a non-limiting example, where an NxN matrix  $X$  is to be transformed, a “DCT-I” transformation may compute each element of a transformed matrix as:

$$X_k = \frac{1}{2}(x_0 + (-1)^k x_{N-1}) + \sum_{n=1}^{N-2} x_n \cos \left[ \frac{\pi}{N-1} nk \right]$$

For  $k = 0, \dots, N-1$ . As a further non-limiting example, a “DCT-II” transformation may compute transformed matrix values as:

$$X_k = \sum_{n=1}^{N-1} x_n \cos \left[ \frac{\pi}{N} \left( n + \frac{1}{2} \right) k \right]$$

For  $k = 0, \dots, N-1$ . As an illustrative example, where blocks are 4 x 4 blocks of pixels, the generalized discrete cosine transform matrix may include a generalized discrete cosine transform II matrix taking the form of:

$$T = \begin{pmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{pmatrix}$$

where  $a$  is  $\frac{1}{2}$ ,  $b$  is  $\sqrt{\frac{1}{2}} \cos \frac{\pi}{8}$ , and  $c$  is  $\sqrt{\frac{1}{2}} \cos \frac{3\pi}{8}$ .

In some implementations, an integer approximation of a transform matrix may be utilized, which may be used for efficient hardware and software implementations. For example, where blocks are 4x4 blocks of pixels, a generalized discrete cosine transform matrix may include a generalized discrete cosine transform II matrix taking the form of:

$$T_{INT} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix}.$$

Inverse B-DCT may be computed by a second matrix multiplication using the same NxN transform matrix; a resulting output may be normalized to recover original values. For instance, an inverse DCT-I may be multiplied by  $\frac{2}{N-1}$  for normalization.

An SA-DCT may be performed on a non-rectangular array of pixels. In an embodiment, an SA-DCT may be computed by performing a one-dimensional version of a DCT such as a DCT-I, DCT-II, or the like against vectors representing vertical columns of pixel values in a shape in interest, followed by resulting values being grouped into horizontal vectors and subjected to a one-dimensional DCT a second time; the second DCT may result in a completed transformation of pixel values. Variations of SA-DCT may further scale and/or normalize by coefficients to correct for mean weighting defects and/or non-orthonormal defects introduced by the above transformation, quantization of outputs of the above transformation and/or inversion of transformation outputs and/or quantized transformation outputs. Further corrections may be performed, without limitation, by preceding the above SA-DCT process by subtracting an individual mean value of a subject image region from each pixel value or a scaled version thereof, potentially in combination with one or other of the scaling processes applied before and/or after transformation, quantization, and/or inverse transformation. Persons skilled in the art, upon reviewing the entirety of this disclosure, will be aware of various alternative or additional variations on an SA-DCT process that may be applied consistently with the above description.

Motion compensation may include an approach to predict a video frame or a portion thereof given previous and/or future frames by accounting for motion of a camera and/or of objects in a video containing and/or represented by current, previous, and/or future frames. Motion compensation may be employed in encoding and decoding of video data for video compression, for example in encoding and decoding using the Motion Picture Experts Group

(MPEG)-2 (also referred to as advanced video coding (AVC)) standard. Motion compensation may describe a picture in terms of a transformation of a reference picture to a current picture. Reference picture may be previous in time or from the future when compared to current picture. When images can be accurately synthesized from previously transmitted and/or stored images, compression efficiency can be improved.

Block partitioning, as used in this disclosure, may refer to a method in video coding to find regions of similar motion. Some form of block partitioning can be found in video codec standards including MPEG-2, H.264 (also referred to as AVC or MPEG-4 Part 10), and H.265 (also referred to as High Efficiency Video Coding (HEVC)). In example block partitioning approaches, non-overlapping blocks of a video frame may be partitioned into rectangular sub-blocks to find block partitions that contain pixels with similar motion. This approach may work well when all pixels of a block partition have similar motion. Motion of pixels in a block may be determined relative to previously coded frames.

Shape-adaptive DCT and/or B-DCT may be effectively used in geometric partitioning with adaptive number of regions. FIG. 1 is an illustration showing a non-limiting example of a residual block (e.g., current block) 100 sized 64x64 or 128x128 with geometric partitioning where there are three segments, S0, S1, and S2 with different prediction errors; although three segments are illustrated in FIG. 1 for exemplary purposes, a greater or lesser number of segments may alternatively or additionally be employed. Current block may be geometrically partitioned according to two line segments (P1P2 and P3P4), which may divide the current block into the three regions S0, S1, and S2. In this example, S0 may have a relatively high prediction error while S1 and S2 can have a relatively lower prediction error. For segment S0 (also referred to as region), the encoder may select and use B-DCT for residual coding. For segments S1 and S2 with low prediction error, the encoder may select and use SA-DCT. The selection of residual encoding transformation can be based on a prediction error (e.g., size of the residual). Because SA-DCT algorithm is relatively simpler in terms of complexity and does not require as many computation as the B-DCT, utilizing SA-DCT for lower prediction error residual coding may improve complexity and processing performance for video encoding and decoding.

Accordingly, and still referring to FIG. 1, SA-DCT may be signaled as an additional transform choice to full block DCT for segments with low prediction errors. What is considered low or high error may be a parameter that can be set at the encoder and may vary based on

application. A choice of transformation type may be signaled in the bitstream. At a decoder, bitstream may be parsed, and for a given current block, a residual may be decoded using a transform type signaled in the bitstream. In some implementations, a number of coefficients associated with the transform may alternatively or additionally be signaled in the bitstream.

5           In more detail, and continuing to refer to FIG. 1, geometric partitioning with an adaptive number of regions may include techniques for video encoding and decoding in which a rectangular block is further divided into two or more regions that may be non-rectangular. For example, FIG. 1 illustrates a non-limiting example of geometric partitioning at the pixel level with an adaptive number of regions. An example rectangular block 100 (which can have a width  
10 of M pixels and a height of N pixels, denoted as MxN pixels) may be divided along a line segment P1P2 and P3P4 into three regions (S0, S1, and S2). When pixels in S0 have similar motion, a motion vector may describe the motion of all pixels in that region; the motion vector can be used to compress region S0. Similarly, when pixels in region S1 have similar motion, an associated motion vector may describe the motion of pixels in region S1. Similarly, when pixels  
15 in region S2 have similar motion, an associated motion vector may describe the motion of pixels in region S2. Such a geometric partition may be signaled to the receiver (e.g., decoder) by encoding positions P1, P2, P3, P4 and/or representations of these positions, such as without limitation using coordinates such as polar coordinates, cartesian coordinates, or the like, indices into predefined templates, or other characterizations of the partitions) in a video bitstream.

20           Still referring to FIG. 1, when encoding video data utilizing geometric partitioning at the pixel level, a line segment P1P2 (or more specifically points P1 and P2) may be determined. In order to determine line segment P1P2 (or more specifically points P1 and P2) that best divides the block when utilizing geometric partitioning at the pixel level, the possible combinations of points P1 and P2 depends on M and N, which are the block width and height. For a block of size  
25 MxN, there are  $(M-1) \times (N-1) \times 3$  possible partitions. Identifying the right partition thus can become a computationally expensive task of evaluating motion estimation for all possible partitions, which can increase the amount of time and/or processing power required to encode a video as compared to encoding using rectangular partitioning (e.g., without geometric partitioning at the pixel level). What constitutes the best or right partition can be determined  
30 according to a metric and may change from implementation to implementation.

In some implementations, and still referring to FIG. 1, partitioning occurs iteratively in that a first partition can be determined (e.g., determine line P1P2 and associated regions) forming two regions, and then one of those regions is further partitioned. For example, the partitioning described with reference to FIG. 1 can be performed to partition a block into two regions. One of those regions can be further partitioned (e.g., to form new region S1 and region S2). The process can continue to perform block level geometric partitioning until a stopping criteria is reached.

FIG. 2 is a system block diagram illustrating an example video encoder 200 capable of SA-DCT and/or B-DCT for geometric partitioning with an adaptive number of regions that can improve complexity and processing performance for video encoding and decoding. The example video encoder 200 receives an input video 205, which can be initially segmented or dividing according to a processing scheme, such as a tree-structured macro block partitioning scheme (e.g., quad-tree plus binary tree). An example of a tree-structured macro block partitioning scheme can include partitioning a picture frame into large block elements called coding tree units (CTU). In some implementations, each CTU can be further partitioned one or more times into a number of sub-blocks called coding units (CU). The final result of this partitioning can include a group of sub-blocks that can be called predictive units (PU). Transform units (TU) can also be utilized. Such a partitioning scheme can include performing geometric partitioning with an adaptive number of regions according to some aspects of the current subject matter.

With continued reference to FIG. 2, the example video encoder 200 includes an intra prediction processor 215, a motion estimation / compensation processor 220 (also referred to as an inter prediction processor) capable of supporting geometric partitioning with an adaptive number of regions, a transform / quantization processor 225, an inverse quantization / inverse transform processor 230, an in-loop filter 235, a decoded picture buffer 240, and an entropy coding processor 245. In some implementations, the motion estimation / compensation processor 220 can perform geometric partitioning. Bitstream parameters that signal geometric partitioning modes can be input to the entropy coding processor 245 for inclusion in the output bitstream 250.

In operation, and continuing to refer to FIG. 2, for each block of a frame of the input video 205, whether to process the block via intra picture prediction or using motion estimation / compensation can be determined. The block can be provided to the intra prediction processor 210 or the motion estimation / compensation processor 220. If the block is to be processed via intra prediction, the intra prediction processor 210 can perform the processing to output the

predictor. If the block is to be processed via motion estimation / compensation, the motion estimation / compensation processor 220 can perform the processing including use of geometric partitioning to output the predictor.

Still referring to FIG. 2, a residual can be formed by subtracting the predictor from the input video. The residual can be received by the transform / quantization processor 225, which can determine whether the prediction error (e.g., residual size) is considered “high” or “low” error (for example, by comparing a size or error metric of the residual to a threshold). Based on the determination, the transform / quantization processor 225 can select a transform type, which can include B-DCT and SA-DCT. In some implementations, the transform / quantization processor 225 selects a transform type of B-DCT where the residual is considered to have a high error and selects a transform type of SA-DCT where the residual is considered to have a low error. Based on the selected transform type, the transform / quantization processor 225 can perform transformation processing (e.g., SA-DCT or B-DCT) to produce coefficients, which can be quantized. The quantized coefficients and any associated signaling information (which can include the selected transform type and/or the number of coefficients used) can be provided to the entropy coding processor 245 for entropy encoding and inclusion in the output bitstream 250. The entropy encoding processor 245 can support encoding of signaling information related to SA-DCT for geometric partitioning with adaptive number of regions. In addition, the quantized coefficients can be provided to the inverse quantization / inverse transformation processor 230, which can reproduce pixels, which can be combined with the predictor and processed by the in loop filter 235, the output of which is stored in the decoded picture buffer 240 for use by the motion estimation / compensation processor 220 that is capable of supporting geometric partitioning with an adaptive number of regions.

Referring now to FIG. 3, a process flow diagram illustrating an example process 300 of encoding a video with SA-DCT for geometric partitioning with an adaptive number of regions that can improve complexity and processing performance for video encoding and decoding is illustrated. At step 310, a video frame may undergo initial block segmentation, for example, using a tree-structured macro block partitioning scheme that may include partitioning a picture frame into CTUs and CUs. At 320, a block may be selected for geometric partitioning. Selection may include identifying according to a metric rule that a block is to be processed according to a

geometric partitioning mode. At step 330, a selected block may be partitioned into three or more non-rectangular regions according to geometric partitioning mode.

At step 340, and still referring to FIG. 3, a transform type (also referred to as a transformation type) for each geometrically partitioned region may be determined. This may include determining whether a prediction error (e.g., residual size) is considered “high” or “low” error (for example, by comparing a size or error metric of the residual to a threshold). Based on determination, a transform type may be selected, for instance using a quadtree plus binary decision tree process as described below, which transform type may include without limitation B-DCT or SA-DCT. In some implementations, a transform type of B-DCT is selected where residual is considered to have a high error and a transform type of SA-DCT is selected where residual is considered to have a low error. Based on the selected transform type, transformation processing (e.g., SA-DCT or B-DCT) may be performed to produce coefficients, which may be quantized.

At step 350, and continuing to refer to FIG. 3, a determined transform type may be signaled in the bitstream. The transformed and quantized residual can be included in the bitstream. In some implementations, the number of transform coefficients can be signaled in the bitstream.

FIG. 4 is a system block diagram illustrating a non-limiting example of a decoder 400 capable of decoding a bitstream 470 using DCT, including without limitation SA-DCT and/or B-DCT, for geometric partitioning with an adaptive number of regions, which may improve complexity and processing performance for video encoding and decoding. Decoder 400 includes an entropy decoder processor 410, an inverse quantization and inverse transformation processor 420, a deblocking filter 430, a frame buffer 440, motion compensation processor 450 and intra prediction processor 460. In some implementations, bitstream 470 includes parameters that signal a geometric partitioning mode and transformation type. In some implementations, bitstream 470 includes parameters that signal the number of transform coefficients. The motion compensation processor 450 can reconstruct pixel information using geometric partitioning as described herein.

In operation, and still referring to FIG. 4, bitstream 470 may be received by the decoder 400 and input to entropy decoder processor 410, which may entropy decode the bitstream into quantized coefficients. Quantized coefficients may be provided to inverse quantization and

inverse transformation processor 420, which may determine a coding transformation type (e.g., B-DCT or SA-DCT) and perform inverse quantization and inverse transformation according to the determined coding transformation type to create a residual signal. In some implementations, inverse quantization and inverse transformation processor 420 may determine a number of  
5 transform coefficients and perform inverse transformation according to the determined number of transform coefficients.

Still referring to FIG. 4, residual signal may be added to an output of motion compensation processor 450 or intra prediction processor 460 according to a processing mode. Output of a motion compensation processor 450 and intra prediction processor 460 may include  
10 a block prediction based on a previously decoded block. A sum of the prediction and residual may be processed by deblocking filter 430 and stored in a frame buffer 440. For a given block, (e.g., CU or PU), when a bitstream 470 signals that a partitioning mode is block level geometric partitioning, motion compensation processor 450 may construct a prediction based on the geometric partition approach described herein.

FIG. 5 is a process flow diagram illustrating an example process 500 of decoding a bitstream using SA-DCT for geometric partitioning with an adaptive number of regions, which can improve complexity and processing performance for video encoding and decoding. At step 510, a bitstream is received, which may include a current block (e.g., CTU, CU, PU). Receiving may include extracting and/or parsing current block and associated signaling information from  
20 bitstream. Decoder may extract or determine one or more parameters that characterize the geometric partitioning. These parameters may include, for example, indices of a start and end of a line segment (e.g., P1, P2, P3, P4); extraction or determining may include identifying and retrieving the parameters from the bitstream (e.g., parsing the bitstream).

At step 520, and still referring to FIG. 5, a first region, a second region, and a third region  
25 of the current block may be determined and according to a geometric partitioning mode. Determining may include determining whether geometric partitioning mode is enabled (e.g., true) for the current block. If geometric partitioning mode is not enabled (e.g., false), decoder may process current block using an alternative partitioning mode. If geometric partitioning mode is enabled (e.g., true), three or more regions may be determined and/or processed.

At optional step 530, and continuing to refer to FIG. 5, a coding transformation type may  
30 be determined. A coding transformation type may be signaled in bitstream. For example,

bitstream may be parsed to determine a coding transformation type, which may specify B-DCT or SA-DCT. Determined coding transformation type may be for decoding a first region, a second region, and/or a third region.

At 540, and still referring to FIG. 5, a current block may be decoded. Decoding of current  
5 block may include using a determined transform type for inverse transformation for each of a first region, a second region, and/or a third region. Decoding may include determining an associated motion information for each region and according to geometric partitioning mode.

Although a few variations have been described in detail above, other modifications or additions are possible. For example, the geometric partitioning can be signaled in the bitstream  
10 based on rate-distortion decisions in the encoder. The coding can be based on a combination of regular pre-defined partitions (e.g., templates), temporal and spatial prediction of the partitioning, and additional offsets. Each geometrically partitioned region can utilize motion compensated prediction or intra-prediction. The boundary of the predicted regions can be smoothed before the residual is added.

15 In some implementations, a quadtree plus binary decision tree (QTBT) may be implemented. In QTBT, at the Coding Tree Unit level, partition parameters of QTBT are dynamically derived to adapt to the local characteristics without transmitting any overhead. Subsequently, at the Coding Unit level, a joint-classifier decision tree structure may eliminate unnecessary iterations and control the risk of false prediction. In some implementations,  
20 geometric partitioning with an adaptive number of regions may be available as an additional partitioning option available at every leaf node of the QTBT.

In some implementations, a decoder may include a partition processor that generates geometric partition for a current block and provides all partition-related information for dependent processes. Partition processor may directly influence motion compensation as it may  
25 be performed segment-wise in case a block is geometrically partitioned. Further, partition processor may provide shape information to intra-prediction processor and transform coding processor.

In some implementations, additional syntax elements may be signaled at different hierarchy levels of the bitstream. For enabling geometric partitioning with an adaptive number of  
30 regions for an entire sequence, an enable flag may be coded in a Sequence Parameter Set (SPS). Further, a CTU flag may be coded at the coding tree unit (CTU) level to indicate whether any

coding units (CU) use geometric partitioning with an adaptive number of regions. A CU flag may be coded to indicate whether a current coding unit utilizes geometric partitioning with an adaptive number of regions. Parameters which specify a line segment on block may be coded. For each region, a flag may be decoded, which may specify whether a current region is inter- or  
5 intra-predicted.

In some implementations, a minimum region size may be specified.

The subject matter described herein provides many technical advantages. For example, some implementations of the current subject matter can provide for partitioning of blocks that reduces complexity while increasing compression efficiency. In some implementations, blocking  
10 artifacts at object boundaries can be reduced.

It is to be noted that any one or more of the aspects and embodiments described herein may be conveniently implemented using digital electronic circuitry, integrated circuitry, specially designed application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs) computer hardware, firmware, software, and/or combinations thereof, as realized  
15 and/or implemented in one or more machines (*e.g.*, one or more computing devices that are utilized as a user computing device for an electronic document, one or more server devices, such as a document server, etc.) programmed according to the teachings of the present specification, as will be apparent to those of ordinary skill in the computer art. These various aspects or features may include implementation in one or more computer programs and/or software that are  
20 executable and/or interpretable on a programmable system including at least one programmable processor, which may be special or general purpose, coupled to receive data and instructions from, and to transmit data and instructions to, a storage system, at least one input device, and at least one output device. Appropriate software coding may readily be prepared by skilled  
programmers based on the teachings of the present disclosure, as will be apparent to those of  
25 ordinary skill in the software art. Aspects and implementations discussed above employing software and/or software modules may also include appropriate hardware for assisting in the implementation of the machine executable instructions of the software and/or software module.

Such software may be a computer program product that employs a machine-readable storage medium. A machine-readable storage medium may be any medium that is capable of  
30 storing and/or encoding a sequence of instructions for execution by a machine (*e.g.*, a computing device) and that causes the machine to perform any one of the methodologies and/or

embodiments described herein. Examples of a machine-readable storage medium include, but are not limited to, a magnetic disk, an optical disc (*e.g.*, CD, CD-R, DVD, DVD-R, etc.), a magneto-optical disk, a read-only memory “ROM” device, a random access memory “RAM” device, a magnetic card, an optical card, a solid-state memory device, an EPROM, an EEPROM,

5 Programmable Logic Devices (PLDs), and/or any combinations thereof. A machine-readable medium, as used herein, is intended to include a single medium as well as a collection of physically separate media, such as, for example, a collection of compact discs or one or more hard disk drives in combination with a computer memory. As used herein, a machine-readable storage medium does not include transitory forms of signal transmission.

10 Such software may also include information (*e.g.*, data) carried as a data signal on a data carrier, such as a carrier wave. For example, machine-executable information may be included as a data-carrying signal embodied in a data carrier in which the signal encodes a sequence of instruction, or portion thereof, for execution by a machine (*e.g.*, a computing device) and any related information (*e.g.*, data structures and data) that causes the machine to perform any one of  
15 the methodologies and/or embodiments described herein.

Examples of a computing device include, but are not limited to, an electronic book reading device, a computer workstation, a terminal computer, a server computer, a handheld device (*e.g.*, a tablet computer, a smartphone, etc.), a web appliance, a network router, a network switch, a network bridge, any machine capable of executing a sequence of instructions that  
20 specify an action to be taken by that machine, and any combinations thereof. In one example, a computing device may include and/or be included in a kiosk.

FIG. 6 shows a diagrammatic representation of one embodiment of a computing device in the exemplary form of a computer system 600 within which a set of instructions for causing a control system to perform any one or more of the aspects and/or methodologies of the present disclosure may be executed. It is also contemplated that multiple computing devices may be  
25 utilized to implement a specially configured set of instructions for causing one or more of the devices to perform any one or more of the aspects and/or methodologies of the present disclosure. Computer system 600 includes a processor 604 and a memory 608 that communicate with each other, and with other components, via a bus 612. Bus 612 may include any of several  
30 types of bus structures including, but not limited to, a memory bus, a memory controller, a

peripheral bus, a local bus, and any combinations thereof, using any of a variety of bus architectures.

Memory 608 may include various components (*e.g.*, machine-readable media) including, but not limited to, a random-access memory component, a read only component, and any combinations thereof. In one example, a basic input/output system 616 (BIOS), including basic routines that help to transfer information between elements within computer system 600, such as during start-up, may be stored in memory 608. Memory 608 may also include (*e.g.*, stored on one or more machine-readable media) instructions (*e.g.*, software) 620 embodying any one or more of the aspects and/or methodologies of the present disclosure. In another example, memory 608 may further include any number of program modules including, but not limited to, an operating system, one or more application programs, other program modules, program data, and any combinations thereof.

Computer system 600 may also include a storage device 624. Examples of a storage device (*e.g.*, storage device 624) include, but are not limited to, a hard disk drive, a magnetic disk drive, an optical disc drive in combination with an optical medium, a solid-state memory device, and any combinations thereof. Storage device 624 may be connected to bus 612 by an appropriate interface (not shown). Example interfaces include, but are not limited to, SCSI, advanced technology attachment (ATA), serial ATA, universal serial bus (USB), IEEE 1394 (FIREWIRE), and any combinations thereof. In one example, storage device 624 (or one or more components thereof) may be removably interfaced with computer system 600 (*e.g.*, via an external port connector (not shown)). Particularly, storage device 624 and an associated machine-readable medium 628 may provide nonvolatile and/or volatile storage of machine-readable instructions, data structures, program modules, and/or other data for computer system 600. In one example, software 620 may reside, completely or partially, within machine-readable medium 628. In another example, software 620 may reside, completely or partially, within processor 604.

Computer system 600 may also include an input device 632. In one example, a user of computer system 600 may enter commands and/or other information into computer system 600 via input device 632. Examples of an input device 632 include, but are not limited to, an alphanumeric input device (*e.g.*, a keyboard), a pointing device, a joystick, a gamepad, an audio input device (*e.g.*, a microphone, a voice response system, etc.), a cursor control device (*e.g.*, a

mouse), a touchpad, an optical scanner, a video capture device (*e.g.*, a still camera, a video camera), a touchscreen, and any combinations thereof. Input device 632 may be interfaced to bus 612 via any of a variety of interfaces (not shown) including, but not limited to, a serial interface, a parallel interface, a game port, a USB interface, a FIREWIRE interface, a direct interface to bus 612, and any combinations thereof. Input device 632 may include a touch screen interface that may be a part of or separate from display 636, discussed further below. Input device 632 may be utilized as a user selection device for selecting one or more graphical representations in a graphical interface as described above.

A user may also input commands and/or other information to computer system 600 via storage device 624 (*e.g.*, a removable disk drive, a flash drive, etc.) and/or network interface device 640. A network interface device, such as network interface device 640, may be utilized for connecting computer system 600 to one or more of a variety of networks, such as network 644, and one or more remote devices 648 connected thereto. Examples of a network interface device include, but are not limited to, a network interface card (*e.g.*, a mobile network interface card, a LAN card), a modem, and any combination thereof. Examples of a network include, but are not limited to, a wide area network (*e.g.*, the Internet, an enterprise network), a local area network (*e.g.*, a network associated with an office, a building, a campus or other relatively small geographic space), a telephone network, a data network associated with a telephone/voice provider (*e.g.*, a mobile communications provider data and/or voice network), a direct connection between two computing devices, and any combinations thereof. A network, such as network 644, may employ a wired and/or a wireless mode of communication. In general, any network topology may be used. Information (*e.g.*, data, software 620, etc.) may be communicated to and/or from computer system 600 via network interface device 640.

Computer system 600 may further include a video display adapter 652 for communicating a displayable image to a display device, such as display device 636. Examples of a display device include, but are not limited to, a liquid crystal display (LCD), a cathode ray tube (CRT), a plasma display, a light emitting diode (LED) display, and any combinations thereof. Display adapter 652 and display device 636 may be utilized in combination with processor 604 to provide graphical representations of aspects of the present disclosure. In addition to a display device, computer system 600 may include one or more other peripheral output devices including, but not limited to, an audio speaker, a printer, and any combinations thereof. Such peripheral

output devices may be connected to bus 612 via a peripheral interface 656. Examples of a peripheral interface include, but are not limited to, a serial port, a USB connection, a FIREWIRE connection, a parallel connection, and any combinations thereof.

5 The foregoing has been a detailed description of illustrative embodiments of the invention. Various modifications and additions can be made without departing from the spirit and scope of this invention. Features of each of the various embodiments described above may be combined with features of other described embodiments as appropriate in order to provide a multiplicity of feature combinations in associated new embodiments. Furthermore, while the foregoing describes a number of separate embodiments, what has been described herein is  
10 merely illustrative of the application of the principles of the present invention. Additionally, although particular methods herein may be illustrated and/or described as being performed in a specific order, the ordering is highly variable within ordinary skill to achieve embodiments as disclosed herein. Accordingly, this description is meant to be taken only by way of example, and not to otherwise limit the scope of this invention.

15 In the descriptions above and in the claims, phrases such as “at least one of” or “one or more of” may occur followed by a conjunctive list of elements or features. The term “and/or” may also occur in a list of two or more elements or features. Unless otherwise implicitly or explicitly contradicted by the context in which it is used, such a phrase is intended to mean any of the listed elements or features individually or any of the recited elements or features in  
20 combination with any of the other recited elements or features. For example, the phrases “at least one of A and B;” “one or more of A and B;” and “A and/or B” are each intended to mean “A alone, B alone, or A and B together.” A similar interpretation is also intended for lists including three or more items. For example, the phrases “at least one of A, B, and C;” “one or more of A, B, and C;” and “A, B, and/or C” are each intended to mean “A alone, B alone, C alone, A and B  
25 together, A and C together, B and C together, or A and B and C together.” In addition, use of the term “based on,” above and in the claims is intended to mean, “based at least in part on,” such that an unrecited feature or element is also permissible.

The subject matter described herein can be embodied in systems, apparatus, methods, and/or articles depending on the desired configuration. The implementations set forth in the  
30 foregoing description do not represent all implementations consistent with the subject matter described herein. Instead, they are merely some examples consistent with aspects related to the

described subject matter. Although a few variations have been described in detail above, other modifications or additions are possible. In particular, further features and/or variations can be provided in addition to those set forth herein. For example, the implementations described above can be directed to various combinations and sub-combinations of the disclosed features and/or combinations and sub-combinations of several further features disclosed above. In addition, the logic flows depicted in the accompanying figures and/or described herein do not necessarily require the particular order shown, or sequential order, to achieve desirable results. Other implementations may be within the scope of the following claims.

**WHAT IS CLAIMED IS:**

1. A decoder, the decoder comprising circuitry configured to:  
 receive a bitstream;  
 determine a first region, a second region, and a third region of a current block according  
 5 to a geometric partitioning mode; and  
 decode the current block using an inverse discrete cosine transformation for each of the  
 first region, the second region, and the third region.
2. The decoder of claim 1, wherein the current block has a size of 128 x 128 or 64 x 64.
3. The decoder of claim 1, wherein a number of coefficients for inverse transformation of  
 10 the first region, the second region, and/or the third is signaled in the bitstream.
4. The decoder of claim 1, further configured to:  
 determine whether the geometric partitioning mode is enabled;  
 determine a first line segment for the current block; and  
 determine a second line segment for the current block;  
 15 wherein:  
 the decoding of the current block includes reconstructing pixel data using the first  
 line segment and the second line segment; and  
 the first line segment and the second line segment partition the current block into  
 the first region, the second region, and the third region.
- 20 5. The decoder of claim 4, wherein the first line segment characterizes the first region and  
 the second line segment characterizes the second region and the third region.
6. The decoder of claim 4, wherein reconstructing pixel data includes computing a predictor  
 for the first region using an associated motion vector contained in the bitstream.
7. The decoder of claim 1, further comprising:  
 25 an entropy decoder processor configured to receive the bitstream and decode the  
 bitstream into quantized coefficients;  
 an inverse quantization and inverse transformation processor configured to process the  
 quantized coefficients including performing an inverse discrete cosine according  
 to the determined coding transformation type;  
 30 a deblocking filter;  
 a frame buffer; and

- an intra prediction processor.
8. The decoder of claim 1, wherein the bitstream includes a parameter indicating whether geometric partitioning mode is enabled for the current block.
9. The decoder of claim 1, wherein the current block forms part of a quadtree plus binary  
5 decision tree.
10. The decoder of claim 1, wherein the current block is a non-leaf node of the quadtree plus binary decision tree.
11. The decoder of claim 1, wherein the current block is a coding tree unit or a coding unit.
12. The decoder of claim 1, wherein the first region is a coding unit or a prediction unit.
- 10 13. A decoder, the decoder comprising circuitry configured to:  
receive a bitstream;  
determine a first region, a second region, and a third region of a current block and  
according to a geometric partitioning mode;  
determine, from a signal contained in the bitstream, a coding transformation type to  
15 decode each of the first region, the second region, and/or the third region, the  
coding transformation type characterizing at least an inverse block discrete cosine  
transformation and an inverse shape adaptive discrete cosine transformation; and  
decode the current block, the decoding of the current block including using the  
determined transformation type for inverse transformation for each of the first  
20 region, the second region and/or the third region.
14. The decoder of claim 13, wherein the current block has a size of 128 x 128 or 64 x 64.
15. The decoder of claim 13, wherein a number of coefficients for inverse transformation of  
the first region, the second region, and/or the third is signaled in the bitstream.
16. The decoder of claim 13, further configured to:  
25 determine whether the geometric partitioning mode is enabled;  
determine a first line segment for the current block; and  
determine a second line segment for the current block;  
wherein:  
the decoding of the current block includes reconstructing pixel data using the first  
30 line segment and the second line segment; and

the first line segment and the second line segment partition the current block into the first region, the second region, and the third region.

17. The decoder of claim 16, wherein the first line segment characterizes the first region and the second line segment characterizes the second region and the third region.
- 5 18. The decoder of claim 16, wherein reconstructing pixel data includes computing a predictor for the first region using an associated motion vector contained in the bitstream.
19. The decoder of claim 13, further comprising:  
an entropy decoder processor configured to receive the bitstream and decode the  
bitstream into quantized coefficients;  
10 an inverse quantization and inverse transformation processor configured to process the quantized coefficients including performing an inverse discrete cosine according to the determined coding transformation type;  
a deblocking filter;  
a frame buffer; and  
15 an intra prediction processor.
20. The decoder of claim 13, wherein the bitstream includes a parameter indicating whether geometric partitioning mode is enabled for the current block.
21. The decoder of claim 13, wherein the current block forms part of a quadtree plus binary decision tree.
- 20 22. The decoder of claim 13, wherein the current block is a non-leaf node of the quadtree plus binary decision tree.
23. The decoder of claim 13, wherein the current block is a coding tree unit or a coding unit.
24. The decoder of claim 13, wherein the first region is a coding unit or a prediction unit.
25. A method comprising:  
25 receiving, by a decoder, a bitstream;  
determining a first region, a second region, and a third region of a current block and according to a geometric partitioning mode;  
determining, from a signal contained in the bitstream, a coding transformation type to  
decode the first region, the second region, and/or the third region, the coding  
30 transformation type characterizing at least an inverse block discrete cosine transformation or an inverse shape adaptive discrete cosine transformation; and

- decoding the current block, the decoding of the current block including using the  
determined transformation type for inverse transformation for each of the first  
region, the second region, and/or the third region.
26. The method of claim 25, wherein the current block has a size of 128 x 128 or 64 x 64.
- 5 27. The method of claim 25, wherein a number of coefficients for inverse transformation of  
the first region, the second region, and/or the third is signaled in the bitstream.
28. The method of claim 25, further comprising:  
determining, by the decoder, whether the geometric partitioning mode is enabled;  
determining, by the decoder, a first line segment for the current block; and  
10 determining, by the decoder, a second line segment for the current block;  
wherein:  
the decoding of the current block includes reconstructing pixel data using the first  
line segment and the second line segment; and  
the first line segment and the second line segment partition the current block into  
15 the first region, the second region, and the third region.
29. The method of claim 28, wherein the first line segment characterizes the first region and  
the second line segment characterizes the second region and the third region.
30. The method of claim 28, wherein reconstructing pixel data includes computing a  
predictor for the first region using an associated motion vector contained in the bitstream.
- 20 31. The method of claim 25, wherein the decoder comprises:  
an entropy decoder processor configured to receive the bitstream and decode the  
bitstream into quantized coefficients;  
an inverse quantization and inverse transformation processor configured to process the  
quantized coefficients including performing an inverse discrete cosine according  
25 to the determined coding transformation type;  
a deblocking filter;  
a frame buffer; and  
an intra prediction processor.
32. The method of claim 25, wherein the bitstream includes a parameter indicating whether  
30 block level geometric partitioning mode is enabled for the block.

33. The method of claim 25, wherein the current block forms part of a quadtree plus binary decision tree.
34. The method of claim 25, wherein the current block is a non-leaf node of the quadtree plus binary decision tree.
- 5 35. The method of claim 25, wherein the current block is a coding tree unit or a coding unit.
36. The method of claim 25, wherein the first region is a coding unit or a prediction unit.