US 20060268990A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2006/0268990 A1**

Lin et al. (43) **Pub. Date: Nov. 30, 2006**

(54) **ADAPTIVE VIDEO ENCODING USING A PERCEPTUAL MODEL**

(75) Inventors: **Chih-Lung Lin**, Redmond, WA (US); **Minghui Xia**, Bellevue, WA (US); **Pohsiang Hsu**, Redmond, WA (US); **Shou-Jen Wu**, Redmond, WA (US)

Correspondence Address:
KLARQUIST SPARKMAN LLP
121 S.W. SALMON STREET
SUITE 1600
PORTLAND, OR 97204 (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA

(21) Appl. No.: **11/137,924**

(22) Filed: **May 25, 2005**

Publication Classification

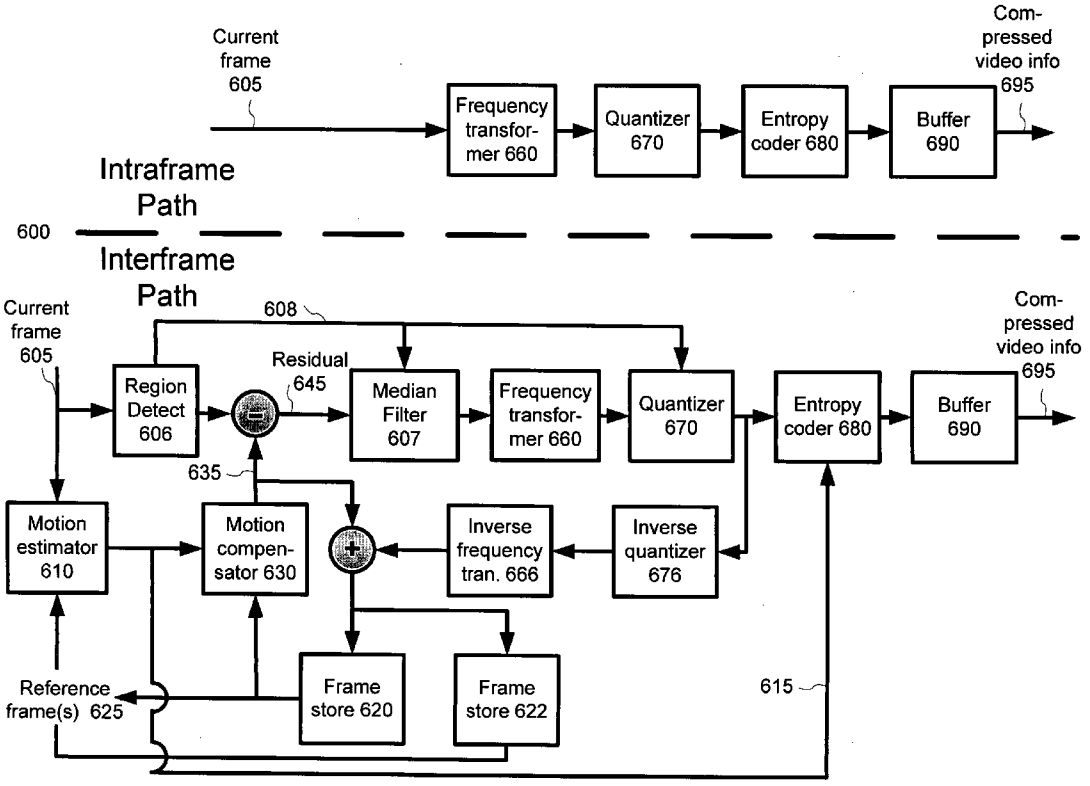
(51) **Int. Cl.**

H04N 11/04	(2006.01)
H04N 11/02	(2006.01)
H04B 1/66	(2006.01)
H04N 7/12	(2006.01)

(52) **U.S. Cl.** **375/240.24**; 375/240.2; 375/240.13

(57) **ABSTRACT**

A video encoder includes a region detector module that classifies blocks of video frames. An adaptive filter module applies a median filter to a block based upon a block classification assigned by the region detector module. An adaptive quantization module quantizes a block according to a quantization method adaptively determined based upon a block classification assigned by the region detection module. In one example, a video encoder adaptively determines a median filter selected using a block classification. In another example, a video encoder adaptively determines whether to drop an isolated last transform coefficient based on the block classification, and/or applies a dead-zone selected using the block classification.



Amazon.com, Inc., Amazon.com Services LLC - Ex. 1005, Page 000001
IPR2024-00921 (Amazon.com, Inc., Amazon.com Services LLC v. Nokia Technologies Oy)

FIG. 1 (PRIOR ART)

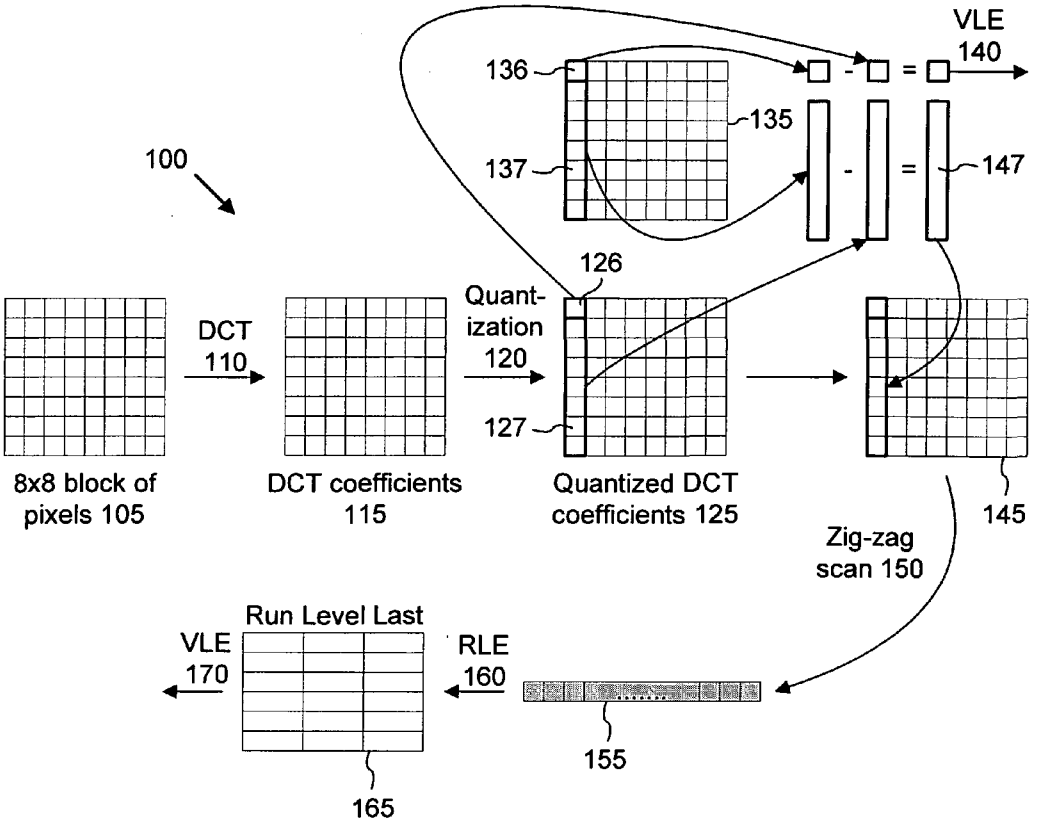


FIG. 2 (PRIOR ART)

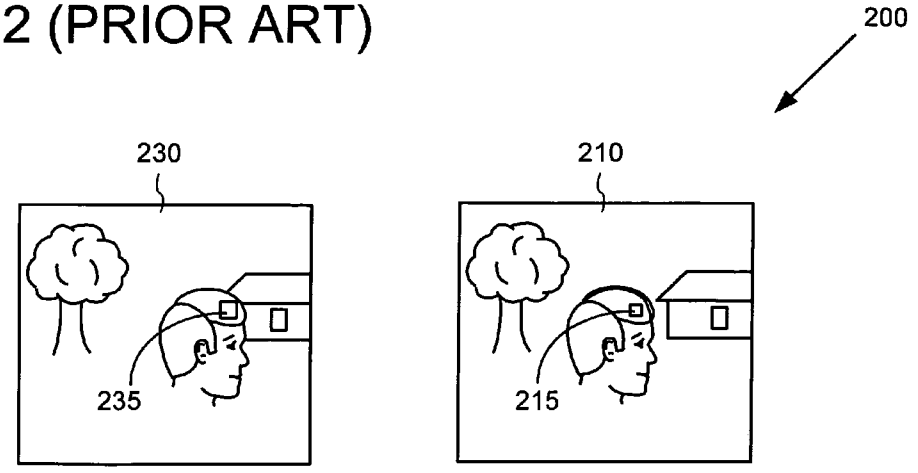
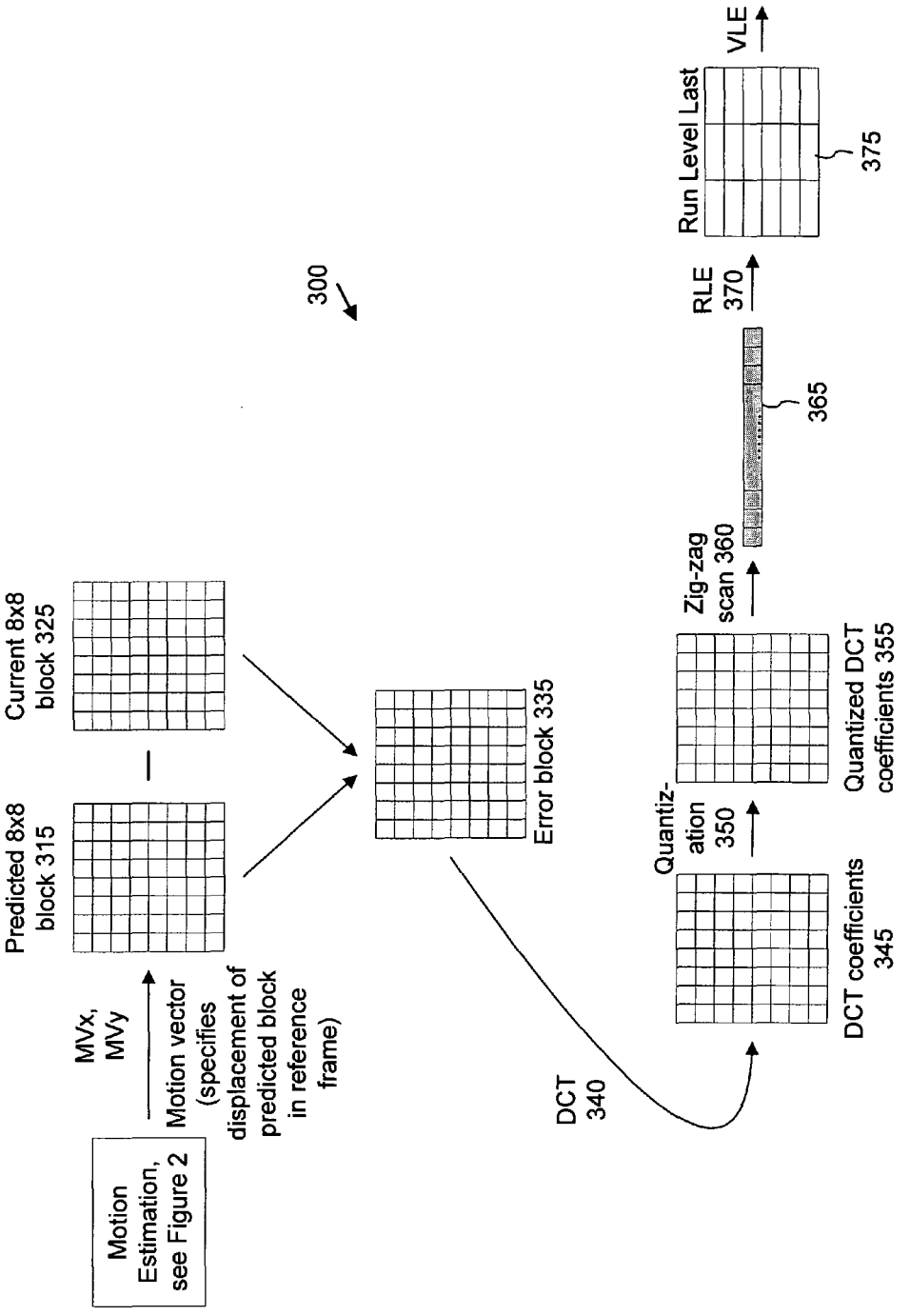


FIG. 3 (PRIOR ART)



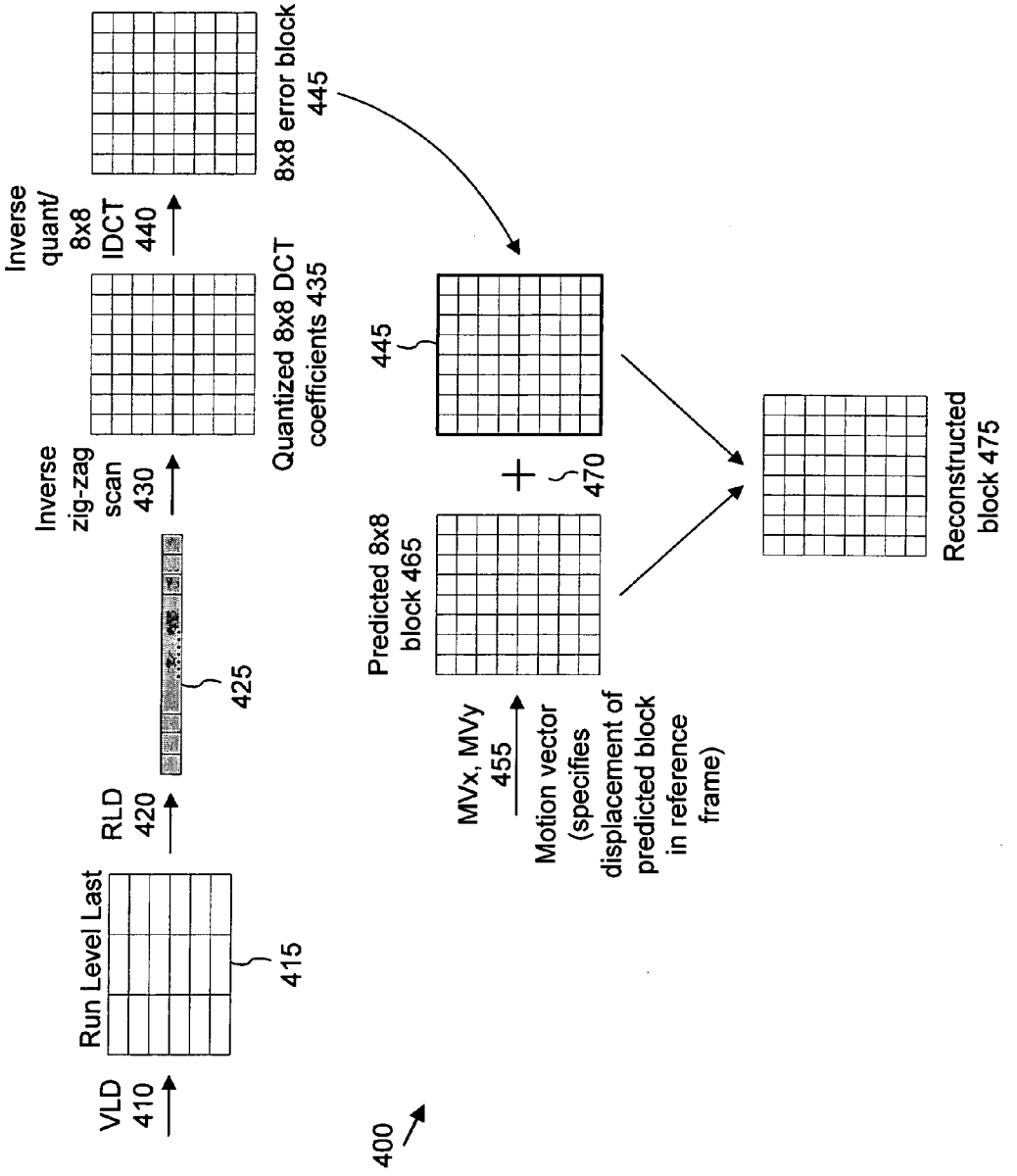
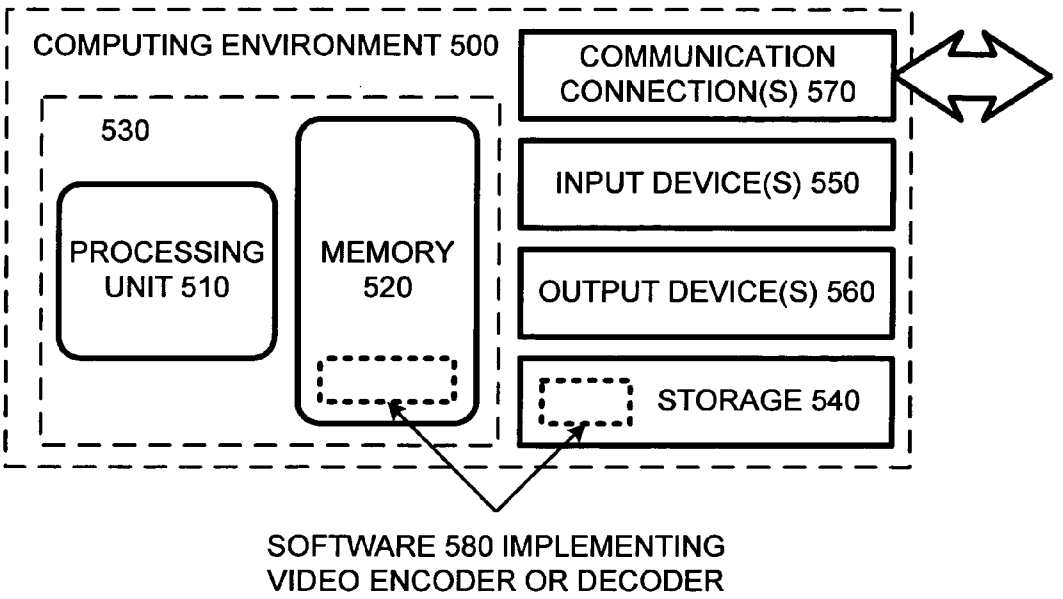


FIG. 4 (PRIOR ART)

FIG. 5


COMPUTING ENVIRONMENT 500
530
PROCESSING UNIT 510
MEMORY 520
COMMUNICATION CONNECTION(S) 570
INPUT DEVICE(S) 550
OUTPUT DEVICE(S) 560
STORAGE 540
SOFTWARE 580 IMPLEMENTING VIDEO ENCODER OR DECODER

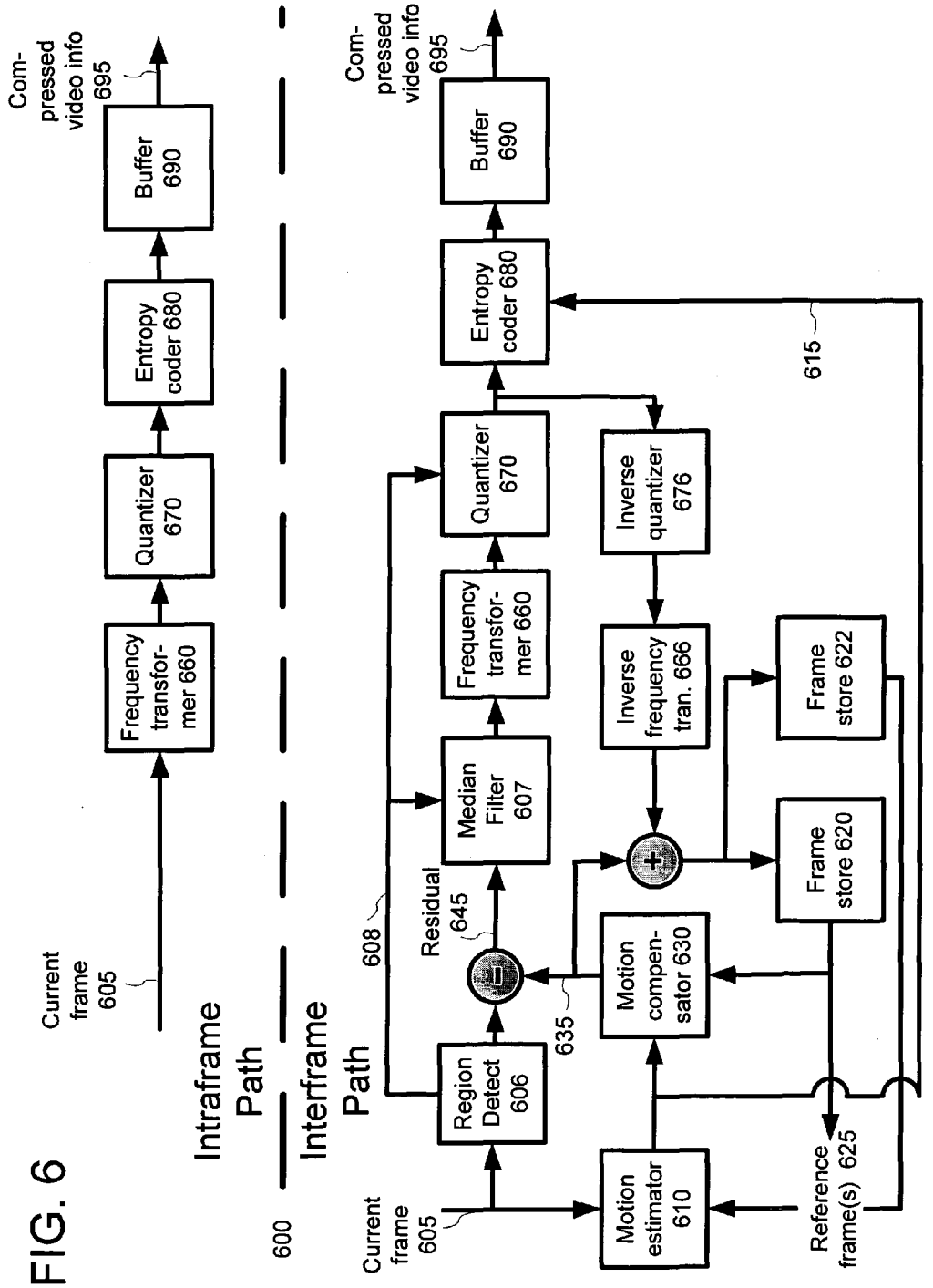

Com-pressed video info 695
Buffer 690
Entropy coder 680
Quantizer 670
Frequency transfor-mer 660
Current frame 605
Intraframe Path
Interframe Path
600
608
Residual 645
635
Region Detect 606
Median Filter 607
Motion estimator 610
Motion compen-sator 630
Inverse frequency tran. 666
Inverse quantizer 676
Frame store 620
Frame store 622
Reference frame(s) 625
615

FIG. 6

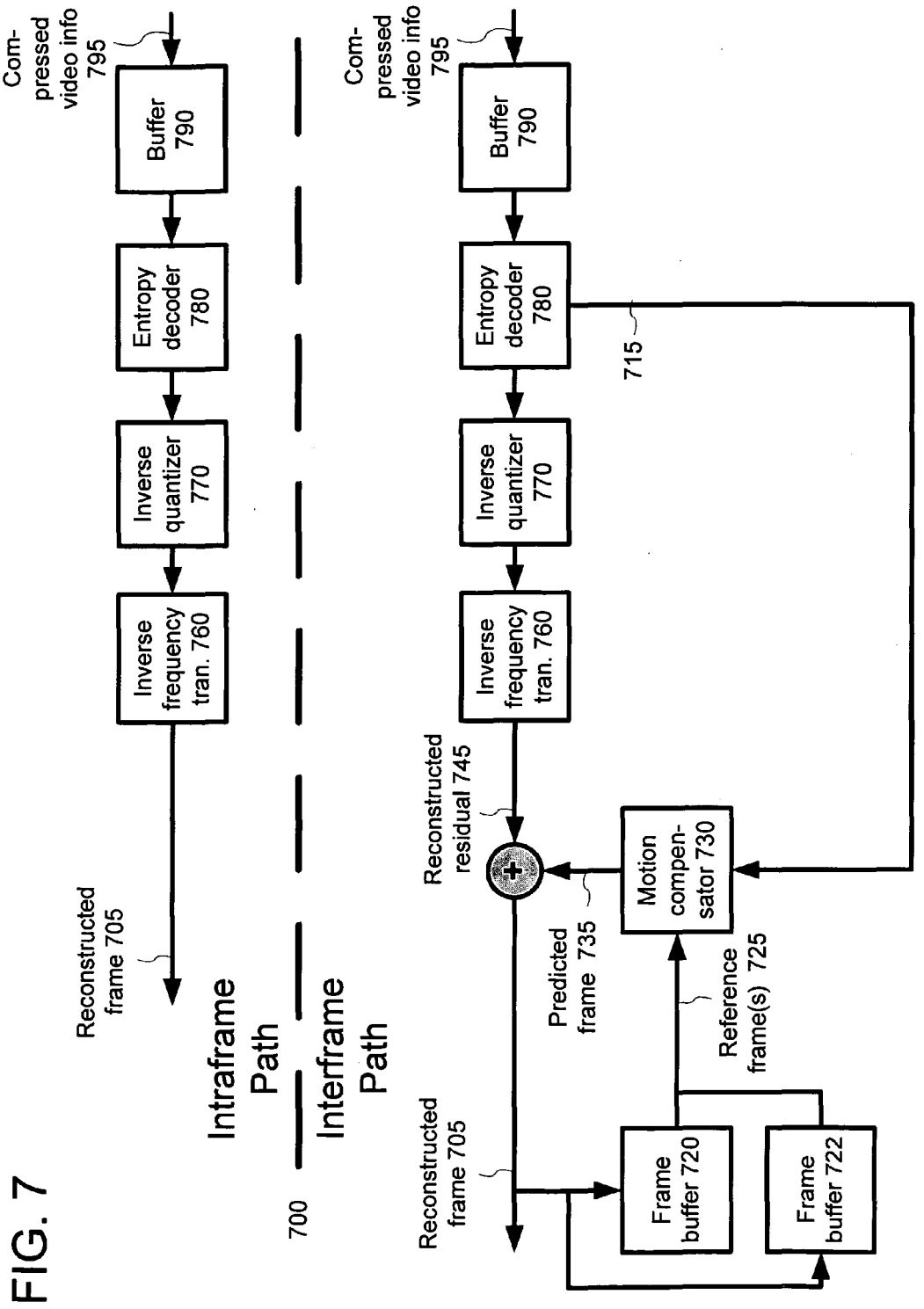

Com-
pressed
video info
795
Buffer
790
Entropy
decoder
780
Inverse
quantizer
770
Inverse
frequency
tran. 760
Reconstructed
frame 705
Intraframe
Path
Interframe
Path
700
Com-
pressed
video info
795
Buffer
790
Entropy
decoder
780
715
Inverse
quantizer
770
Inverse
frequency
tran. 760
Reconstructed
residual 745
Motion
compen-
sator 730
Predicted
frame 735
Reference
frame(s) 725
Reconstructed
frame 705
Frame
buffer 720
Frame
buffer 722

FIG. 7

FIG. 8

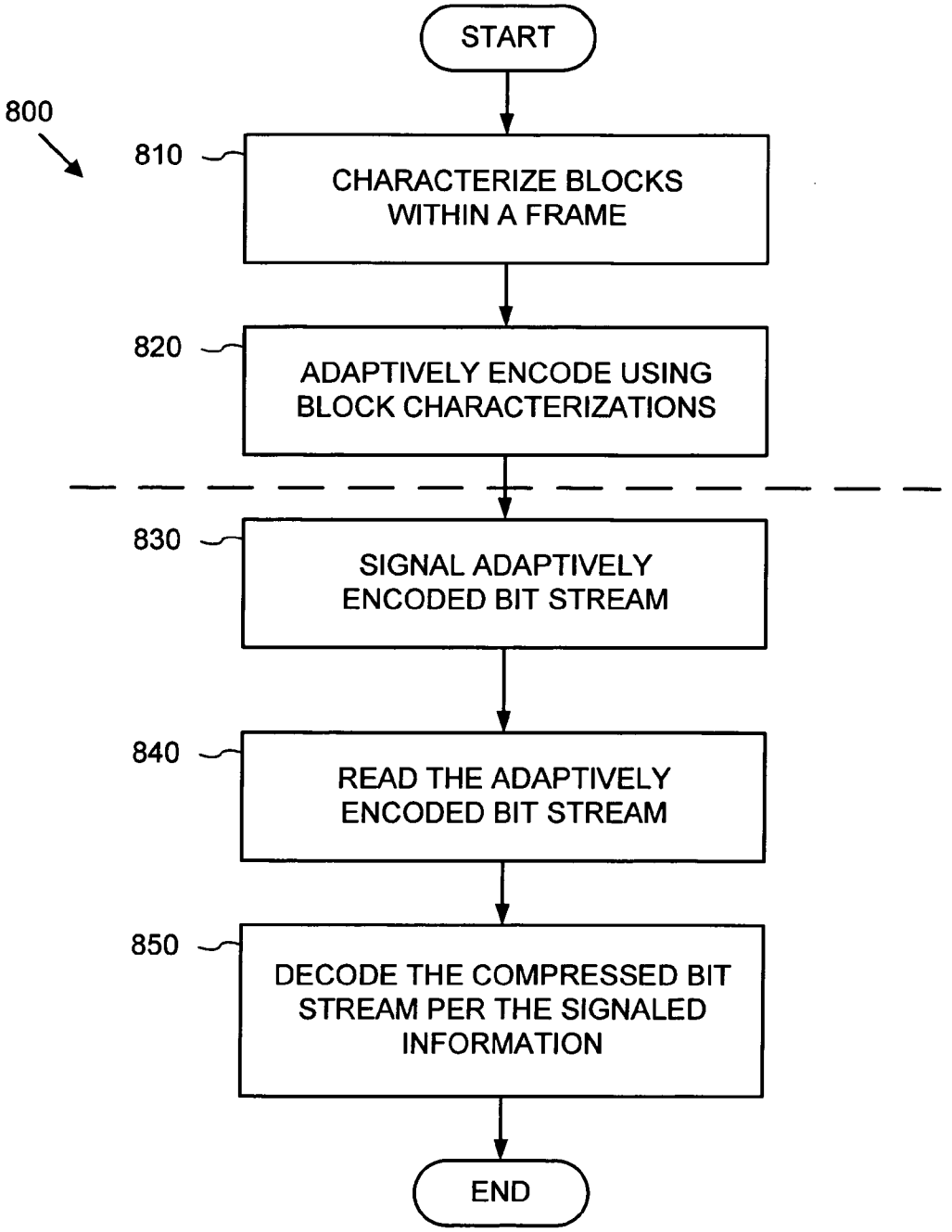

START
800
810
CHARACTERIZE BLOCKS WITHIN A FRAME
820
ADAPTIVELY ENCODE USING BLOCK CHARACTERIZATIONS
830
SIGNAL ADAPTIVELY ENCODED BIT STREAM
840
READ THE ADAPTIVELY ENCODED BIT STREAM
850
DECODE THE COMPRESSED BIT STREAM PER THE SIGNALED INFORMATION
END

FIG. 9

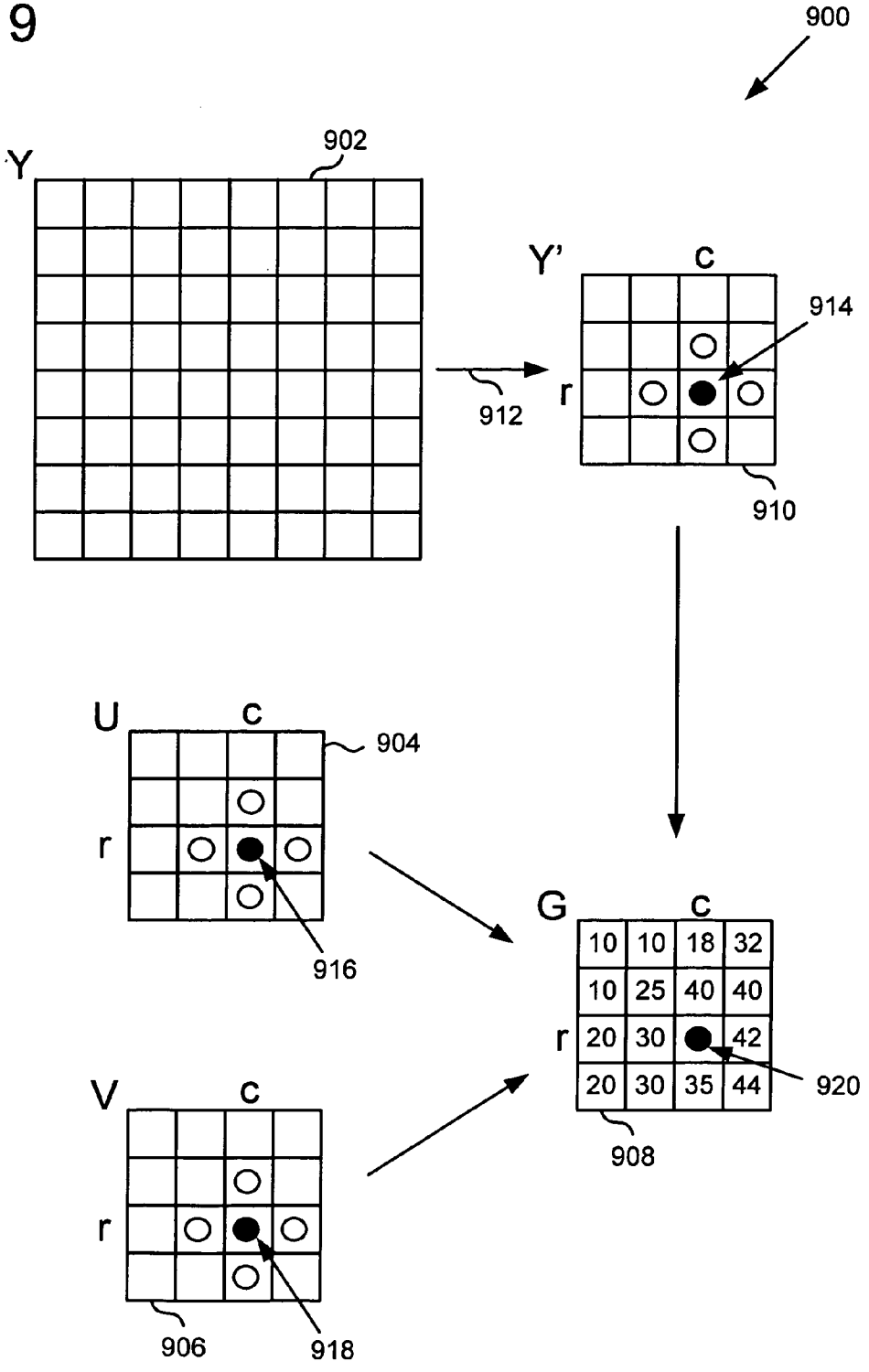

900
Y
902
Y'
c
r
912
914
910
U
c
r
904
916
V
c
r
906
918
G
c
r
10 10 18 32
10 25 40 40
20 30 42
20 30 35 44
920
908

FIG. 10

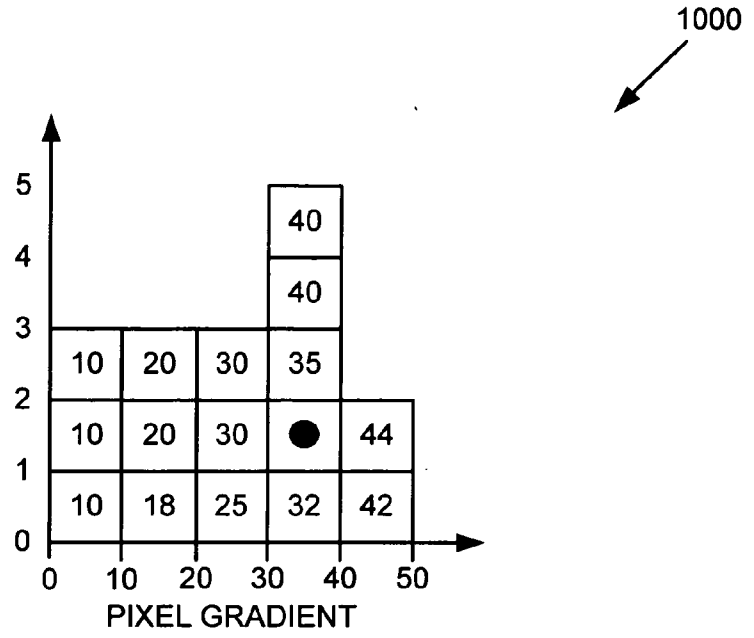

1000
5
4
3
2
1
0
40
40
10
20
30
35
10
20
30
44
10
18
25
32
42
0
10
20
30
40
50
PIXEL GRADIENT

FIG. 11

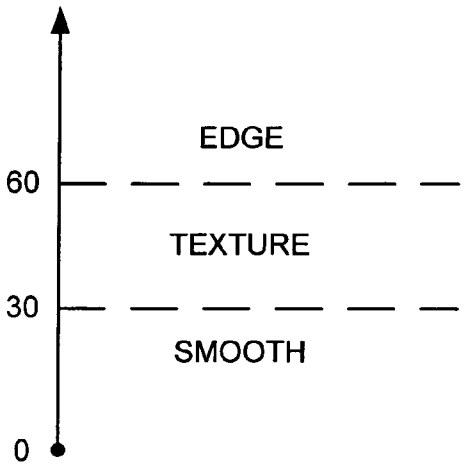

EDGE
60
TEXTURE
30
SMOOTH
0

FIG. 12

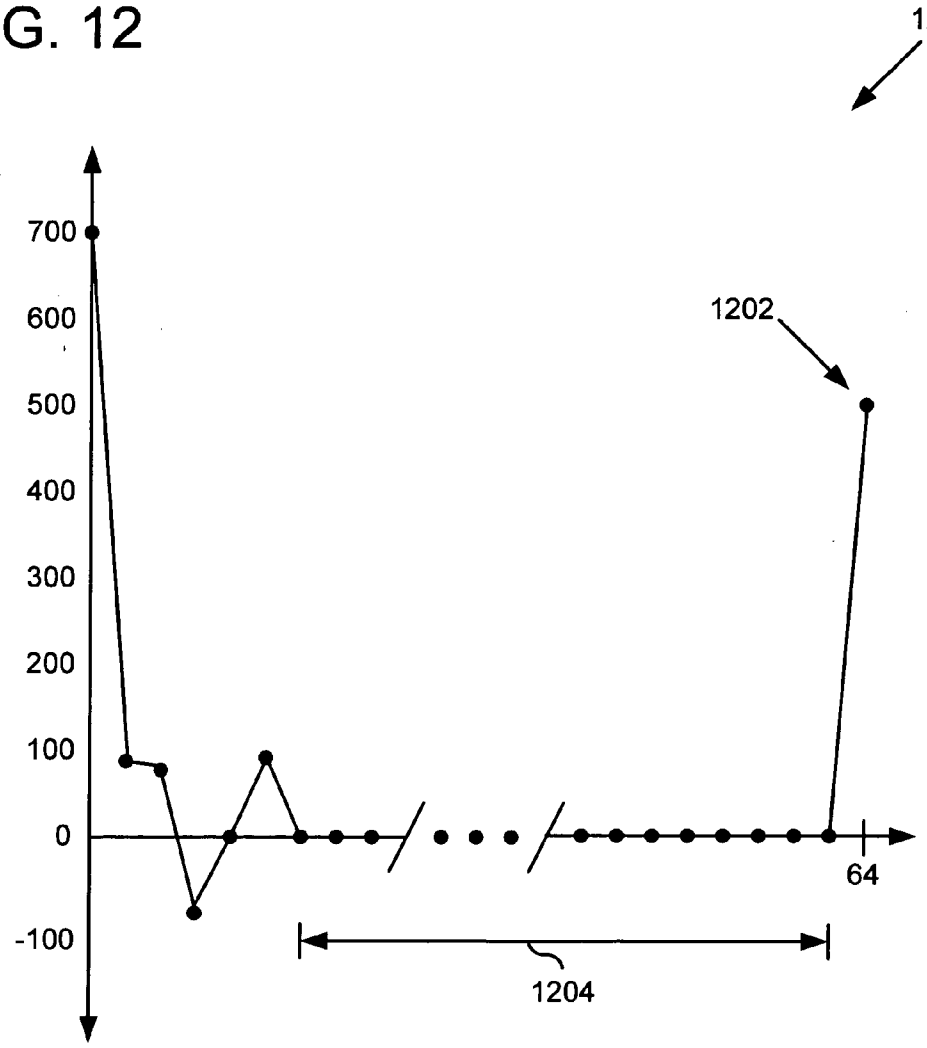


FIG. 13

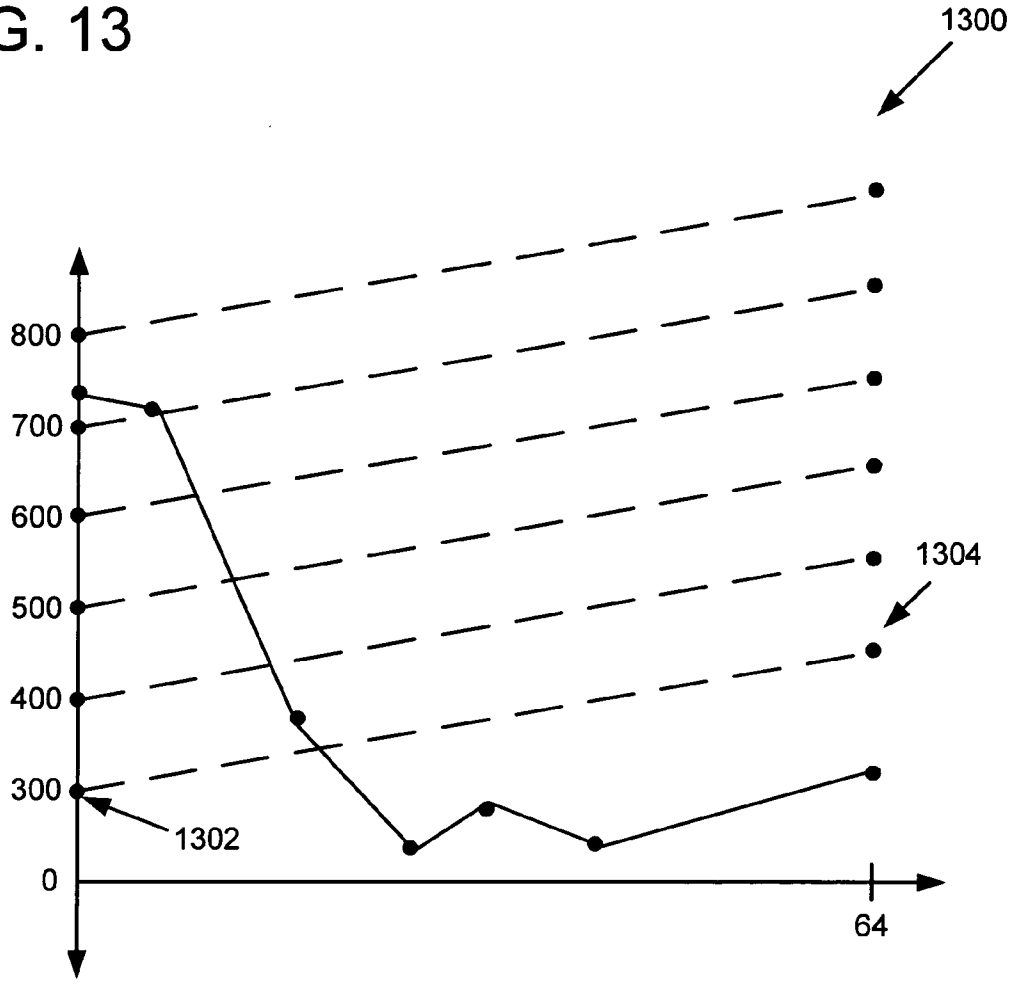
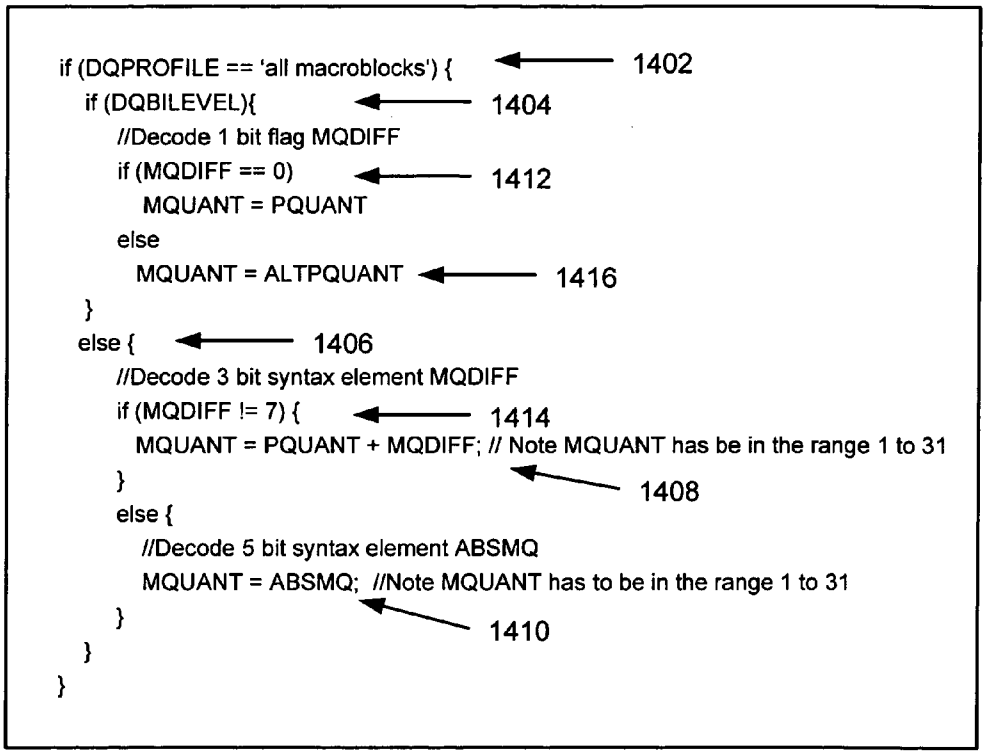


FIG. 14


1400
if (DQPROFILE == 'all macroblocks') {
1402
if (DQBILEVEL){
1404
//Decode 1 bit flag MQDIFF
if (MQDIFF == 0)
1412
MQUANT = PQUANT
else
MQUANT = ALTPQUANT
1416
}
else {
1406
//Decode 3 bit syntax element MQDIFF
if (MQDIFF != 7) {
1414
MQUANT = PQUANT + MQDIFF; // Note MQUANT has be in the range 1 to 31
}
1408
else {
//Decode 5 bit syntax element ABSMQ
MQUANT = ABSMQ; //Note MQUANT has to be in the range 1 to 31
}
1410
}
}

ADAPTIVE VIDEO ENCODING USING A PERCEPTUAL MODEL

FIELD

[0001] The described technology relates to video compression, and more specifically, to adaptive video encoding using a perceptual model.

COPYRIGHT AUTHORIZATION

[0002] A portion of the disclosure of this patent document contains material that is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

BACKGROUND

[0003] Digital video consumes large amounts of storage and transmission capacity. A typical raw digital video sequence includes 15 or 30 frames per second. Each frame can include tens or hundreds of thousands of pixels (also called pels). Each pixel represents a tiny element of the picture. In raw form, a computer commonly represents a pixel with 24 bits. Thus, the number of bits per second, or bit rate, of a typical raw digital video sequence can be 5 million bits/second or more.

[0004] Most computers and computer networks lack the resources to process raw digital video. For this reason, engineers use compression (also called coding or encoding) to reduce the bit rate of digital video. Compression can be lossless, in which quality of the video does not suffer but decreases in bit rate are limited by the complexity of the video. Or, compression can be lossy, in which quality of the video suffers but decreases in bit rate are more dramatic. Decompression reverses compression.

[0005] In general, video compression techniques include intraframe compression and interframe compression. Intraframe compression techniques compress individual frames, typically called I-frames or key frames. Interframe compression techniques compress frames with reference to preceding and/or following frames, and the compressed frame are typically called predicted frames, P-frames, or B-frames.

[0006] For example, Microsoft Corporation's Windows Media Video, Version 8 ["WMV8"] includes a video encoder and a video decoder. The WMV8 encoder uses intraframe and interframe compression, and the WMV8 decoder uses intraframe and interframe decompression.

Intraframe Compression in WMV8

[0007] **FIG. 1** illustrates prior art block-based intraframe compression **100** of a block **105** of pixels in a key frame in the WMV8 encoder. A block is a set of pixels, for example, an 8×8 arrangement of samples for pixels (just pixels, for short). The WMV8 encoder splits a key video frame into 8×8 blocks and applies an 8×8 Discrete Cosine Transform ["DCT"]**110** to individual blocks such as the block **105**. A DCT is a type of frequency transform that converts the 8×8 block of pixels (spatial information) into an 8×8 block of DCT coefficients **115**, which are frequency information. The DCT operation itself is lossless or nearly lossless. Compared to the original pixel values, however, the DCT coefficients are more efficient for the encoder to compress since most of the significant information is concentrated in low frequency coefficients (conventionally, the upper left of the block **115**) and many of the high frequency coefficients (conventionally, the lower right of the block **115**) have values of zero or close to zero.

[0008] The encoder then quantizes **120** the DCT coefficients, resulting in an 8×8 block of quantized DCT coefficients **125**. For example, the encoder applies a uniform, scalar quantization step size to each coefficient. Quantization is lossy. Since low frequency DCT coefficients tend to have higher values, quantization results in loss of precision but not complete loss of the information for the coefficients. On the other hand, since high frequency DCT coefficients tend to have values of zero or close to zero, quantization of the high frequency coefficients typically results in contiguous regions of zero values. In addition, in some cases high frequency DCT coefficients are quantized more coarsely than low frequency DCT coefficients, resulting in greater loss of precision/information for the high frequency DCT coefficients.

[0009] The encoder then prepares the 8×8 block of quantized DCT coefficients **125** for entropy encoding, which is a form of lossless compression. The exact type of entropy encoding can vary depending on whether a coefficient is a DC coefficient (lowest frequency), an AC coefficient (other frequencies) in the top row or left column, or another AC coefficient.

[0010] The encoder encodes the DC coefficient **126** as a differential from the DC coefficient **136** of a neighboring 8×8 block, which is a previously encoded neighbor (e.g., top or left) of the block being encoded. (**FIG. 1** shows a neighbor block **135** that is situated to the left of the block being encoded in the frame.) The encoder entropy encodes **140** the differential.

[0011] The entropy encoder can encode the left column or top row of AC coefficients as a differential from a corresponding column or row of the neighboring 8×8 block. **FIG. 1** shows the left column **127** of AC coefficients encoded as a differential **147** from the left column **137** of the neighboring (to the left) block **135**. The differential coding increases the chance that the differential coefficients have zero values. The remaining AC coefficients are from the block **125** of quantized DCT coefficients.

[0012] The encoder scans **150** the 8×8 block **145** of predicted, quantized AC DCT coefficients into a one-dimensional array **155** and then entropy encodes the scanned AC coefficients using a variation of run length coding **160**. The encoder selects an entropy code from one or more run/level/last tables **165** and outputs the entropy code.

Interframe Compression in WMV8

[0013] Interframe compression in the WMV8 encoder uses block-based motion compensated prediction coding followed by transform coding of the residual error. **FIGS. 2 and 3** illustrate the block-based interframe compression for a predicted frame in the WMV8 encoder. In particular, **FIG. 2** illustrates motion estimation for a predicted frame **210** and **FIG. 3** illustrates compression of a prediction residual for a motion-estimated block of a predicted frame.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1005, Page 000014

[0014] For example, the WMV8 encoder splits a predicted frame into 8×8 blocks of pixels. Groups of four 8×8 blocks form macroblocks. For each macroblock, a motion estimation process is performed. The motion estimation approximates the motion of the macroblock of pixels relative to a reference frame, for example, a previously coded, preceding frame. In **FIG. 2**, the WMV8 encoder computes a motion vector for a macroblock **215** in the predicted frame **210**. To compute the motion vector, the encoder searches in a search area **235** of a reference frame **230**. Within the search area **235**, the encoder compares the macroblock **215** from the predicted frame **210** to various candidate macroblocks in order to find a candidate macroblock that is a good match. Various prior art motion estimation techniques are described in U.S. Pat. No. 6,418,166. After the encoder finds a good matching macroblock, the encoder outputs information specifying the motion vector (entropy coded) for the matching macroblock so the decoder can find the matching macroblock during decoding. When decoding the predicted frame **210** with motion compensation, a decoder uses the motion vector to compute a prediction macroblock for the macroblock **215** using information from the reference frame **230**. The prediction for the macroblock **215** is rarely perfect, so the encoder usually encodes 8×8 blocks of pixel differences (also called the error or residual blocks) between the prediction macroblock and the macroblock **215** itself.

[0015] **FIG. 3** illustrates an example of computation and encoding of an error block **335** in the WMV8 encoder. The error block **335** is the difference between the predicted block **315** and the original current block **325**. The encoder applies a DCT **340** to the error block **335**, resulting in an 8×8 block **345** of coefficients. The encoder then quantizes **350** the DCT coefficients, resulting in an 8×8 block of quantized DCT coefficients **355**. The quantization step size is adjustable. Quantization results in loss of precision, but usually not complete loss of the information for the coefficients.

[0016] The encoder then prepares the 8×8 block **355** of quantized DCT coefficients for entropy encoding. The encoder scans **360** the 8×8 block **355** into a one-dimensional array **365** with 64 elements, such that coefficients are generally ordered from lowest frequency to highest frequency, which typically creates long runs of zero values.

[0017] The encoder entropy encodes the scanned coefficients using a variation of run length coding **370**. The encoder selects an entropy code from one or more run/level/last tables **375** and outputs the entropy code.

[0018] **FIG. 4** shows an example of a corresponding decoding process **400** for an inter-coded block. Due to the quantization of the DCT coefficients, the reconstructed block **475** is not identical to the corresponding original block. The compression is lossy.

[0019] In summary of **FIG. 4**, a decoder decodes (**410**, **420**) entropy-coded information representing a prediction residual using variable length decoding **410** with one or more run/level/last tables **415** and run length decoding **420**. The decoder inverse scans **430** a one-dimensional array **425** storing the entropy-decoded information into a two-dimensional block **435**. The decoder inverse quantizes and inverse discrete cosine transforms (together, **440**) the data, resulting in a reconstructed error block **445**. In a separate motion compensation path, the decoder computes a predicted block **465** using motion vector information **455** for displacement from a reference frame. The decoder combines **470** the predicted block **465** with the reconstructed error block **445** to form the reconstructed block **475**.

[0020] The amount of change between the original and reconstructed frame is termed the distortion and the number of bits required to code the frame is termed the rate for the frame. The amount of distortion is roughly inversely proportional to the rate. In other words, coding a frame with fewer bits (greater compression) will result in greater distortion, and vice versa.

Standards for Video Compression and Decompression

[0021] Aside from WMV8, several other versions of Windows Media Video use video compression and decompression, including Windows Media Video 9. Aside from these, several international standards relate to video compression and decompression. These standards include the Motion Picture Experts Group ["MPEG"] 1, 2, and 4 standards and the H.261, H.262, H.263, and H.264 standards from the International Telecommunication Union ["ITU"]. Like WMV8, encoders according to these products and standards use a combination of intraframe and interframe compression.

Differential Quantization

[0022] In general, differential quantization is a technique in which the amount of quantization applied to various macroblocks or blocks within a single video frame can vary. Differential quantization has been adopted or used in various standards. One benefit of differential quantization is to control bit rate at finer resolution to meet hardware requirements. One common problem that occurs when it is used, however, is that the visual quality is compromised, especially when it is used in low bit rate encoding. For example, signaling quantization parameters individually per each block in a frame of video can consume a significant proportion of bits in the compressed bitstream, especially at low bit rates, which bits could otherwise be used to encode better quality video in other ways.

[0023] U.S. Patent Application Publication No. 20050013500 describes various differential quantization techniques.

Adaptive Quantization

[0024] U.S. Patent Application Publication No. 20050036699 describes various adaptive quantization techniques. With adaptive multiple quantization, a video or other digital media codec can adaptively select among multiple quantizers to apply to transform coefficients based on content or bit rate constraints, so as to improve quality through rate-distortion optimization. The switch in quantizers can be signaled at the sequence level or frame level of the bitstream syntax, or can be implicitly specified in the syntax.

Rate Control

[0025] U.S. Patent Application Publication No. 20020186890 describes various rate and quality control techniques in which median filtering is adjusted. Based upon the buffer level, a video encoder changes the median filter kernel applied to video information. If the buffer starts to get too full, the video encoder increases the size of the kernel, which tends to smooth the video information, introduce slight blurriness, and deplete the buffer.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1005, Page 000015

[0026] Given the critical importance of video compression and decompression to digital video, it is not surprising that video compression and decompression are richly developed fields. Whatever the benefits of previous video compression and decompression techniques, however, they do not have the advantages of the following techniques and tools.

SUMMARY

[0027] The described technologies provide methods and systems for adaptive video encoding using a perceptual model. The following summary describes a few of the features described in the detailed description, but is not intended to summarize the technology.

[0028] In one example, for each of plural blocks of video data, a computer method classifies blocks of video data according to content of the blocks, and adaptively encodes the blocks based at least in part upon the block classification. In one such example, and depending at least in part on the block classification, adaptive encoding includes either adaptive median filtering, selectively dropping a last non-zero transform coefficient, adaptively re-sizing the dead zone, or adjusting a quantization step size.

[0029] In another example, blocks of video data are classified based in part on a block gradient which is determined by plural pixel gradients. In one example, a block classification such as smooth, texture, or edge is based at least in part on a block value. In one such example, a block value is determined based on an average of pixel gradients, in another example, the block value is determined based on a median of pixel gradients.

[0030] In certain examples, a pixel gradient can be quantified as a sum of differences, where the sum of differences is determined based upon both luminance and or chrominance samples. In one such example, luminance samples are downsampled for the block before the sum of differences is determined.

[0031] In certain examples, adaptive median filtering is performed on a motion compensation prediction residual and includes changing filter dimensions depending on the block classification.

[0032] In another example, selective dropping of a last non-zero coefficient further depends at least in part on a number of consecutive zero coefficients preceding the last non-zero coefficient in a zig-zag scan. In other examples, a quantization step size adjustment includes applying differential quantization depending at least in part on the block classification.

[0033] An example video encoder includes a region detector module for classifying blocks of video frames, an adaptive filter module for median filtering motion compensation error data based upon block classifications, and an adaptive quantization module for differentially quantizing the motion compensation error data based upon the block classifications.

[0034] In another example, a video encoder includes an adaptive quantization module that selectively drops isolated last non-zero transform coefficients based on the block classifications, and/or changes quantization dead-zones based on the block classifications.

[0035] Additional features and advantages will be made apparent from the following detailed description, which proceeds with reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0036] **FIG. 1** is a diagram showing block-based intraframe compression of an 8×8 block of pixels according to the prior art.

[0037] **FIG. 2** is a diagram showing motion estimation in a video encoder according to the prior art.

[0038] **FIG. 3** is a diagram showing block-based interframe compression for an 8×8 block of prediction residuals in a video encoder according to the prior art.

[0039] **FIG. 4** is a diagram showing block-based interframe decompression for an 8×8 block of prediction residuals in a video decoder according to the prior art.

[0040] **FIG. 5** is a block diagram of a suitable computing environment in which several described embodiments may be implemented.

[0041] **FIG. 6** is a block diagram of a generalized video encoder system used in several described embodiments.

[0042] **FIG. 7** is a block diagram of a generalized video decoder system used in several described embodiments.

[0043] **FIG. 8** is a flow chart of an adaptive video encoding method.

[0044] **FIG. 9** is a diagram showing computation of a pixel gradient using luminance and chrominance data.

[0045] **FIG. 10** is a histogram graph of plural pixel gradients.

[0046] **FIG. 11** is a graph of an example block value characterization framework.

[0047] **FIG. 12** is a graph of an example scan of a block of transform coefficients, showing a trailing non-zero coefficient.

[0048] **FIG. 13** is a graph of a frequency dependent dead-zone quantizer function.

[0049] **FIG. 14** is a program listing of an example method of differential quantization.

DETAILED DESCRIPTION

[0050] For purposes of illustration, the various aspects of the innovations described herein are incorporated into or used by embodiments of a video encoder and decoder (together, "codec") illustrated in **FIGS. 6-7**. In alternative embodiments, the innovations described herein can be implemented independently or in combination in the context of other digital signal compression systems, and implementations may provide adaptive video encoding using a perceptual model in compliance with any of various video codec standards. In general, the innovations described herein can be implemented in a computing device, such as illustrated in **FIG. 5**. Additionally, a video encoder incorporating the described innovations or a decoder processing output created utilizing the described innovations can be implemented in various combinations of software and/or in dedicated or programmable digital signal processing hardware in other digital signal processing devices.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1005, Page 000016

EXAMPLE COMPUTING ENVIRONMENT

[0051] **FIG. 5** illustrates a generalized example of a suitable computing environment **500** in which several of the described embodiments may be implemented. The computing environment **500** is not intended to suggest any limitation as to scope of use or functionality, as the techniques and tools may be implemented in diverse general-purpose or special-purpose computing environments.

[0052] With reference to **FIG. 5**, the computing environment **500** includes at least one processing unit **510** and memory **520**. In **FIG. 5**, this most basic configuration **530** is included within a dashed line. The processing unit **510** executes computer-executable instructions and may be a real or a virtual processor. In a multi-processing system, multiple processing units execute computer-executable instructions to increase processing power. The memory **520** may be volatile memory (e.g., registers, cache, RAM), non-volatile memory (e.g., ROM, EEPROM, flash memory, etc.), or some combination of the two. The memory **520** stores software **580** implementing a video encoder (with adaptive encoding options according to a perceptual model) or decoder.

[0053] A computing environment may have additional features. For example, the computing environment **500** includes storage **540**, one or more input devices **550**, one or more output devices **560**, and one or more communication connections **570**. An interconnection mechanism (not shown) such as a bus, controller, or network interconnects the components of the computing environment **500**. Typically, operating system software (not shown) provides an operating environment for other software executing in the computing environment **500**, and coordinates activities of the components of the computing environment **500**.

[0054] The storage **540** may be removable or non-removable, and includes magnetic disks, magnetic tapes or cassettes, CD-ROMs, DVDs, or any other medium which can be used to store information and which can be accessed within the computing environment **500**. The storage **540** stores instructions for the software **580** implementing the video encoder or decoder.

[0055] The input device(s) **550** may be a touch input device such as a keyboard, mouse, pen, or trackball, a voice input device, a scanning device, or another device that provides input to the computing environment **500**. For audio or video encoding, the input device(s) **550** may be a sound card, video card, TV tuner card, or similar device that accepts audio or video input in analog or digital form, or a CD-ROM or CD-RW that reads audio or video samples into the computing environment **500**. The output device(s) **560** may be a display, printer, speaker, CD-writer, or another device that provides output from the computing environment **500**.

[0056] The communication connection(s) **570** enable communication over a communication medium to another computing entity. The communication medium conveys information such as computer-executable instructions, audio or video input or output, or other data in a modulated data signal. A modulated data signal is a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media include wired or wireless techniques implemented with an electrical, optical, RF, infrared, acoustic, or other carrier.

[0057] The techniques and tools can be described in the general context of computer-readable media. Computer-readable media are any available media that can be accessed within a computing environment. By way of example, and not limitation, with the computing environment **500**, computer-readable media include memory **520**, storage **540**, communication media, and combinations of any of the above.

[0058] The techniques and tools can be described in the general context of computer-executable instructions, such as those included in program modules, being executed in a computing environment on a target real or virtual processor. Generally, program modules include routines, programs, libraries, objects, classes, components, data structures, etc. that perform particular tasks or implement particular abstract data types.

[0059] The functionality of the program modules may be combined or split between program modules as desired in various embodiments. Computer-executable instructions for program modules may be executed within a local or distributed computing environment.

[0060] For the sake of presentation, the detailed description uses terms like "indicate,""choose,""obtain," and "apply" to describe computer operations in a computing environment. These terms are high-level abstractions for operations performed by a computer, and should not be confused with acts performed by a human being. The actual computer operations corresponding to these terms vary depending on implementation.

EXAMPLE VIDEO ENCODER AND DECODER

[0061] **FIG. 6** is a block diagram of a generalized video encoder **600** and **FIG. 7** is a block diagram of a generalized video decoder **700**.

[0062] The relationships shown between modules within the encoder and decoder indicate the main flow of information in the encoder and decoder; other relationships are not shown for the sake of simplicity. In particular, unless indicated otherwise, **FIGS. 6 and 7** generally do not show side information indicating the encoder settings, modes, tables, etc. used for a video sequence, frame, macroblock, block, etc. Such side information is sent in the output bit stream, typically after entropy encoding of the side information. The format of the output bit stream can be a Windows Media Video format, VC-1 format, H.264/AVC format, or another format.

[0063] The encoder **600** and decoder **700** are block-based and use a 4:2:0 macroblock format. Each macroblock includes four 8×8 luminance blocks (at times treated as one 16×16 macroblock) and two 8×8 chrominance blocks. The encoder **600** and decoder **700** also can use a 4:1:1 macroblock format with each macroblock including four 8×8 luminance blocks and four 4×8 chrominance blocks. **FIGS. 6 and 7** show processing of video frames. More generally, the techniques described herein are applicable to video pictures, including progressive frames, interlaced fields, or frames that include a mix of progressive and interlaced content. Alternatively, the encoder **600** and decoder **700** are object-

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1005, Page 000017

based, use a different macroblock or block format, or perform operations on sets of pixels of different size or configuration.

[0064] Depending on implementation and the type of compression desired, modules of the encoder or decoder can be added, omitted, split into multiple modules, combined with other modules, and/or replaced with like modules. In alternative embodiments, encoder or decoders with different modules and/or other configurations of modules perform one or more of the described techniques.

[0065] **FIG. 6** is a block diagram of a general video encoder system **600**. The encoder system **600** receives a sequence of video frames including a current frame **605**, and produces compressed video information **695** as output. Particular embodiments of video encoders typically use a variation or supplemented version of the generalized encoder **600**.

[0066] The encoder system **600** compresses predicted frames and key frames. For the sake of presentation, **FIG. 6** shows a path for key frames through the encoder system **600** and a path for predicted frames. Many of the components of the encoder system **600** are used for compressing both key frames and predicted frames. The exact operations performed by those components can vary depending on the type of information being compressed.

[0067] A current frame **605** is evaluated by a region detector **606** in order to detect or classify regions of the frame according to content in those regions. For example, the region detector **606** may characterize macroblocks, blocks, or other sets of pixels of the current frame **605** as smooth, texture, or edge regions depending on their content. These characterizations are used to adapt how the regions are encoded. For example, macroblocks or blocks are evaluated to determine whether they are perceptually (1) smooth, (2) textured, or (3) represent edge or transition regions. Alternatively, the region detector **606** used other and/or additional characterizations according to the same perceptual criteria or different perceptual criteria. Moreover, although **FIG. 6** shows the region detector **606** only in the interframe path, the encoder system **600** may also include the region detector **606** in the intraframe path.

[0068] A predicted frame (also called P-frame, B-frame, or inter-coded frame) is represented in terms of prediction (or difference) from one or more reference (or anchor) frames. A prediction residual is the difference between what was predicted and the original frame. In contrast, a key frame (also called I-frame, intra-coded frame) is compressed without reference to other frames.

[0069] If the current frame **605** is a forward-predicted frame, a motion estimator **610** estimates motion of macroblocks or other sets of pixels of the current frame **605** with respect to a reference frame, which is the reconstructed previous frame **625** buffered in a frame store (e.g., frame store **620**). If the current frame **605** is a bi-directionally-predicted frame (a B-frame), a motion estimator **610** estimates motion in the current frame **605** with respect to two reconstructed reference frames. Typically, a motion estimator estimates motion in a B-frame with respect to a temporally previous reference frame and a temporally future reference frame. Accordingly, the encoder system **600** can comprise separate stores **620** and **622** for backward and forward reference frames. Various techniques are known for providing motion estimation.

[0070] The motion estimator **610** can estimate motion by pixel, ½ pixel, ¼ pixel, or other increments, and can switch the resolution of the motion estimation on a frame-by-frame basis or other basis. The resolution of the motion estimation can be the same or different horizontally and vertically. The motion estimator **610** outputs as side information motion information **615** such as motion vectors. A motion compensator **630** applies the motion information **615** to the reconstructed frame(s) **625** to form a motion-compensated current frame **635**. The prediction is rarely perfect, however, and the difference between the motion-compensated current frame **635** and the original current frame **605** is the prediction residual **645**.

[0071] In **FIG. 6**, an adaptive median filter **607** is applied to the prediction residual **645**. Optionally, the adaptive median filter **607** utilizes region characterization **608** information generated by the region detector **606** to determine a median filter configuration (e.g., kernel shape or dimensions). For example, for a texture region, a strong, 5×5 median filter is used to filter the prediction residual. A weak, 3×1 median filter is used to filter the prediction residual for a smooth region or an edge region. Optionally, a user is provided the opportunity to vary the median filter. For example, a user may select stronger median filtering. In such an example, a strong, 5×5 median filter is used for texture regions while a weak, 3×3 filter is used for smooth or edge regions. A stronger filter is applied to texture regions because, compared to smooth regions, human perception is less sensitive to distortion in the texture regions. This facilitates encoding the texture regions with fewer bits at a particular perceptual quality (compared to the amount of bits needed to encode smooth regions at the same perceptual quality). Further variations in median filters are contemplated for adaptive encoding as computing resources evolve.

[0072] A frequency transformer **660** converts the spatial domain video information into frequency domain (i.e., spectral) data. For block-based video frames, the frequency transformer **660** applies a discrete cosine transform ["DCT"] or variant of DCT to blocks of the pixel data or prediction residual data, producing blocks of DCT coefficients. Alternatively, the frequency transformer **660** applies another conventional frequency transform such as a Fourier transform or uses wavelet or subband analysis. In some embodiments, the frequency transformer **660** applies an 8×8, 8×4, 4×8, or other size frequency transforms (e.g., DCT) to prediction residuals for predicted frames.

[0073] A quantizer **670** then quantizes the blocks of spectral data coefficients. Optionally, the quantizer **670** is an adaptive quantizer that adaptively quantizes blocks based on a block characterization or value information **608** generated by the region detector **606**. If an adaptive quantizer is utilized, one or more adaptive quantization mechanisms are employed based on the information **608** generated by the region detector **606**. In one example, an isolated last non-zero transform coefficient may be dropped (reduced to zero), for example, in texture regions. As another example, an adaptive adjustment of a dead-zone size for quantization is provided during transform coefficient quantization in texture regions. As still another example, a differential quantization metric is provided during transform coefficient quantization, for example, to categorically adjust quantization in smooth vs. non-smooth regions.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1005, Page 000018

[0074] When a reconstructed current frame is needed for subsequent motion estimation/compensation, an inverse quantizer **676** performs inverse quantization on the quantized spectral data coefficients. An inverse frequency transformer **666** then performs the inverse of the operations of the frequency transformer **660**, producing a reconstructed prediction residual (for a predicted frame) or a reconstructed key frame.

[0075] If the current frame **605** was a key frame, the reconstructed key frame is taken as the reconstructed current frame (not shown). If the current frame **605** was a predicted frame, the reconstructed prediction residual is added to the motion-compensated current frame **635** to form the reconstructed current frame. If desirable, a frame store (e.g., frame store **620**) buffers the reconstructed current frame for use in predicting another frame. In some embodiments, the encoder applies a deblocking filter to the reconstructed frame to adaptively smooth discontinuities in the blocks of the frame.

[0076] The entropy coder **680** compresses the output of the quantizer **670** as well as certain side information (e.g., motion information **615** and/or quantization step sizes). Typical entropy coding techniques include arithmetic coding, differential coding, Huffman coding, run length coding, LZ coding, dictionary coding, and combinations of the above. The entropy coder **680** typically uses different coding techniques for different kinds of information (e.g., DC coefficients, AC coefficients, different kinds of side information), and can choose from among multiple code tables within a particular coding technique.

[0077] The entropy coder **680** puts compressed video information **695** in the buffer **690**. A buffer level indicator is fed back to bit rate adaptive modules.

[0078] The compressed video information **695** is depleted from the buffer **690** at a constant or relatively constant bit rate and stored for subsequent streaming at that bit rate. Therefore, the level of the buffer **690** is primarily a function of the entropy of the filtered, quantized video information, which affects the efficiency of the entropy coding. Alternatively, the encoder system **600** streams compressed video information immediately following compression, and the level of the buffer **690** also depends on the rate at which information is depleted from the buffer **690** for transmission.

[0079] Before or after the buffer **690**, the compressed video information **695** can be channel coded for transmission over the network. The channel coding can apply error detection and correction data to the compressed video information **695**.

[0080] **FIG. 7** is a block diagram of a general video decoder system **700**. The decoder system **700** receives information **795** for a compressed sequence of video frames and produces output including a reconstructed frame **705**. Particular embodiments of video decoders typically use a variation or supplemented version of the generalized decoder **700**.

[0081] The decoder system **700** decompresses predicted frames and key frames. For the sake of presentation, **FIG. 7** shows a path for key frames through the decoder system **700** and a path for predicted frames. Many of the components of the decoder system **700** are used for decompressing both key frames and predicted frames. The exact operations performed by those components can vary depending on the type of information being decompressed.

[0082] A buffer **790** receives the information **795** for the compressed video sequence and makes the received information available to the entropy decoder **780**. The buffer **790** typically receives the information at a rate that is fairly constant over time, and includes a jitter buffer to smooth short-term variations in bandwidth or transmission. The buffer **790** can include a playback buffer and other buffers as well. Alternatively, the buffer **790** receives information at a varying rate. Before or after the buffer **790**, the compressed video information can be channel decoded and processed for error detection and correction.

[0083] The entropy decoder **780** entropy decodes entropy-coded quantized data as well as entropy-coded side information (e.g., motion information **715** and/or quantization step sizes), typically applying the inverse of the entropy encoding performed in the encoder. Entropy decoding techniques include arithmetic decoding, differential decoding, Huffman decoding, run length decoding, LZ decoding, dictionary decoding, and combinations of the above. The entropy decoder **780** frequently uses different decoding techniques for different kinds of information (e.g., DC coefficients, AC coefficients, different kinds of side information), and can choose from among multiple code tables within a particular decoding technique.

[0084] A motion compensator **730** applies motion information **715** to one or more reference frames **725** to form a prediction **735** of the frame **705** being reconstructed. For example, the motion compensator **730** uses a macroblock motion vector to find a macroblock in a reference frame **725**. A frame buffer (e.g., frame buffer **720**) stores previously reconstructed frames for use as reference frames. Typically, B-frames have more than one reference frame (e.g., a temporally previous reference frame and a temporally future reference frame). Accordingly, the decoder system **700** can comprise separate frame buffers **720** and **722** for backward and forward reference frames.

[0085] The motion compensator **730** can compensate for motion at pixel, ½ pixel, ¼ pixel, or other increments, and can switch the resolution of the motion compensation on a frame-by-frame basis or other basis. The resolution of the motion compensation can be the same or different horizontally and vertically. Alternatively, a motion compensator applies another type of motion compensation. The prediction by the motion compensator is rarely perfect, so the decoder **700** also reconstructs prediction residuals.

[0086] When the decoder needs a reconstructed frame for subsequent motion compensation, a frame buffer (e.g., frame buffer **720**) buffers the reconstructed frame for use in predicting another frame. In some embodiments, the decoder applies a deblocking filter to the reconstructed frame to adaptively smooth discontinuities in the blocks of the frame.

[0087] An inverse quantizer **770** inverse quantizes entropy-decoded data. In general, the inverse quantizer applies uniform, scalar inverse quantization to the entropy-decoded data with a step-size that varies on a frame-by-frame basis or other basis, where the quantization step size information is signaled in the bitstream.

[0088] An inverse frequency transformer **760** converts the quantized, frequency domain data into spatial domain video

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1005, Page 000019

information. For block-based video frames, the inverse frequency transformer **760** applies an inverse DCT ["IDCT"] or variant of IDCT to blocks of the DCT coefficients, producing pixel data or prediction residual data for key frames or predicted frames, respectively. Alternatively, the frequency transformer **760** applies another conventional inverse frequency transform such as a Fourier transform or uses wavelet or subband synthesis. In some embodiments, the inverse frequency transformer **760** applies an 8×8, 8×4, 4×8, or other size inverse frequency transforms (e.g., IDCT) to prediction residuals for predicted frames.

EXAMPLE PERCEPTUAL MODEL OVERVIEW

[**0089**] Perceptual importance of a local region within a video frame changes spatially due to variance of detail complexity over the frame. Human eyes are more sensitive to distortions in smooth regions and edge regions (e.g., object boundaries) in video frames than to distortions in texture regions. Existing video codecs fail to adequately exploit this aspect of the human perception. Using the described techniques, smooth regions and edge regions can be coded at higher quality for a given bit rate by allocating more bits (introducing less distortion) to the smooth regions and edge regions and allocating fewer bits (reducing quality, introducing more distortion) to texture regions. One goal is to improve the overall perceptual quality of the coded video at the same bit rate compared to the previous codec. Various video codec features are described that adaptively allocate bits across blocks or macroblocks of a video frame according to their perceptual characterization, thereby allocating bits differentially buy providing more uniform perceptual quality of the coded video. Often, it is desirable to improve the perceived quality at the same bitrate compared with existing codecs. At other times, it is desirable to maintain a present quality at a reduced bitrate. Various aspects of these features are encoder improvements that may not require any change in the bitstream. The provided features are applicable to both the current encoder profiles (VC-1 and Microsoft Research video encoders) as well as previous profiles (WMV Advanced, WMV3, etc.).

[**0090**] Various features are discussed that may be used in adaptive encoding that employs a perceptual model for video codec. Such features are generally described as (1) detection of different regions (e.g., smooth, edge, texture, etc.), (2) adaptive median filtering, (3) adaptive isolated transform coefficient dropping, (4) adaptive dead zone sizing during quantization, and (5) adaptive differential quantization.

[**0091**] Methods and systems are described for detecting or characterizing different regions of a video frame. Using this characterization information, the video encoding is adapted. For example, more bits are spent on smooth regions and edge regions, while fewer bits are spent on texture regions. These distinctions allow controlling perceivable distortion—when distortion must be introduced to lower bit rate, more distortion is added where it is less easily perceived and less distortion is added where it is more easily perceived.

[**0092**] Improved features include fast detection of different region types using a gradient comparison. For example, variations between adjacent pixels in a block are determined by comparing luminance and/or chrominance sample values. Determined differences between adjacent pixels are used to determine the extent of intensity variance within a block, macroblock, or other video frame region (hereafter "block"). Various metrics can be used to determine the intensity variance between pixel data within a block in order to determine how a human would perceive the block, for example, as smooth or textured.

[**0093**] A median intensity variance can be computed for a block in order to characterize the block. For example, a histogram can be used to determine a median variance for a block. Characterized blocks can be labeled, for example, as smooth, edge, or texture. Alternatively, other and/or additional labels are used. A median variance can also be used as input to a more precise function, with or without requiring labels, where the more precise function provides finer grained control encoding adaptation based on a determined block median.

[**0094**] One or more of the encoding variations are provided based on block characterization. The variations may be used together or separately, and they may be used in combination with other and/or additional encoding variations. In a first variation, an adaptive median filtering is provided based on block characterization. For example, increased median filtering is provided for texture regions. In a second variation, an isolated last non-zero transform coefficient may be dropped, for example, in texture regions. In a third variation, an adaptive adjustment of a dead zone is provided during transform coefficient quantization if desirable in texture regions. In a fourth variation, a differential quantization metric is applied during transform coefficient quantization, for example, depending on whether a region is smooth or non-smooth.

[**0095**] Optionally, values or distinctions used to characterize blocks or to alter various aspects of adaptive encoding may depend on one or more user settings. For example, a user selects an encoding scenario, wizard profile, or other high-level description of an encoding path, and values associated with the scenario/profile/description are set for one or more of the region detection, adaptive median filtering, coefficient dropping conditions, dead zone variance, or adaptive differential quantization. Or, the value for one or more of these options is directly set by a user through a user interface. Alternatively, one or more of these options has a value set when an encoder is installed in a computer system or device, depending on the system/device profile. In another example, one or more of these options is set adaptively by the encoder based on how much computational power is available.

[**0096**] Various aspects of these features are described throughout this disclosure. However, neither the titles of the features, nor the placement within paragraphs of the description of various aspects of features, are meant to limit how various aspects are used or combined with aspects of other features. After reading this disclosure, one of ordinary skill in the art will appreciate that the description proceeds with titles and examples in order to instruct the reader, and that once the concepts are grasped, aspects of the features are applied in practice with no such pedagogical limitations.

EXAMPLE ADAPTIVE VIDEO CODING METHOD

[**0097**] **FIG. 8** is a flow chart of an example method of adaptive video encoding using a perceptual model. One

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1005, Page 000020

aspect of adaptive encoding in the method is to provide a characterization for the level of detail in a block. For example, blocks representing relatively smooth regions in a video frame, such as a blue sky, may be characterized as less acceptable for introducing distortion. Over simplification of detail representation (e.g., lossy compression of video data) in an encoding scheme can be more easily perceived by humans in such smooth regions. In contrast, texture regions can often be lossy compressed relatively further without human perception of the added distortion. For example, a forest of trees may be characterized as a texture region that allows greater simplification of detail representation without human perception of the distortion introduced.

[0098] With reference to **FIG. 8**, the video encoder **600** described above implements adaptive encoding techniques in a process **800** that characterizes blocks (e.g., blocks of macroblocks, macroblocks, other regions, etc.) of a video frame (e.g., smooth region, edge region, texture region, etc.) and adapts one or more encoding techniques according to the characterization. The described techniques provided adaptive encoding within a frame, such as on a block, macroblock (MB) or a group of MB basis (e.g., a region, or a "block" generally). The techniques use information to classify different blocks of the image and to encode them accordingly. More particularly, the video encoder **600** characterizes blocks to classify the importance of content within a frame.

[0099] At **810**, the video encoder **600** characterizes a block based on the determined intensity variance within the block of the video frame. For example, the sum of the differences between a pixel and its adjacent pixels is computed for the pixels in the block. This sum of differences value indicates an extend of intensity variance between a pixel and its surrounding pixels. For example, surrounding pixels comprise two or more other pixels adjacent to or nearly adjacent to a pixel, such as above or below, to the left or right, or diagonal to a pixel. The difference between a pixel's intensity and the intensities of its surrounding pixels is computed based on differences in luminance and/or chrominance data. A computed difference value is assigned to the pixel (e.g., a pixel gradient). A difference value is computed in this way for plural or all pixels in a block (e.g., a block gradient), or for some sub-sampled set thereof. The difference values assigned to pixels in a block are evaluated to determine a characterization for the block (e.g., a block value). In one example, the pixel gradients for pixels in a block are evaluated to determine a median difference value for the block gradient (e.g., a block median). Optionally, the block value is used to classify a block (e.g., smooth, edge, or texture; texture or non-texture; smooth or non-smooth; etc.). Thus, intensity differences between pixels within a block provide a measure of intensity variance for a block, macroblock, or video frame region (generally, "block").

[0100] A block median is not required to determine a block value. An intensity variance or block characterization may also be based on an average value for difference values assigned to pixels in the block (e.g., a block average). The block median or average (e.g., a block value) can be used to classify the block and/or can be used as input to a finer grained control function. The characterization or control function adaptively varies one or more aspects of encoding.

[0101] Alternatively, instead of computing an intensity variance to characterize a block, the encoder **600** uses another metric to vary adaptive encoding according to one or more of the adaptive encoding variations described herein.

[0102] At **820**, the encoder adaptively encodes blocks of the video frame based on the block characterization. One or more encoding features are adapted and provided based on the block characterization. In one implementation, an adaptive median filtering is provided based on block characterization. For example, increased median filtering is provided for texture regions. An isolated last non-zero transform coefficient may be dropped, for example, in texture regions. An adaptive adjustment of a dead zone is provided during transform coefficient quantization if desirable in texture regions. A differential quantization pattern is applied during transform coefficient quantization, for example, in smooth regions vs. non-smooth regions, and the differential quantization information is signaled in an output bit stream. In common encoding scenarios, regions classified as texture are quantized more strongly, which saves bits.

[0103] Although **FIG. 8** shows the characterizing stage **810** preceding the adaptive encoding stage **820** for multiple blocks in a frame, these stages may also occur iteratively on a block-by-block basis in the frame or be ordered on some other basis.

[0104] At **830**, the encoder signals the adaptively encoded bit stream. When differential quantization is used by the encoder to encode based on block characterization, for example, the video encoder **600** encodes information in the compressed bit stream using a signaling scheme described below for signaling the differential quantization to the video decoder **700**.

[0105] At **840**, a corresponding video decoder **700** reads the adaptively encoded bit stream. For example, the video decoder **700** reads the signaled differential quantization information. At **850**, the decoder **700** decodes the compressed bit stream, for example, dequantizing blocks according to signaled differential quantization information.

EXAMPLE REGION DETECTION OR CLASSIFICATION

[0106] **FIG. 9** is a diagram showing block-based **900** region detection or characterization using luminance and/or chrominance data. The luminance block "Y" shown **902** is an 8×8 block. Although not required, in this example, chrominance blocks **904**, **906** for the pixel block are also used in computing a gradient block **908**. Although not required, as shown in this example, the luminance block **902** is downsampled **912** by a factor of 2 horizontally and vertically (e.g., by simple averaging of pairs of pixels) to create a luminance block **910** that matches the 4×4 dimensions of the chrominance blocks.

[0107] As shown in the downsampled luminance block **910**, a pixel **914** is compared to four pixels near it, and a sum of the difference between the pixel **914** and its surrounding vertical and horizontal pixels is computed. In this example, the pixel **914** is located at position Y'(r, c). Thus an equation provides the sum of the differences for luminance intensity values for this pixel **914** as compared to its surrounding pixels:

$$L_1(r, c)=[|Y(r, c)-Y(r, c-1)|+|Y(r, c)-Y(r-1, c)/+/Y(r, c)-Y(r, c+1)|+|Y(r, c)-Y(r+1, c)|]/4$$

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1005, Page 000021

[0108] As shown, Y'(r, c) is the luminance component of the pixel **914** at row r and column c in the downsampled block Y'. The output of the equation L_1 provides an indication of how the pixel **914** differs in luminance intensity from its pixel neighbors within the block Y'. This luminance intensity difference measurement is an example of a pixel gradient.

[0109] Optionally, chrominance data **904**, **906** may be considered alone instead of luminance data, or may be considered together with luminance data to determine intensity differences for region detection or classification. Thus, an equation for the sum of the differences for luminance intensity values and chrominance intensity values for corresponding pixels **914**, **916**, **918** in luminance and chrominance blocks for the pixel block can be represented as the average of the differences in intensity values of the surrounding pixels as shown in the following equation:

$$G_1(r,c)=\{[|Y(r,c)-Y(r,c-1)|+|Y(r,c)-Y(r-1,c)|+|Y(r,c)-Y(r,c+1)|+|Y(r,c)-Y(r+1,c)|]+[|U(r,c)-U(r,c-1)|+|U(r,c)-U(r-1,c)|+|U(r,c)-U(r,c+1)|+|U(r,c)-U(r+1,c)|]+[|V(r,c)-V(r,c-1)|+|V(r,c)-V(r-1,c)|+|V(r,c)-V(r,c+1)|+|V(r,c)-V(r+1,c)|]\}/12$$

[0110] The output of the equation $G_1(r, c)$ is an example of a pixel gradient for the pixel located at (r, c), and the pixel gradient provides an indication of how the pixel **914**, **916**, **918** differs in luminance and chrominance intensity from its surrounding pixel neighbors. In this example, the pixel gradient value $G_1(r, c)$ is based only on pixels that are immediately vertical or horizontal, but does not consider other pixels in the neighborhood. It is contemplated that other pixel data may also be considered in creation of a pixel gradient in other variations of region detection or classification. For example, diagonal pixels could be considered as part of, or instead of the provided arrangement. Or, intensity differences across a longer stretch (e.g., 2 or 3 pixels) could be considered.

[0111] Notice also, that the output of the preceding equation provides only an indication of how a single pixel differs from its neighbors in luminance and chrominance intensity. In order to characterize the intensity variance for an entire block, the same analysis is performed on plural or all pixels within the block. In one such example, a block **908** of pixel gradients is created that contains plural pixel gradients, and a block gradient is derived therefrom. A block gradient provides a more complete indication of how pixels within a block differ in intensity from each other. As noted, computing a pixel gradient or a gradient block may include luminance comparisons alone, chrominance comparisons alone, or both luminance and chrominance comparisons together.

[0112] If desirable, the above equation $G_1(r, c)$ may be varied to account for missing block boundary values. For example, pixels outside the block may be extrapolated or assumed to be the same as other adjacent pixel data within the block when adapting the equation $G_1(r, c)$ to account for boundary values. Or, the denominator of the equations may be reduced and surrounding pixels in certain directions ignored in the comparisons, for example, where those surrounding pixels are outside of the block. As shown, a block **908** of pixel gradients may provide pixel gradient data for all pixels in the block. Or, a block **908** of pixel gradients may include pixel gradient data for less than all pixels in the block.

EXAMPLE SELECTION OF BLOCK VALUE

[0113] **FIG. 10** is a histogram graph of plural pixel gradients in the block **908** of **FIG. 9**. More specifically, the histogram **1000** provides a visualization of how a block is characterized or valued. In this example, there are eight pixel gradient values below **30**, and eight pixel gradient values above **30**. Thus, a median value for this block gradient is **30**. (For an even number of candidates, the median can be computed as the average of the two middle candidate values, or as one or the other of the two middle candidate values.) The median value may be used to determine a label for the block as smooth, texture, or edge.

[0114] Of course, other metrics may be used to characterize or value blocks once the pixel gradients or blocks of pixel gradients are obtained. For example, blocks may be characterized according to an average of pixel gradient values. Finally, once a block value is assigned it can be used in a characterization scheme (e.g., smooth or non-smooth; smooth, texture, edge; etc.) or in a finer grained control function, in order to determine how the block value is used in an adaptive encoding strategy.

[0115] Thus, a block median may be selected by ordering plural pixel gradients and selecting a median gradient value from the ordered values. For example, a sample set of pixel gradients within a block, such as {**10**, **14**, **28**, **36**, **38**}, will have a block value assigned equal to the median pixel gradient in the set, or 28. In another example, a block value is determined based on the average gradient in the set, or 25.2. Of course, the set may be obtained from a complete block gradient, or a subset thereof.

EXAMPLE USE OF REGION CLASSIFICATION INFORMATION

[0116] As previously discussed, a block is valued according to some region detection or classification scheme. For example, a region type depends on a gradient index histogram. One or more various encoding mechanisms are adapted using the block characterization or value.

[0117] **FIG. 11** is a graph of an example block value characterization, continuing the example of **FIGS. 9 and 10**. As shown, a block value in the range from 0 up to and including 30 will be characterized as a smooth block. A block value in the range of greater than 30 but less than or equal to 60 will be characterized as a texture graph, and a block value greater than 60 will be assigned an edge characterization.

[0118] Table 1 provides an adaptive coding scheme using block characterizations as described with reference to **FIG. 11**. As shown, blocks will be coded distinctly in terms of one or more adaptive features. For example, four features are proposed as follows:

TABLE 1

Characterization	Median Filter	Drop-Last-Coef	Dead Zone	D-Quant
Smooth	Weak	No	No	Lower QP
Edge	Weak	No	No	Higher QP
Texture	Strong	Yes	Yes	Higher QP

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1005, Page 000022

[0119] In Table 1, the term QP denotes a quantization size. The various adaptive features are discussed throughout this document and will be further discussed below. Alternatively, an encoder uses another mapping of adaptive feature decisions to block characterizations.

EXAMPLE ADAPTIVE MEDIAN FILTERING

[0120] Optionally, an encoder provides a median filter in different sizes or strengths based on different block characterizations. In the encoder **600** of **FIG. 6**, an adaptive median filter **607** is applied to the prediction residual **645**, where the region detector provides information **608** comprising a block classification used to determine an adaptive filter.

[0121] As known in the art, simple translational motion estimation and compensation involve prediction with a motion vector which indicates how to move a set of pixels or a block. The movement of the set of pixels or block is expressed as a displacement of pixels. The simple displacement of a set of pixels or block often causes a prediction error. Thus, a bit stream often includes an error block for mitigating this prediction error. When motion compensation works, encoding a block of prediction error or residual data requires fewer bits than encoding the same area with intraframe compression, because the prediction error often includes less energy.

[0122] Even when motion compensation succeeds for a block, there may be information content in the prediction residual for the block. The information content appears as isolated, "salt and pepper" noise values among areas with zero residual values for motion-compensated prediction has mostly succeeded, or the information content may appear as edges where a new part of an object has been exposed that was not in a reference frame. A median filter tends to smooth residual information (removing isolated noise values) while at the same time preserving useful detail (such as edges) in the residual information. The smoothing adds redundancy, which makes subsequent lossless compression more efficient. To control how much the video information is filtered, the video encoder changes the median filter kernel applied to the residual information based upon the block characterizations.

[0123] For example, for a texture region, a strong 5×5 median filter is used to filter the prediction residual. A weak 3×1 median filter is used to filter the prediction residual for a smooth region or an edge region. In the encoder **600** of **FIG. 6**, the median filter is applied to the motion compensation errors block before the errors block **335** is encoded using the frequency transformer **660** and quantizer **670**. Alternatively, other and/or additional median filter kernel shapes and sizes are available for the adaptive median filtering.

[0124] Optionally, a user is provided the opportunity to vary the median filter. For example, a user may select stronger median filtering. In one such implementation, a strong 5×5 median filter is used for texture regions while a weak 3×3 filter is used for smooth or edge regions.

[0125] A stronger filter is applied to texture regions because, when distortion must be introduced to meet coding constraints, human perception is less sensitive to distortion introduced in the texture regions, whereas human perception is more sensitive to distortion introduced in smooth regions. Further variations in median filters are contemplated for adaptive encoding as computing resources increase.

EXAMPLE ADAPTIVE ISOLATED TRANSFORM COEFFICIENT DROPPING

[0126] Before a block of transform coefficients is entropy coded using run length coding, run level coding, or many other types of entropy coding, the block of transform coefficients is scanned into a one-dimensional array according to a scan pattern (often called a zigzag pattern). After quantization of the transform coefficients (and even before quantization to some extent), one common pattern in the transform coefficients is to have non-zero values at the start of the array (corresponding to the low frequency coefficients conventionally shown at the top left of a two-dimensional block), followed by mostly zero values, followed by an isolated non-zero transform coefficient (corresponding to a high frequency coefficient conventionally shown at the bottom right of the two-dimensional block). For example, for an 8×8 block of quantized transform coefficients (64 values), the resulting scan often has only a few non-zero coefficients at the beginning followed by many zeros, and then possibly just a few more non-zero coefficients at the end (e.g., 700, 90, 90, −89, 0, 100, 0, 0, 0, . . . , 500).

[0127] **FIG. 12** is a graph of an example scan of a block of transform coefficients. As shown in the example graph **1200**, an isolated last non-zero coefficient **1202** is defined as the last non-zero transform coefficient of each block that has a large run value **1204** for zero-value coefficients (e.g.; run value greater than some threshold), after zigzag scanning. The cost of entropy coding such an isolated, trailing non-zero coefficient **1202** is often high, relative to the benefits of coding the coefficient **1202** in term of quality improvement.

[0128] Optionally, an encoder provides an adaptive quantizer **670** or other module, that drops the last non-zero coefficient(s) in the scan, based on a characterization of the perceptual region. For example, the last non-zero coefficient(s) may be dropped in texture or edge regions, or just for texture regions.

[0129] In another example, if a block is characterized as a texture block, and there is a certain number of zeros before a last non-zero coefficient (e.g., 40 zeros, 50 zeros, 60 zeros, etc.), the last non-zero coefficient is dropped (changed to zero). The number of zeros is contemplated to vary depending on the desired likelihood for dropping trailing coefficients, evolving computing efficiency, a desired level of detail, and/or a user indication of desired detail.

EXAMPLE ADAPTIVE DEAD-ZONE DURING QUANTIZATION

[0130] Optionally, adaptive sizing of a dead zone is used during quantizing the transform coefficients. A dead-zone is the region within which zero output is generated for some transform coefficient during quantization.

[0131] Table 2 is an example of how an encoder adaptively quantizes transform coefficients by changing the size of the dead zone.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1005, Page 000023

TABLE 2

Coeff Value	Quantized Smooth	Quantized Edge or Texture
$0 < x \leqq 100$	0	0
$100 < x \leqq 200$	1	0
$200 < x \leqq 300$	2	0
$300 < x \leqq 400$	3	0
$400 < x \leqq 500$	4	1
$500 < x \leqq 600$	5	2
$600 < x \leqq 700$	6	3
$700 < x \leqq 800$	7	4
$800 < x \leqq 900$	8	5

[0132] As shown in Table 2, a larger dead-zone is used for edge regions or texture regions. For a smooth region, a transform coefficient with a magnitude of anything less than or equal to 100 is assigned a quantized value of 0, and typically will be reconstructed as 0. For a non-smooth region, a transform coefficient with a magnitude of anything less than or equal to 400 is assigned a quantized value of 0, and typically will be reconstructed as 0. As a result, the non-smooth regions are more likely to include redundancy (due to introduction of distortion), and fewer bits will be required in lossless compression to represent the non-smooth regions as compared to smooth regions.

[0133] In the encoder of **FIG. 6**, the region detector **606** provides information **608** that an adaptive quantizer **670** uses to adaptively quantize the transform coefficients according to a block characterization. For example, a larger dead-zone may be applied to texture or edge regions, and a smaller dead zone for smooth regions.

[0134] In another example, an encoder adaptively quantizes transform coefficients in a frequency dependent manner (or, in practice, in a manner that depends on order in a one-dimensional array following zigzag scanning). **FIG. 13** is a graph of a frequency dependent dead-zone quantizer function. As shown in the graph **1300**, the dead-zone (initially, 0-300) **1302** increases in size as the frequency of coefficients increases (finally, 0-450) **1304**. In the encoder **600** of **FIG. 6**, the region detector **606** characterizes a block as a texture or edge block, and this information **608** may be utilized by an adaptive quantizer **670** to provide frequency or order dependent dead zone resizing as shown in the increasing dead-zone function **1300** of **FIG. 13**.

[0135] Although a fine-grained function is shown in **FIG. 13**, this is not required. In another example, an adaptive quantizer provides low frequency quantization (for the first **32** coefficients of the scan) with different dead zone size than high frequency quantization (for the last 32 bits of the scan). Table 3 shows an adaptive quantization scheme with different dead zone sizes for on low-frequency and high-frequency coefficients.

TABLE 3

Coeff Value	Low Frequency	High Frequency
$0 < x \leqq 100$	0	0
$100 < x \leqq 200$	1	0
$200 < x \leqq 300$	2	0
$300 < x \leqq 400$	3	0
$400 < x \leqq 500$	4	1
$500 < x \leqq 600$	5	2
$600 < x \leqq 700$	6	3
$700 < x \leqq 800$	7	4
$800 < x \leqq 900$	8	5

[0136] Quantization for different dead zone sizes depending on frequency/order may be used in conjunction with or instead of different dead zone sizes for different types of content/block characterizations.

[0137] Moreover, for the sake of simplicity, Tables 2 and 3 and **FIG. 12** illustrate a coarse quantization with a quantization step size of 100. In practice, other quantization step sizes (e.g., 1, 2, 4, 6, etc.) are typically used, where the different quantization step sizes essentially trade off bit rate versus quality/fidelity. The dead zone sizes for these step sizes may similarly vary depending on block characterization and/or coefficient order. For example, the dead zone size is 1* the quantization step size in a first quantizer used for one kind of block, and it is 2* the quantization step size in a second quantizer used for another kind of block.

[0138] When blocks are a quantized distinctly, quantizer selection information may be provided to the decoder (along with step size information) as side information.

[0139] Alternatively, instead of using separate quantizers for different dead zone sizes, a single quantizer is used with different effective dead zone sizes for different types of blocks. In other words, the same set of quantized value tokens is basically used for coefficients regardless of dead zone size, but when the dead zone size is increased, the zero-value token is simply used for the one-value token, two-value token, etc. to map a wider range of coefficients around zero to the quantized value of zero. The other tokens (e.g., three-value token, four-value token, etc.) are the same. The one-value token, two-value token, etc. are unused, which makes entropy coding a bit less efficient, but selection between different quantizers and reconstruction rules is not needed.

EXAMPLE DIFFERENTIAL QUANTIZATION

[0140] Using differential quantization, an encoder may apply coarser quantization to some blocks while applying finer quantization to other blocks, depending on the block characterizations for those blocks. For example, smooth regions in a picture are coded with lower QP while texture regions or edge regions are coded with higher QP. This scheme will save bits in texture or edge regions by introducing distortion not likely to be noticed, and improve the overall perceptual quality of the encoded video.

[0141] The concept of differential quantization (often called D-Quant for short) is that macroblocks or other blocks within a video frame can be quantized using different quantization values (e.g., QPs). The quantization values for the decoder to use in reconstruction are then signaled in the bitstream.

[0142] An example syntax describes how macroblock level QP (MQUANT) is reconstructed at a decoder. A video clip header may indicate whether bi-level quantization is

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1005, Page 000024

employed. Then, if bi-level quantization is employed, for each video frame within the clip, a syntax element indicates a bi-level profile for the frame (e.g., DQPROFILE==All-Macroblocks) **1402**.

[**0143**] **FIG. 14** is a program listing of an example method of differential quantization **1400**. Alternatively, the encoder and decoder use another signaling syntax.

[**0144**] With reference to the pseudocode in **FIG. 14**, MQUANT is used by a decoder to dequantize a block. Thus, MQUANT is reconstructed at the decoder according to the program listing **1400**.

[**0145**] The method determines whether the frame contains differential quantization **1402**. Next, the method determines **1404** whether a macroblock is assigned one of two bi-level quantization values (e.g., PQUANT or ALTQUANT), or whether MQUANT is assigned **1406** for a macroblock by one of two equations **1408**, **1410**.

[**0146**] For example, a frame header may provide information from which PQUANT and ALTPQUANT are reconstructed as default quantization values.

[**0147**] If a simple bi-level quantization is indicated for the frame **1404**, the method determines which default value to use by a single bit in a macroblock header **1412**.

[**0148**] If an assigned quantization is indicated **1406**, then if MQDIFF is not equal to seven **1414**, then MQUANT is set equal to PQUANT plus MQDIFF. However, if MQDIFF is equal to seven, then MQUANT is set according to a five bit value for a given macroblock.

[**0149**] MQDIFF is a variable-sized syntax element present in certain macroblocks in I and P frames. The MQDIFF element is present when the picture layer syntax element DQPROFILE is set to indicate 'All Macroblocks'. The syntax depends on the DQBILEVEL syntax element as described below.

[**0150**] If DQBILEVEL=1, then MQDIFF is a 1 bit syntax element and the ABSMQ syntax element does not follow in the bitstream. If MQDIFF=zero **1412**, then MQUANT=PQUANT (meaning that PQUANT is used to dequantize the current macroblock). If MQDIFF=one **1416**, then MQUANT=ALTPQUANT.

[**0151**] If DQBILEVEL=zero **1406**, then MQDIFF is a 3 bit syntax element present in the bit stream. In this case MQDIFF decodes either to an MQUANT differential or to an escape code as follows. If MQDIFF does not equal seven **1410**, then MQDIFF is the differential added to PQUANT, and the ABSMQ syntax element does not follow in the bitstream. If MQDIFF is seven **1410**, ABSMQ directly encodes the value of MQUANT.

Alternatives

[**0152**] Having described and illustrated the principles of our invention with reference to illustrated examples, it will be recognized that the examples can be modified in arrangement and detail without departing from such principles. Additionally, as will be apparent to ordinary computer scientists, portions of the examples or complete examples can be combined with other portions of other examples in whole or in part. It should be understood that the programs, processes, or methods described herein are not related or limited to any particular type of computer apparatus, unless indicated otherwise. Various types of general purpose or specialized computer apparatus may be used with or perform operations in accordance with the teachings described herein. Elements of the illustrated embodiment shown in software may be implemented in hardware and vice versa. Techniques from one example can be incorporated into any of the other examples.

[**0153**] In view of the many possible embodiments to which the principles of the invention may be applied, it should be recognized that the details are illustrative only and should not be taken as limiting the scope of our invention. Rather, we claim as our invention all such embodiments as may come within the scope and spirit of the following claims and equivalents thereto.

We claim:

1. A computerized method comprising:

for each of plural blocks of video data, classifying the block according to content of the block and adaptively encoding the block based at least in part upon the block classification, wherein the adaptive encoding includes using two or more of:

adaptive median filtering depending at least in part on the block classification,

selective dropping of a last non-zero transform coefficient depending at least in part on the block classification,

adaptive dead zone re-sizing depending at least in part on the block classification, or

quantization step size adjustment depending at least in part on the block classification.

2. The method of claim 1, wherein the classifying the block includes:

determining a block gradient for the block, including determining a pixel gradient for each of plural pixels of the block, the pixel gradient between one or more samples for the pixel and one or more corresponding samples from surrounding pixels; and

evaluating the block gradient to determine a block value.

3. The method of claim 2 wherein the block classification is based at least in part on the block value, the block classification comprising one of smooth, texture, and edge.

4. The method of claim 2 wherein the block value is determined based on an average of pixel gradients.

5. The method of claim 2 wherein the block value is determined based on a median of pixel gradients.

6. The method of claim 2, wherein the pixel gradient is quantified as a sum of differences, and wherein the sum of differences is determined based upon both luminance and chrominance samples.

7. The method of claim 6, further including downsampling the luminance samples for the block before the sum of differences is determined.

8. The method of claim 2, wherein the determining the pixel gradient comprises using only pixel data from the plural pixels of the block.

9. The method of claim 2, wherein evaluating the block gradient to determine the block value comprises ordering pixel gradient values numerically and selecting a median value as the block value.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1005, Page 000025

10. The method of claim 1 wherein the adaptive median filtering includes changing filter dimensions depending on the block classification, and wherein the adaptive median filtering is performed on a motion compensation prediction residual.

11. The method of claim 1 wherein the selective dropping of a last non-zero coefficient further depends at least in part on

a number of consecutive zero coefficients preceding the last non-zero coefficient in a zig-zag scan.

12. The method of claim 1 wherein the quantization step size adjustment includes applying differential quantization depending at least in part on the block classification.

13. The method of claim 1 wherein the adaptive encoding further comprises, applying adaptive quantization dependent upon,

a characterization of the block, and

a frequency characterization of a coefficient within a block.

14. The method of claim 1 wherein the adaptive dead zone re-sizing includes switching between plural quantizer dead zone sizes depending at least in part on the block classification.

15. A video decoder, decoding an output of an encoder performing the method of claim 1.

16. A computer readable medium including computer executable instructions for performing the method of claim 1.

17. A video encoder comprising:

a region detector module for classifying blocks of video frames;

an adaptive filter module for median filtering motion compensation error data for one or more of the blocks, wherein the median filtering is based at least in part upon block classifications assigned by the region detector module; and

an adaptive quantization module for differentially quantizing the motion compensation error data based at least in part upon the block classifications assigned by the region detection module.

18. The video encoder of claim 17 wherein the adaptive quantization module is also for:

selectively dropping isolated last non-zero transform coefficients based at least in part on the block classifications, and/or

changing quantization dead-zones based at least in part on the block classifications.

19. A computer-readable program carrying medium having thereon computer-executable instructions comprising:

instructions for determining, for each of plural blocks of video data, pixel gradients for pixels within the block, the pixel gradients comprising sums of differences between video data of a subject pixel and plural surrounding pixels;

instructions for determining, for each of the plural blocks, a perceptual characterization of the block based on the plural pixel gradients for the block; and

instructions for adaptively encoding the plural blocks based on the perceptual characterizations of the plural blocks.

20. The computer-readable program carrying medium of claim 19 wherein the computer-executable instructions for adaptively encoding comprise:

instructions for applying a median filter selected using one of the perceptual characterizations;

instructions for dropping a trailing transform coefficient based at least in part on one of the perceptual characterizations; and/or

instructions for applying in quantization a dead-zone selected using one of the perceptual characterizations.

* * * * *