US 20030156646A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2003/0156646 A1

Hsu et al. (43) Pub. Date: Aug. 21, 2003

(54) MULTI-RESOLUTION MOTION ESTIMATION AND COMPENSATION

(75) Inventors: **Pohsiang Hsu**, Redmond, WA (US); **Chih-Lung Lin**, Redmond, WA (US); **Ming-Chieh Lee**, Bellevue, WA (US)

Correspondence Address:
KLARQUIST SPARKMAN, LLP
One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, OR 97204 (US)

(73) Assignee: **Microsoft Corporation**

(21) Appl. No.: **10/322,351**

(22) Filed: **Dec. 17, 2002**

Related U.S. Application Data

(60) Provisional application No. 60/341,674, filed on Dec. 17, 2001.

Publication Classification

(51) **Int. Cl.**[7] **H04N 7/12**

(52) **U.S. Cl.** **375/240.16**; 375/240.13

(57) **ABSTRACT**

Techniques and tools for motion estimation and compensation are described. For example, a video encoder adaptively switches between different motion resolutions, which allows the encoder to select a suitable resolution for a particular video source or coding circumstances.

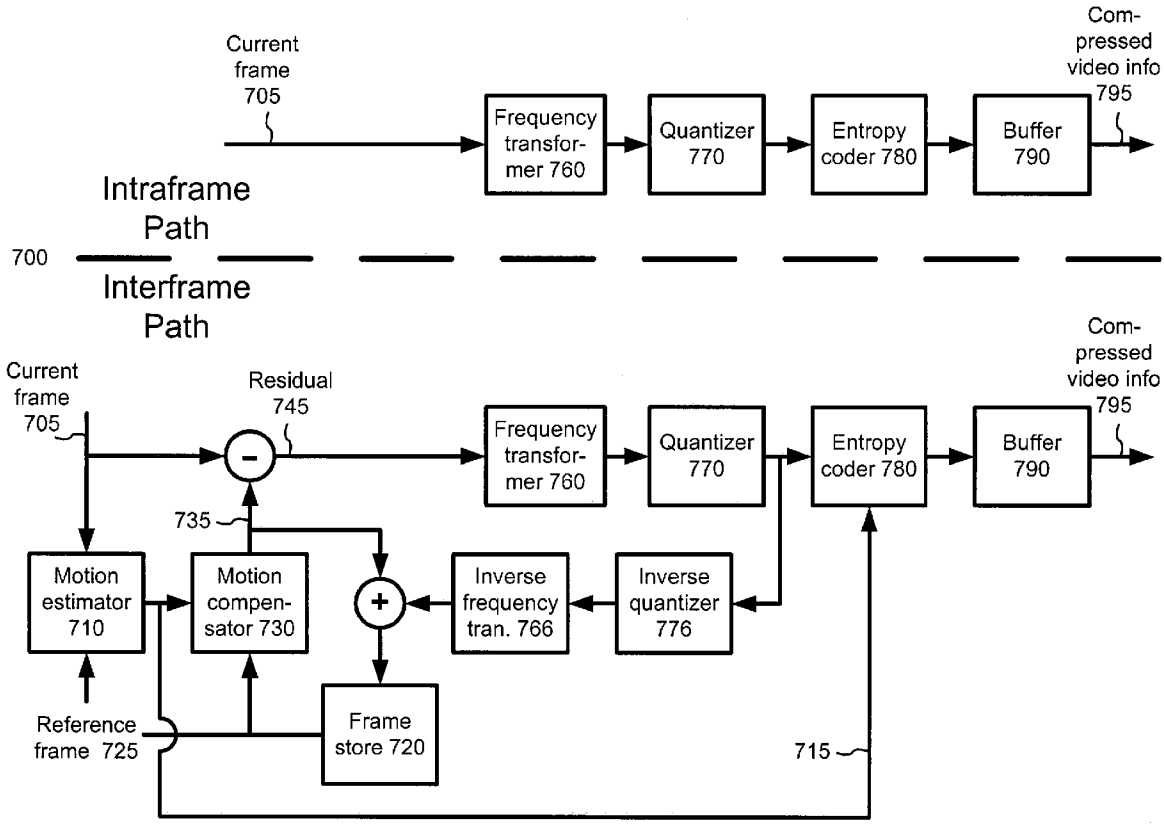


Figure 1, prior art

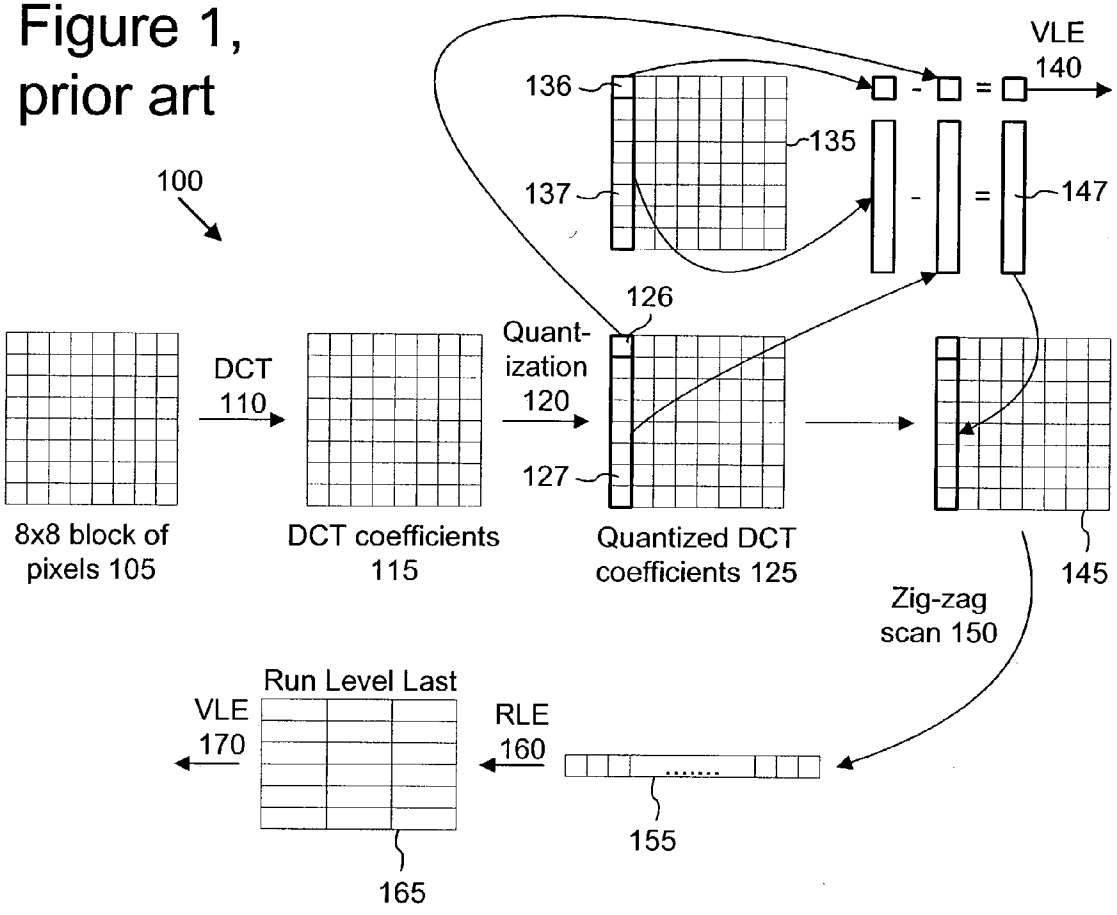


Figure 2, prior art

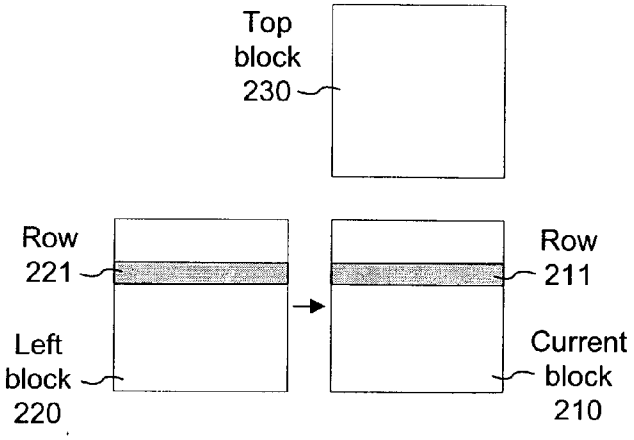


Figure 3, prior art

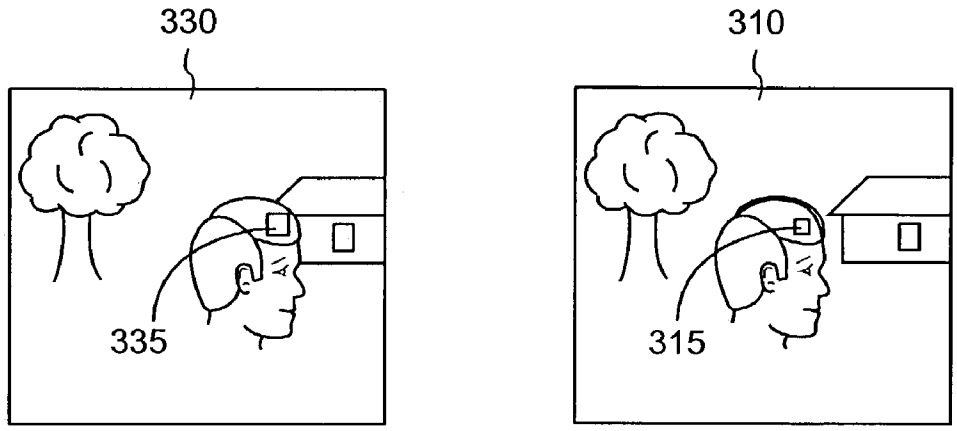


Figure 6

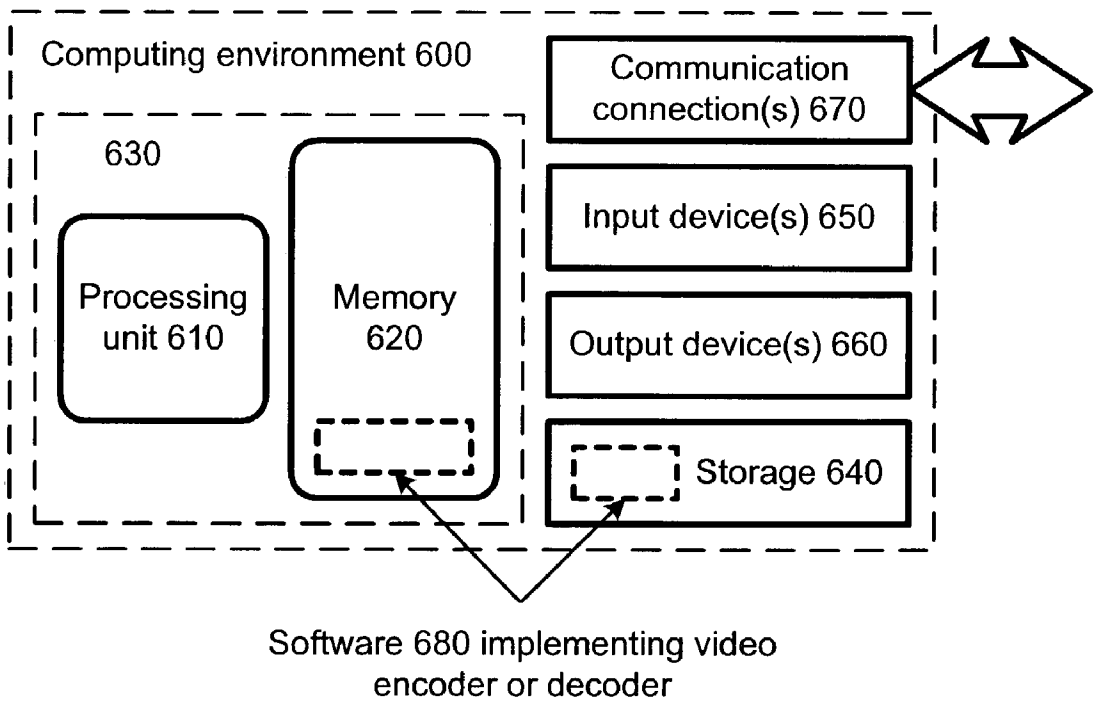
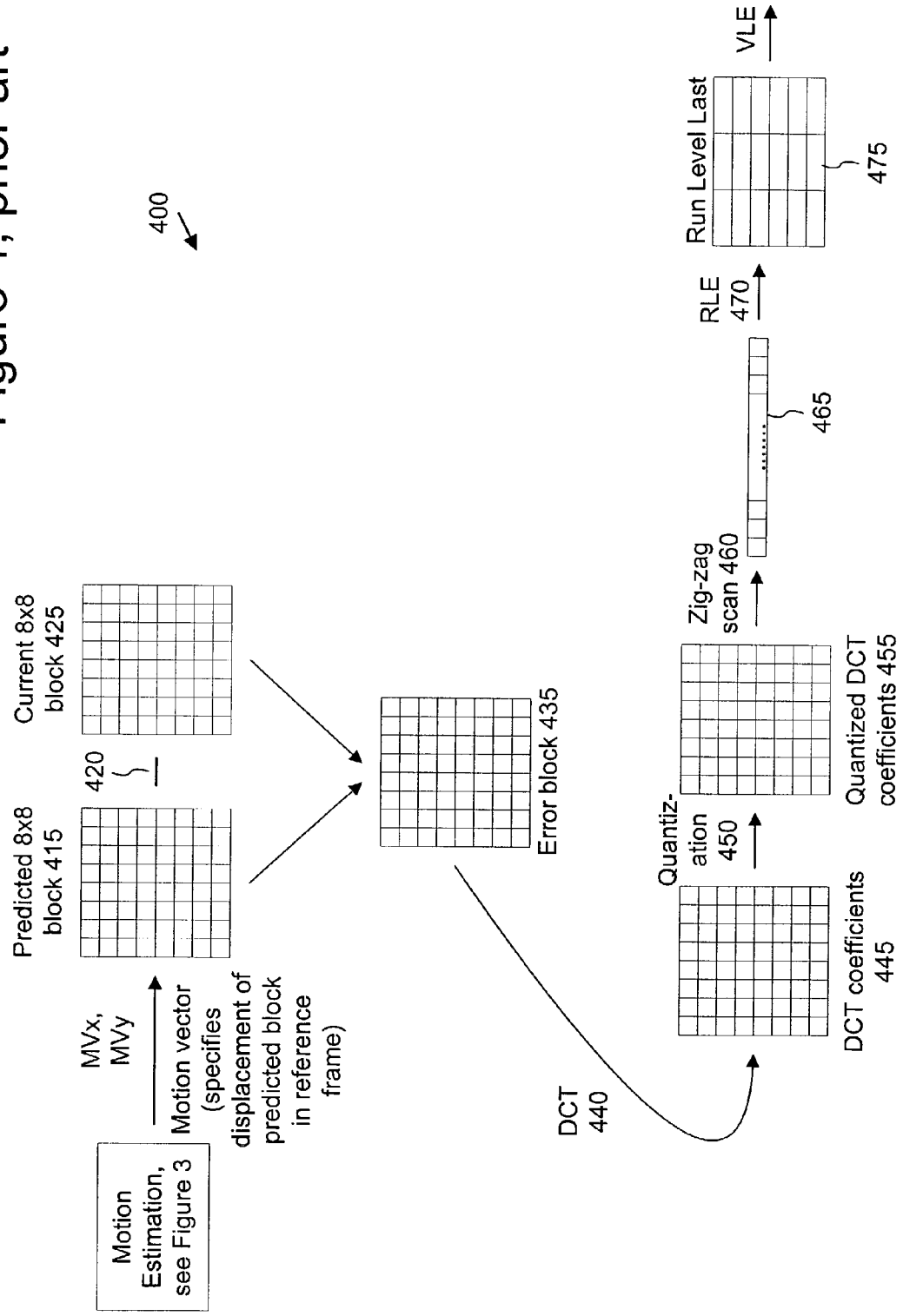


Figure 4, prior art



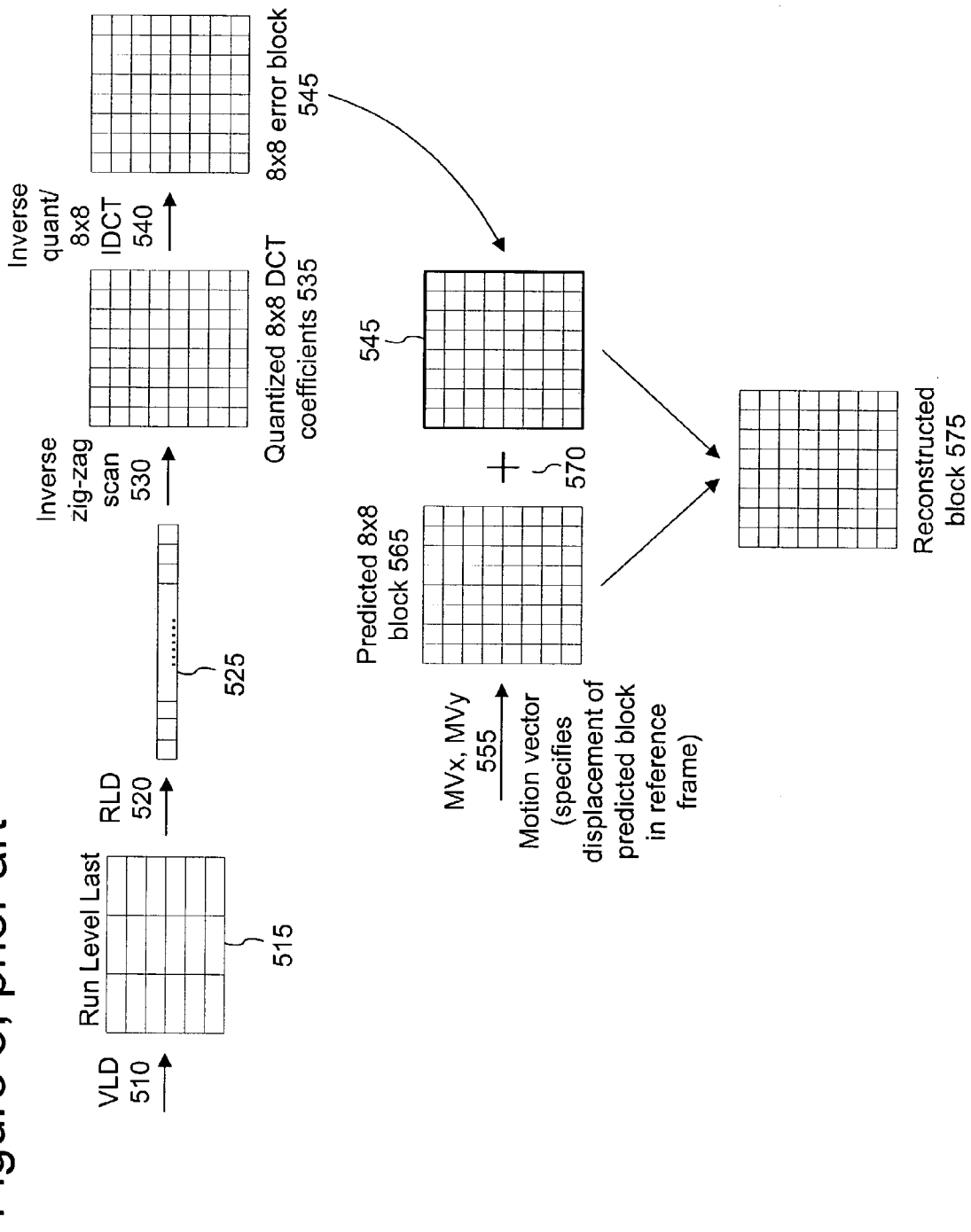
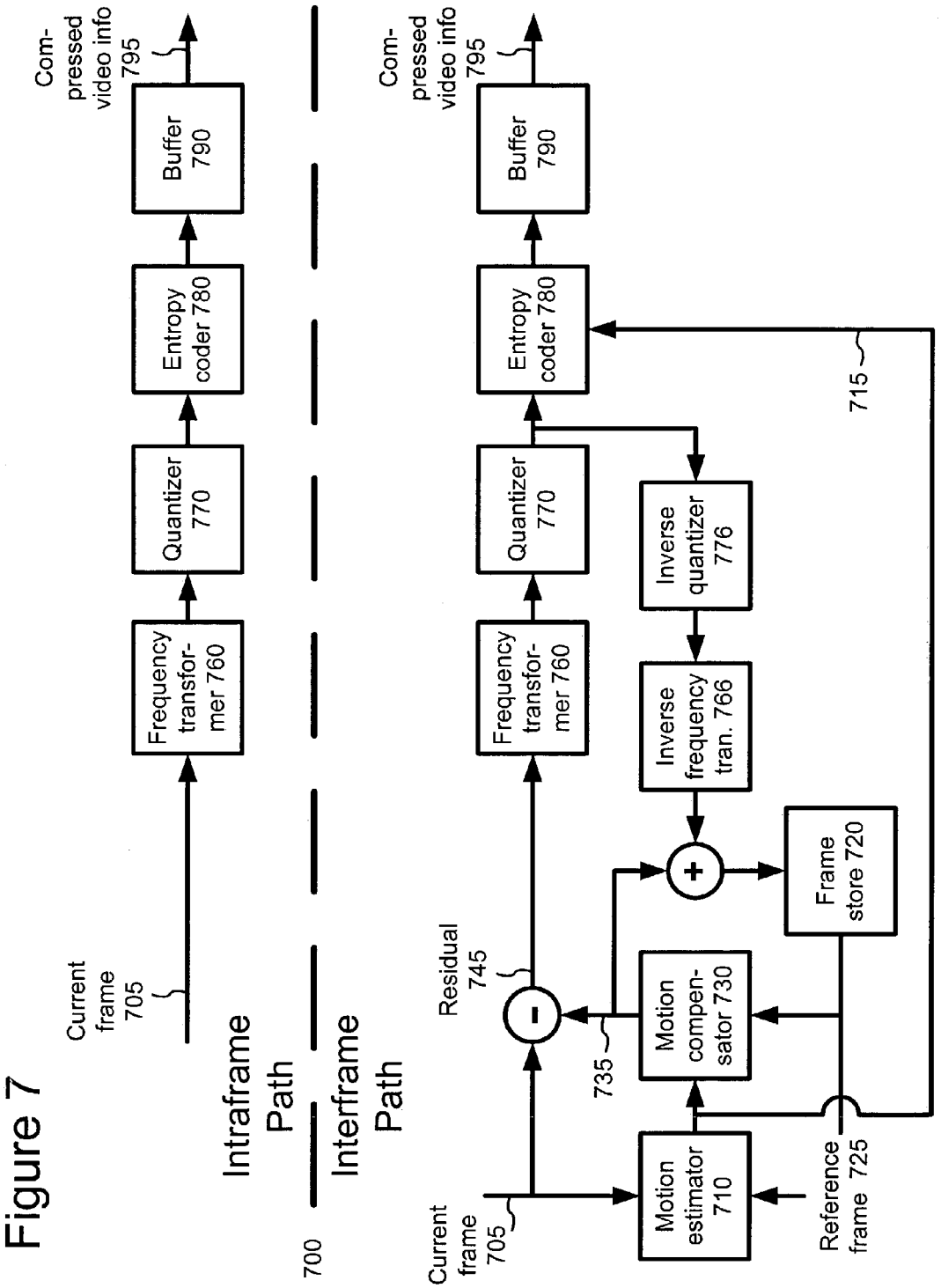


Figure 5, prior art

Figure 7


Current frame 705
Frequency transfor-mer 760
Quantizer 770
Entropy coder 780
Buffer 790
Com-pressed video info 795
Intraframe Path
Interframe Path
700
Residual 745
Motion estimator 710
Motion compen-sator 730
735
Inverse frequency tran. 766
Inverse quantizer 776
Frame store 720
Reference frame 725
715

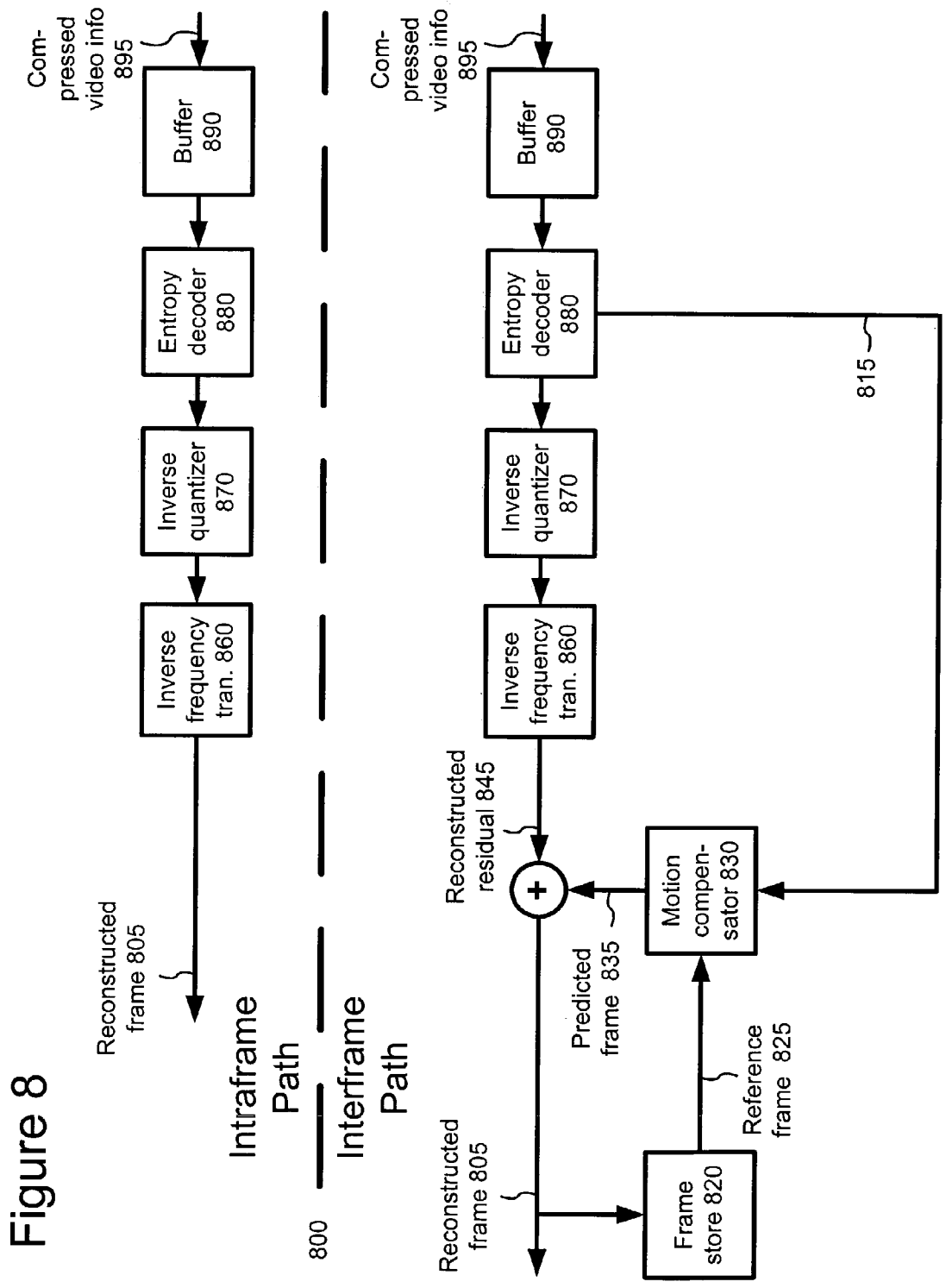

Com-
pressed
video info
895
Buffer
890
Entropy
decoder
880
Inverse
quantizer
870
Inverse
frequency
tran. 860
Reconstructed
frame 805
Intraframe
Path
800
Interframe
Path
Com-
pressed
video info
895
Buffer
890
Entropy
decoder
880
815
Inverse
quantizer
870
Inverse
frequency
tran. 860
Reconstructed
residual 845
+
Motion
compen-
sator 830
Predicted
frame 835
Reconstructed
frame 805
Frame
store 820
Reference
frame 825

Figure 8

Figure 9

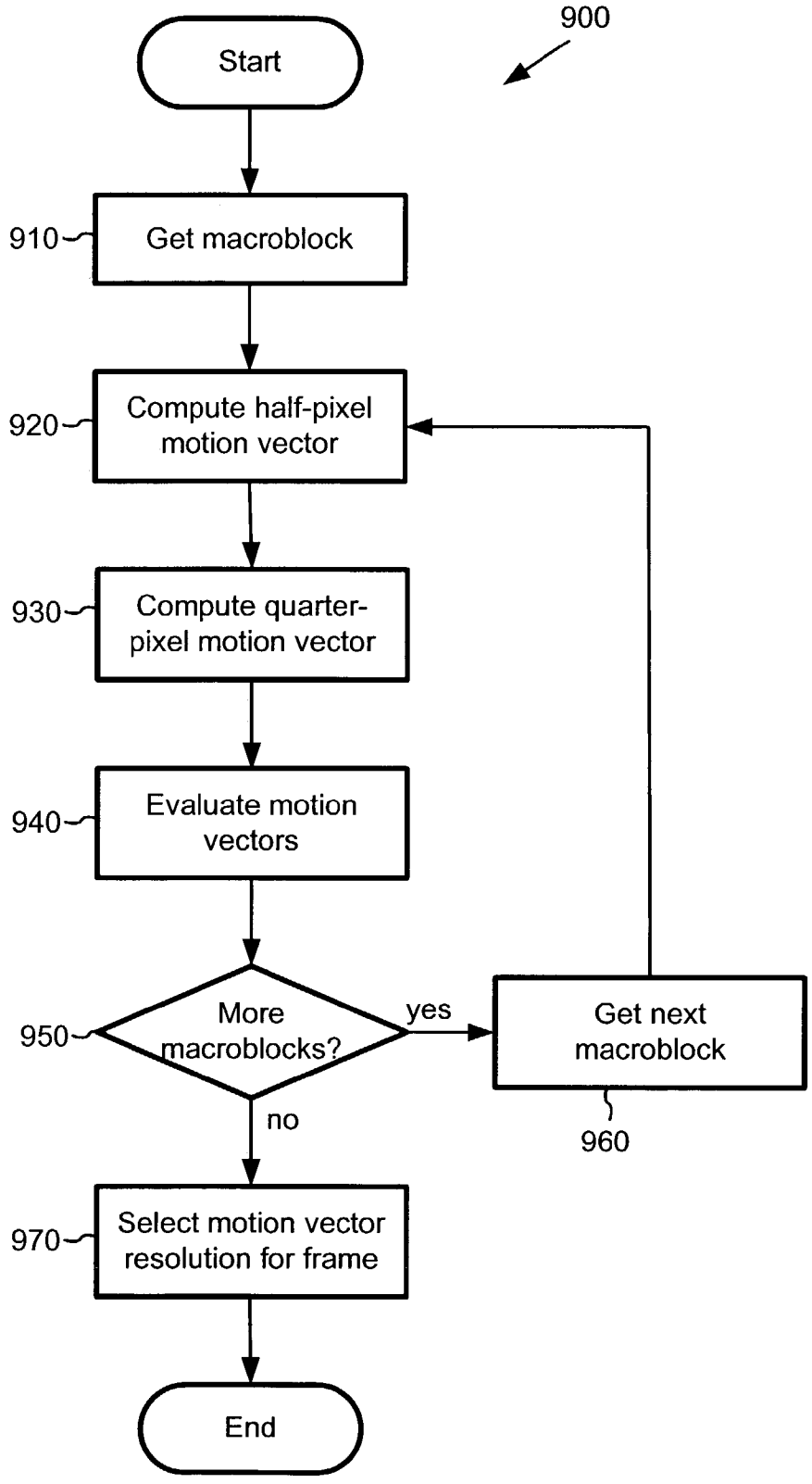

900
Start
910
Get macroblock
920
Compute half-pixel motion vector
930
Compute quarter-pixel motion vector
940
Evaluate motion vectors
950
More macroblocks?
yes
Get next macroblock
960
no
970
Select motion vector resolution for frame
End

Figure 10a

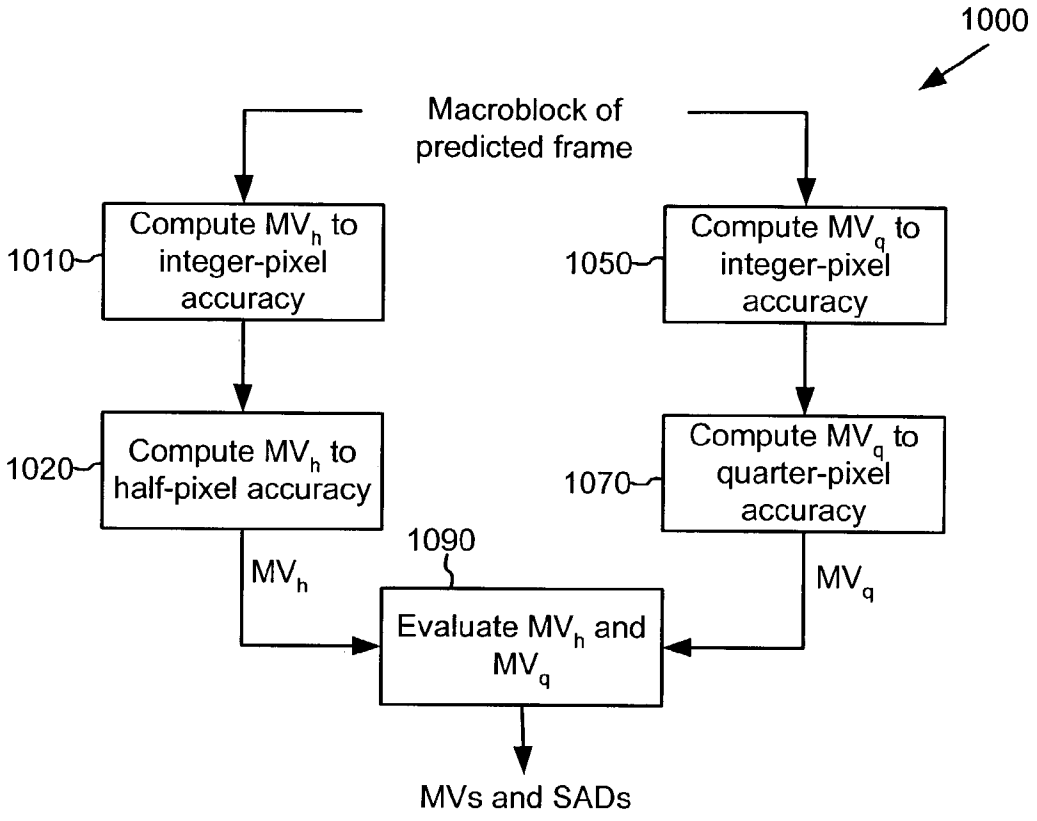

1000
Macroblock of predicted frame
1010
Compute MV_h to integer-pixel accuracy
1020
Compute MV_h to half-pixel accuracy
1050
Compute MV_q to integer-pixel accuracy
1070
Compute MV_q to quarter-pixel accuracy
1090
MV_h
MV_q
Evaluate MV_h and MV_q
MVs and SADs

Figure 10b

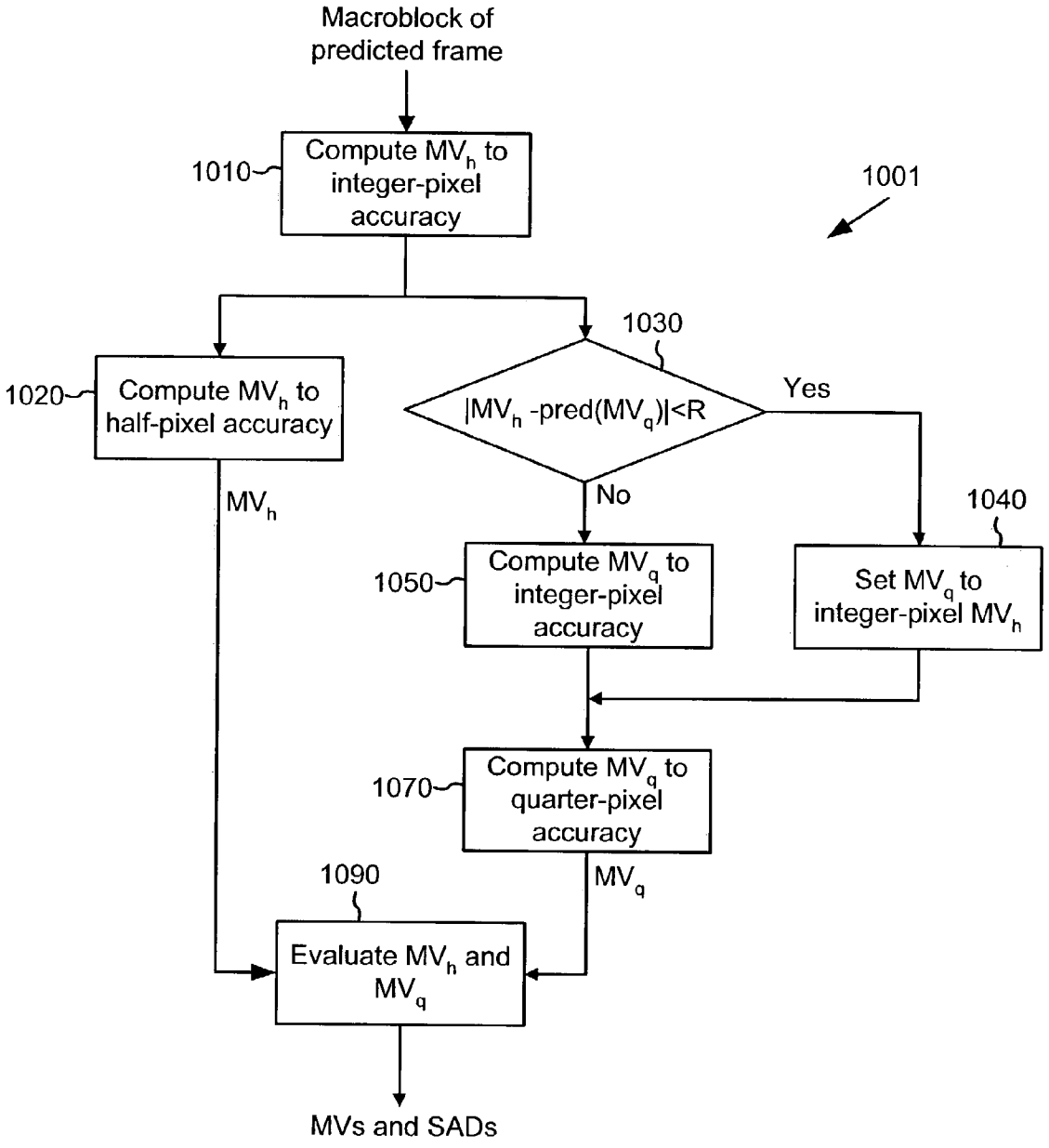
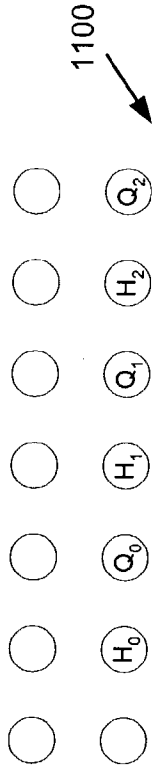
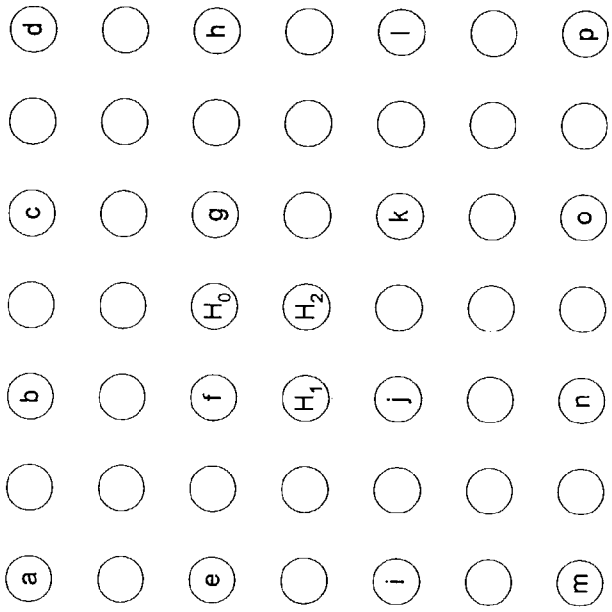

Macroblock of predicted frame
1010
Compute MV_h to integer-pixel accuracy
1001
1030
1020
Compute MV_h to half-pixel accuracy
$|MV_h - pred(MV_q)| < R$
Yes
No
MV_h
1040
1050
Compute MV_q to integer-pixel accuracy
Set MV_q to integer-pixel MV_h
1070
Compute MV_q to quarter-pixel accuracy
1090
MV_q
Evaluate MV_h and MV_q
MVs and SADs

Figure 11



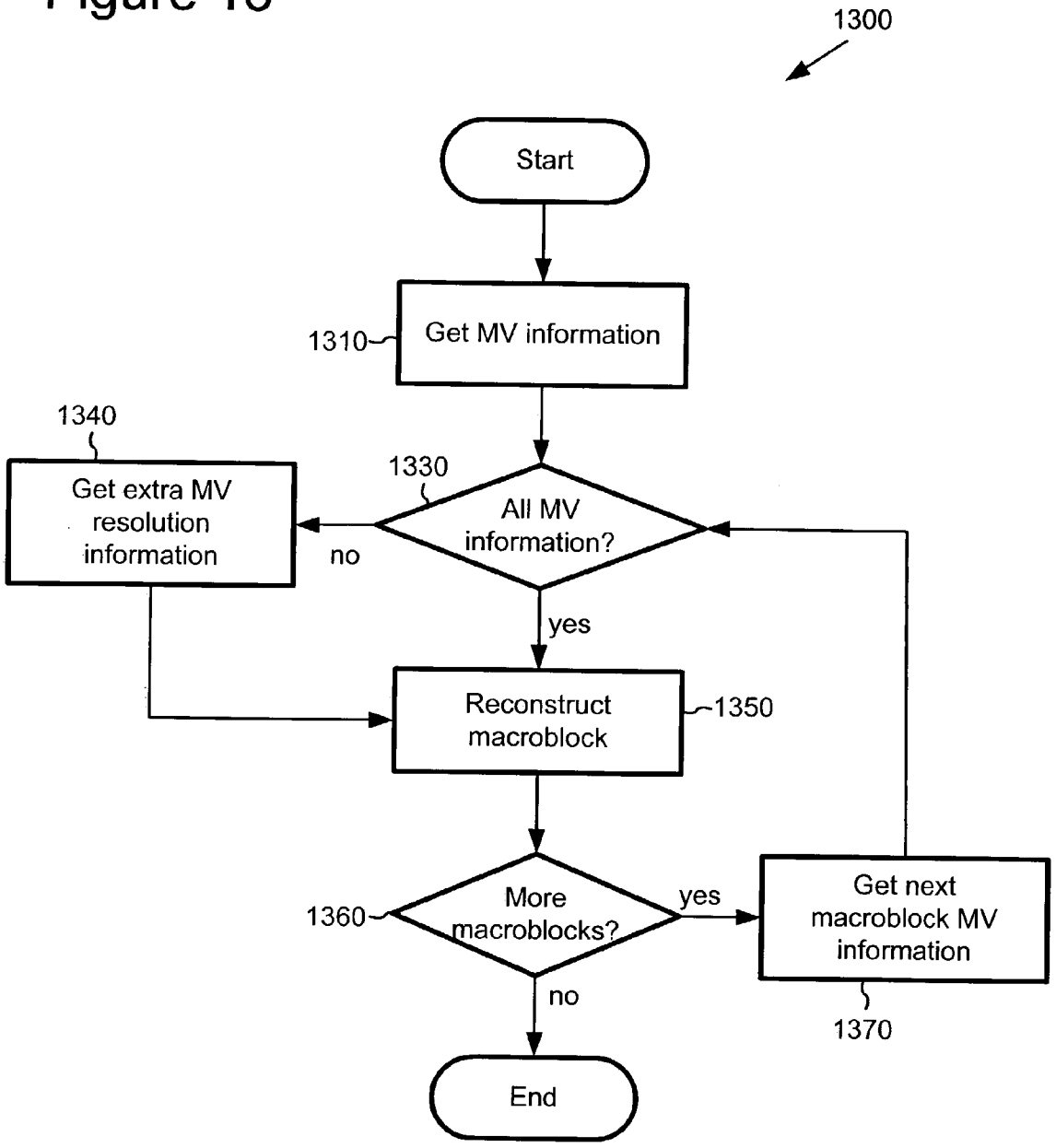
- Each circle denotes a pixel location in quarter-pixel units.
- I denotes an integer-pixel location.
- $H_0,\ldots,H_7$ denote searched half-pixel locations.
- $Q_0,\ldots,Q_7$ denote searched quarter-pixel locations

Figure 12



- a, b, c,..., p denote integer-pixel locations.
- H_0, H_1, H_2 denote half-pixel locations.

Figure 13


1300
Start
1310
Get MV information
1340
Get extra MV resolution information
1330
All MV information?
no
yes
Reconstruct macroblock
1350
1360
More macroblocks?
yes
Get next macroblock MV information
1370
no
End

MULTI-RESOLUTION MOTION ESTIMATION AND COMPENSATION

RELATED APPLICATION INFORMATION

[0001] The present application claims the benefit of U.S. Provisional Patent Application Serial No. 60/341,674, entitled "Techniques and Tools for Video Encoding and Decoding," filed Dec. 17, 2001, the disclosure of which is incorporated by reference. The following concurrently filed U.S. patent applications relate to the present application: 1) U.S. patent application Ser. No. aa/bbb,ccc, entitled, "Motion Compensation Loop With Filtering," filed concurrently herewith; 2) U.S. patent application Ser. No. aa/bbb, ccc, entitled, "Spatial Extrapolation of Pixel Values in Intraframe Video Coding and Decoding," filed concurrently herewith; and 3) U.S. patent application Ser. No. aa/bbb,ccc, entitled, "Sub-Block Transform Coding of Prediction Residuals," filed concurrently herewith.

TECHNICAL FIELD

[0002] Techniques and tools for motion estimation and compensation are described. For example, a video encoder adaptively switches between different motion resolutions, which allows the encoder to select a suitable resolution for a particular video source or coding circumstances.

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 bitrate, 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 bitrate of digital video. Compression can be lossless, in which quality of the video does not suffer but decreases in bitrate are limited by the complexity of the video. Or, compression can be lossy, in which quality of the video suffers but decreases in bitrate 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 are called typically called predicted frames, P-frames, or B-frames.

[0006] Microsoft Corporation's Windows Media Video, Version 7 ["WMV7"] includes a video encoder and a video decoder. The WMV7 encoder uses intraframe and interframe compression, and the WMV7 decoder uses intraframe and interframe decompression.

[0007] A. Intraframe Compression in WMV7

[0008] **FIG. 1** illustrates block-based intraframe compression (**100**) of a block (**105**) of pixels in a key frame in the WMV7 encoder. A block is a set of pixels, for example, an 8×8 arrangement of pixels. The WMV7 encoder splits a key video frame into 8×8 blocks of pixels 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.

[0009] 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, which is analogous to dividing each coefficient by the same value and rounding. For example, if a DCT coefficient value is 163 and the step size is 10, the quantized DCT coefficient value is 16. Quantization is lossy. The reconstructed DCT coefficient value will be 160, not 163. 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.

[0010] 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.

[0011] 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.

[0012] 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.

[0013] 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.

[0014] A key frame contributes much more to bitrate than a predicted frame. In low or mid-bitrate applications, key frames are often critical bottlenecks for performance, so efficient compression of key frames is critical.

[0015] **FIG. 2** illustrates a disadvantage of intraframe compression such as shown in **FIG. 1**. In particular, exploitation of redundancy between blocks of the key frame is limited to prediction of a subset of frequency coefficients (e.g., the DC coefficient and the left column (or top row) of AC coefficients) from the left **(220)** or top **(230)** neighboring block of a block **(210)**. The DC coefficient represents the average of the block, the left column of AC coefficients represents the averages of the rows of a block, and the top row represents the averages of the columns. In effect, prediction of DC and AC coefficients as in WMV7 limits extrapolation to the row-wise (or column-wise) average signals of the left (or top) neighboring block. For a particular row **(221)** in the left block **(220)**, the AC coefficients in the left DCT coefficient column for the left block **(220)** are used to predict the entire corresponding row **(211)** of the block **(210)**. The disadvantages of this prediction include:

[0016] 1) Since the prediction is based on averages, the far edge of the neighboring block has the same influence on the predictor as the adjacent edge of the neighboring block, whereas intuitively the far edge should have a smaller influence.

[0017] 2) Only the average pixel value across the row (or column) is extrapolated.

[0018] 3) Diagonally oriented edges or lines that propagate from either predicting block (top or left) to the current block are not predicted adequately.

[0019] 4) When the predicting block is to the left, there is no enforcement of continuity between the last row of the top block and the first row of the extrapolated block.

[0020] B. Interframe Compression in WMV7

[0021] Interframe compression in the WMV7 encoder uses block-based motion compensated prediction coding followed by transform coding of the residual error.

[0022] **FIGS. 3 and 4** illustrate the block-based interframe compression for a predicted frame in the WMV7 encoder. In particular, **FIG. 3** illustrates motion estimation for a predicted frame **(310)** and **FIG. 4** illustrates compression of a prediction residual for a motion-estimated block of a predicted frame.

[0023] The WMV7 encoder splits a predicted frame into 8×8 blocks of pixels. Groups of 4 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. 3**, the WMV7 encoder computes a motion vector for a macroblock **(315)** in the predicted frame **(310)**. To compute the motion vector, the encoder searches in a search area **(335)** of a reference frame **(330)**. Within the search area **(335)**, the encoder compares the macroblock **(315)** from the predicted frame **(310)** to various candidate macroblocks in order to find a candidate macroblock that is a good match. The encoder can check candidate macroblocks every pixel or every ½ pixel in the search area **(335)**, depending on the desired motion estimation resolution for the encoder. Other video encoders check at other increments, for example, every ¼ pixel. For a candidate macroblock, the encoder checks the difference between the macroblock **(315)** of the predicted frame **(310)** and the candidate macroblock and the cost of encoding the motion vector for that macroblock. After the encoder finds a good matching macroblock, the block matching process ends. The encoder outputs the motion vector (entropy coded) for the matching macroblock so the decoder can find the matching macroblock during decoding. When decoding the predicted frame **(310)**, a decoder uses the motion vector to compute a prediction macroblock for the macroblock **(315)** using information from the reference frame **(330)**. The prediction for the macroblock **(315)** 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 **(315)** itself.

[0024] Motion estimation and compensation are effective compression techniques, but various previous motion estimation/compensation techniques (as in WMV7 and elsewhere) have several disadvantages, including:

[0025] 1) The resolution of the motion estimation (i.e., pixel, ½ pixel, ¼ pixel increments) does not adapt to the video source. For example, for different qualities of video source (clean vs. noisy), the video encoder uses the same resolution of motion estimation, which can hurt compression efficiency.

[0026] 2) For ¼ pixel motion estimation, the search strategy fails to adequately exploit previously completed computations to speed up searching.

[0027] 3) For ¼ pixel motion estimation, the search range is too large and inefficient. In particular, the horizontal resolution is the same as the vertical resolution in the search range, which does not match the motion characteristics of many video signals.

[0028] 4) For ¼ pixel motion estimation, the representation of motion vectors is inefficient to the extent bit allocation for horizontal movement is the same as bit allocation for vertical resolution.

[0029] **FIG. 4** illustrates the computation and encoding of an error block **(435)** for a motion-estimated block in the WMV7 encoder. The error block **(435)** is the difference between the predicted block **(415)** and the original current block **(425)**. The encoder applies a DCT **(440)** to error block **(435)**, resulting in 8×8 block **(445)** of coefficients. Even more than was the case with DCT coefficients for pixel values, the significant information for the error block **(435)** is concentrated in low frequency coefficients (conventionally, the upper left of the block **(445)**) and many of the high frequency coefficients have values of zero or close to zero (conventionally, the lower right of the block **(445)**).

[0030] The encoder then quantizes **(450)** the DCT coefficients, resulting in an 8×8 block of quantized DCT coefficients **(455)**. The quantization step size is adjustable. Again, 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 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.

[0031] The encoder then prepares the 8×8 block (455) of quantized DCT coefficients for entropy encoding. The encoder scans (460) the 8×8 block (455) into a one dimensional array (465) with 64 elements, such that coefficients are generally ordered from lowest frequency to highest frequency, which typical creates long runs of zero values.

[0032] The encoder entropy encodes the scanned coefficients using a variation of run length coding (470). The encoder selects an entropy code from one or more run/level/last tables (475) and outputs the entropy code.

[0033] FIG. 5 shows the decoding process (500) for an inter-coded block. Due to the quantization of the DCT coefficients, the reconstructed block (575) is not identical to the corresponding original block. The compression is lossy.

[0034] In summary of FIG. 5, a decoder decodes (510, 520) entropy-coded information representing a prediction residual using variable length decoding and one or more run/level/last tables (515). The decoder inverse scans (530) a one-dimensional array (525) storing the entropy-decoded information into a two-dimensional block (535). The decoder inverse quantizes and inverse discrete cosine transforms (together, 540) the data, resulting in a reconstructed error block (545). In a separate path, the decoder computes a predicted block (565) using motion vector information (555) for displacement from a reference frame. The decoder combines (570) the predicted block (555) with the reconstructed error block (545) to form the reconstructed block (575).

[0035] 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. 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. One of the goals of a video compression scheme is to try to improve the rate-distortion—in other words to try to achieve the same distortion using fewer bits (or the same bits and lower distortion).

[0036] Compression of prediction residuals as in WMV7 can dramatically reduce bitrate while slightly or moderately affecting quality, but the compression technique is less than optimal in some circumstances. The size of the frequency transform is the size of the prediction residual block (e.g., an 8×8 DCT for an 8×8 prediction residual). In some circumstances, this fails to exploit localization of error within the prediction residual block.

[0037] C. Post-processing with a Deblocking Filter in WMV7

[0038] For block-based video compression and decompression, quantization and other lossy processing stages introduce distortion that commonly shows up as blocky artifacts—perceptible discontinuities between blocks.

[0039] To reduce the perceptibility of blocky artifacts, the WMV7 decoder can process reconstructed frames with a deblocking filter. The deblocking filter smoothes the boundaries between blocks.

[0040] While the deblocking filter in WMV7 improves perceived video quality, it has several disadvantages. For example, the smoothing occurs only on reconstructed output in the decoder. Therefore, prediction processes such as motion estimation cannot take advantage of the smoothing. Moreover, the smoothing by the post-processing filter can be too extreme.

[0041] D. Standards for Video Compression and Decompression

[0042] Aside from WMV7, 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, and H.263 standards from the International Telecommunication Union ["ITU"]. Like WMV7, these standards use a combination of intraframe and interframe compression, although the standards typically differ from WMV7 in the details of the compression techniques used. For additional detail about the standards, see the standards' specifications themselves.

[0043] 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

[0044] In summary, the detailed description is directed to various techniques and tools for motion estimation and compensation. These techniques and tools address several of the disadvantages of motion estimation and compensation according to the prior art. The various techniques and tools can be used in combination or independently.

[0045] According to a first set of techniques and tools, a video encoder adaptively switches between multiple different motion resolutions, which allows the encoder to select a suitable resolution for a particular video source or coding circumstances. For example, the encoder adaptively switches between pixel, half-pixel, and quarter-pixel resolutions. The encoder can switch based upon a closed-loop decision involving actual coding with the different options, or based upon an open-loop estimation. The encoder switches resolutions on a frame-by-frame basis or other basis.

[0046] According to a second set of techniques and tools, a video encoder uses previously computed results from a first resolution motion estimation to speed up another resolution motion estimation. For example, in some circumstances, the encoder searches for a quarter-pixel motion vector around an integer-pixel motion vector that was also used in half-pixel motion estimation. Or, the encoder uses previously computed half-pixel location values in computation of quarter-pixel location values.

[0047] According to a third set of techniques and tools, a video encoder uses a search range with different directional resolutions. This allows the encoder and decoder to place greater emphasis on directions likely to have more motion, and to eliminate the calculation of numerous sub-pixel values in the search range. For example, the encoder uses a search range with quarter-pixel increments and resolution horizontally, and half-pixel increments and resolution ver-

tically. The search range is effectively quarter the size of a full quarter-by-quarter-pixel search range, and the encoder eliminates calculation of many of the quarter-pixel location points.

[0048] According to a fourth set of techniques and tools, a video encoder uses a motion vector representation with different bit allocation for horizontal and vertical motion. This allows the encoder to reduce bitrate by eliminating resolution that is less essential to quality. For example, the encoder represents a quarter-pixel motion vector by adding 1 bit to a half-pixel motion vector code to indicate a corresponding quarter-pixel location.

[0049] Additional features and advantages will be made apparent from the following detailed description of different embodiments that proceeds with reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0050] **FIG. 1** is a diagram showing block-based intraframe compression of an 8×8 block of pixels according to prior art.

[0051] **FIG. 2** is a diagram showing prediction of frequency coefficients according to the prior art.

[0052] **FIG. 3** is a diagram showing motion estimation in a video encoder according to the prior art.

[0053] **FIG. 4** 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.

[0054] **FIG. 5** is a diagram showing block-based intraframe decompression for an 8×8 block of prediction residuals according to the prior art.

[0055] **FIG. 6** is a block diagram of a suitable computing environment in which several described embodiments may be implemented.

[0056] **FIG. 7** is a block diagram of a generalized video encoder system used in several described embodiments.

[0057] **FIG. 8** is a block diagram of a generalized video decoder system used in several described embodiments.

[0058] **FIG. 9** is a flowchart showing a technique for selecting a motion estimation resolution for a predicted frame in a video encoder.

[0059] **FIGS. 10***a* and **10***b* are flowcharts showing techniques for computing and evaluating motion vectors of a predicted frame in a video encoder.

[0060] **FIG. 11** is a chart showing search locations for sub-pixel motion estimation.

[0061] **FIG. 12** is a chart showing sub-pixel locations with values computed by interpolation in sub-pixel motion estimation.

[0062] **FIG. 13** is a flowchart showing a technique for entropy decoding motion vectors of different resolutions in a video decoder.

DETAILED DESCRIPTION

[0063] The present application relates to techniques and tools for video encoding and decoding. In various described embodiments, a video encoder incorporates techniques that improve the efficiency of interframe coding, a video decoder incorporates techniques that improve the efficiency of interframe decoding, and a bitstream format includes flags and other codes to incorporate the techniques.

[0064] The various techniques and tools can be used in combination or independently. Different embodiments implement one or more of the described techniques and tools.

I. Computing Environment

[0065] **FIG. 6** illustrates a generalized example of a suitable computing environment (**600**) in which several of the described embodiments may be implemented. The computing environment (**600**) 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.

[0066] With reference to **FIG. 6**, the computing environment (**600**) includes at least one processing unit (**610**) and memory (**620**). In **FIG. 6**, this most basic configuration (**630**) is included within a dashed line. The processing unit (**610**) 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 (**620**) 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 (**620**) stores software (**680**) implementing a video encoder or decoder.

[0067] A computing environment may have additional features. For example, the computing environment (**600**) includes storage (**640**), one or more input devices (**650**), one or more output devices (**660**), and one or more communication connections (**670**). An interconnection mechanism (not shown) such as a bus, controller, or network interconnects the components of the computing environment (**600**). Typically, operating system software (not shown) provides an operating environment for other software executing in the computing environment (**600**), and coordinates activities of the components of the computing environment (**600**).

[0068] The storage (**640**) 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 (**600**). The storage (**640**) stores instructions for the software (**680**) implementing the video encoder or decoder.

[0069] The input device(s) (**650**) 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 (**600**). For audio or video encoding, the input device(s) (**650**) 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**30** RW that reads audio or video samples into the computing environment (**600**). The output device(s) (**660**) may be a display, printer, speaker, CD-writer, or another device that provides output from the computing environment (**600**).

[0070] The communication connection(s) (670) 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.

[0071] The techniques and tools can be described in the general context of computer-executable 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 (600), computer-readable media include memory (620), storage (640), communication media, and combinations of any of the above.

[0072] 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. 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.

[0073] For the sake of presentation, the detailed description uses terms like "determine,""select,""adjust," 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.

II. Generalized Video Encoder and Decoder

[0074] **FIG. 7** is a block diagram of a generalized video encoder (700) and **FIG. 8** is a block diagram of a generalized video decoder (800).

[0075] 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, **FIGS. 7 and 8** usually 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 bitstream, typically after entropy encoding of the side information. The format of the output bitstream can be Windows Media Video version 8 format or another format.

[0076] The encoder (700) and decoder (800) are block-based and use a 4:2:0 macroblock format with each macroblock including 4 luminance 8×8 luminance blocks (at times treated as one 16×16 macroblock) and two 8×8 chrominance blocks. Alternatively, the encoder (700) and decoder (800) are object-based, use a different macroblock or block format, or perform operations on sets of pixels of different size or configuration than 8×8 blocks and 16×16 macroblocks.

[0077] 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.

[0078] A. Video Encoder

[0079] **FIG. 7** is a block diagram of a general video encoder system (700). The encoder system (700) receives a sequence of video frames including a current frame (705), and produces compressed video information (795) as output. Particular embodiments of video encoders typically use a variation or supplemented version of the generalized encoder (700).

[0080] The encoder system (700) compresses predicted frames and key frames. For the sake of presentation, **FIG. 7** shows a path for key frames through the encoder system (700) and a path for forward-predicted frames. Many of the components of the encoder system (700) 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.

[0081] A predicted frame [also called p-frame, b-frame for bi-directional prediction, or inter-coded frame] is represented in terms of prediction (or difference) from one or more other 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.

[0082] If the current frame (705) is a forward-predicted frame, a motion estimator (710) estimates motion of macroblocks or other sets of pixels of the current frame (705) with respect to a reference frame, which is the reconstructed previous frame (725) buffered in the frame store (720). In alternative embodiments, the reference frame is a later frame or the current frame is bi-directionally predicted. The motion estimator (710) 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 (710) outputs as side information motion information (715) such as motion vectors. A motion compensator (730) applies the motion information (715) to the reconstructed previous frame (725) to form a motion-compensated current frame (735). The prediction is rarely perfect, however, and the difference between the motion-compensated current frame (735) and the original current frame (705) is the prediction residual (745). Alternatively, a motion estimator and motion compensator apply another type of motion estimation/compensation.

[0083] A frequency transformer (760) converts the spatial domain video information into frequency domain (i.e., spectral) data. For block-based video frames, the frequency transformer (760) applies a discrete cosine transform ["DCT"] or variant of DCT to blocks of the pixel data or prediction residual data, producing blocks of DCT coeffi-

cients. Alternatively, the frequency transformer (**760**) applies another conventional frequency transform such as a Fourier transform or uses wavelet or subband analysis. In embodiments in which the encoder uses spatial extrapolation (not shown in **FIG. 7**) to encode blocks of key frames, the frequency transformer (**760**) can apply a re-oriented frequency transform such as a skewed DCT to blocks of prediction residuals for the key frame. In other embodiments, the frequency transformer (**760**) applies an 8×8, 8×4, 4×8, or other size frequency transforms (e.g., DCT) to prediction residuals for predicted frames.

[**0084**] A quantizer (**770**) then quantizes the blocks of spectral data coefficients. The quantizer applies uniform, scalar quantization to the spectral data with a step-size that varies on a frame-by-frame basis or other basis. Alternatively, the quantizer applies another type of quantization to the spectral data coefficients, for example, a non-uniform, vector, or non-adaptive quantization, or directly quantizes spatial domain data in an encoder system that does not use frequency transformations. In addition to adaptive quantization, the encoder (**700**) can use frame dropping, adaptive filtering, or other techniques for rate control.

[**0085**] When a reconstructed current frame is needed for subsequent motion estimation/compensation, an inverse quantizer (**776**) performs inverse quantization on the quantized spectral data coefficients. An inverse frequency transformer (**766**) then performs the inverse of the operations of the frequency transformer (**760**), producing a reconstructed prediction residual (for a predicted frame) or a reconstructed key frame. If the current frame (**705**) was a key frame, the reconstructed key frame is taken as the reconstructed current frame (not shown). If the current frame (**705**) was a predicted frame, the reconstructed prediction residual is added to the motion-compensated current frame (**735**) to form the reconstructed current frame. The frame store (**720**) buffers the reconstructed current frame for use in predicting the next frame. In some embodiments, the encoder applies a deblocking filter to the reconstructed frame to adaptively smooth discontinuities in the blocks of the frame.

[**0086**] The entropy coder (**780**) compresses the output of the quantizer (**770**) as well as certain side information (e.g., motion information (**715**), spatial extrapolation modes, quantization step size). 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 (**780**) 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.

[**0087**] The entropy coder (**780**) puts compressed video information (**795**) in the buffer (**790**). A buffer level indicator is fed back to bitrate adaptive modules.

[**0088**] The compressed video information (**795**) is depleted from the buffer (**790**) at a constant or relatively constant bitrate and stored for subsequent streaming at that bitrate. Therefore, the level of the buffer (**790**) 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 (**700**) streams compressed video information immediately following compression, and the level of the buffer (**790**) also depends on the rate at which information is depleted from the buffer (**790**) for transmission.

[**0089**] Before or after the buffer (**790**), the compressed video information (**795**) can be channel coded for transmission over the network. The channel coding can apply error detection and correction data to the compressed video information (**795**).

[**0090**] B. Video Decoder

[**0091**] **FIG. 8** is a block diagram of a general video decoder system (**800**). The decoder system (**800**) receives information (**895**) for a compressed sequence of video frames and produces output including a reconstructed frame (**805**). Particular embodiments of video decoders typically use a variation or supplemented version of the generalized decoder (**800**).

[**0092**] The decoder system (**800**) decompresses predicted frames and key frames. For the sake of presentation, **FIG. 8** shows a path for key frames through the decoder system (**800**) and a path for forward-predicted frames. Many of the components of the decoder system (**800**) 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.

[**0093**] A buffer (**890**) receives the information (**895**) for the compressed video sequence and makes the received information available to the entropy decoder (**880**). The buffer (**890**) 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 (**890**) can include a playback buffer and other buffers as well. Alternatively, the buffer (**890**) receives information at a varying rate. Before or after the buffer (**890**), the compressed video information can be channel decoded and processed for error detection and correction.

[**0094**] The entropy decoder (**880**) entropy decodes entropy-coded quantized data as well as entropy-coded side information (e.g., motion information (**815**), spatial extrapolation modes, quantization step size), 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 (**880**) 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.

[**0095**] If the frame (**805**) to be reconstructed is a forward-predicted frame, a motion compensator (**830**) applies motion information (**815**) to a reference frame (**825**) to form a prediction (**835**) of the frame (**805**) being reconstructed. For example, the motion compensator (**830**) uses a macroblock motion vector to find a macroblock in the reference frame (**825**). A frame buffer (**820**) stores previous reconstructed frames for use as reference frames. The motion compensator (**830**) 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 **(800)** also reconstructs prediction residuals.

[0096] When the decoder needs a reconstructed frame for subsequent motion compensation, the frame store **(820)** buffers the reconstructed frame for use in predicting the next frame. In some embodiments, the encoder applies a deblocking filter to the reconstructed frame to adaptively smooth discontinuities in the blocks of the frame.

[0097] An inverse quantizer **(870)** 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. Alternatively, the inverse quantizer applies another type of inverse quantization to the data, for example, a non-uniform, vector, or non-adaptive quantization, or directly inverse quantizes spatial domain data in a decoder system that does not use inverse frequency transformations.

[0098] An inverse frequency transformer **(860)** converts the quantized, frequency domain data into spatial domain video information. For block-based video frames, the inverse frequency transformer **(860)** 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 **(860)** applies another conventional inverse frequency transform such as a Fourier transform or uses wavelet or subband synthesis. In embodiments in which the decoder uses spatial extrapolation (not shown in **FIG. 8**) to decode blocks of key frames, the inverse frequency transformer **(860)** can apply a re-oriented inverse frequency transform such as a skewed IDCT to blocks of prediction residuals for the key frame. In other embodiments, the inverse frequency transformer **(860)** applies an 8×8, 8×4, 4×8, or other size inverse frequency transforms (e.g., IDCT) to prediction residuals for predicted frames.

III. Intraframe Encoding and Decoding

[0099] In one or more embodiments, a video encoder exploits redundancies in typical still images in order to code the I-frame information using a smaller number of bits. For additional detail about intraframe encoding and decoding in some embodiments, see U.S. patent application Ser. No. aa/bbb,ccc, entitled "Spatial Extrapolation of Pixel Values in Intraframe Video Coding and Decoding," filed concurrently herewith.

IV. Interframe Encoding and Decoding

[0100] Inter-frame coding exploits temporal redundancy between frames to achieve compression. Temporal redundancy reduction uses previously coded frames as predictors when coding the current frame.

[0101] A. Motion Estimation

[0102] In one or more embodiments, a video encoder exploits temporal redundancies in typical video sequences in order to code the information using a smaller number of bits. The video encoder uses motion estimation/compensation of a macroblock or other set of pixels of a current frame with respect to a reference frame. A video decoder uses corresponding motion compensation. Various features of the motion estimation/compensation can be used in combination or independently. These features include, but are not limited to:

[0103] 1a) Adaptive switching of the resolution of motion estimation/compensation. For example, the resolution switches between quarter-pixel and half-pixel resolutions.

[0104] 1b) Adaptive switching of the resolution of motion estimation/compensation depending on a video source with a closed loop or open loop decision.

[0105] 1c) Adaptive switching of the resolution of motion estimation/compensation on a frame-by-frame basis or other basis.

[0106] 2a) Using previously computed results of a first motion resolution evaluation to speed up a second motion resolution evaluation.

[0107] 2b) Selectively using integer-pixel motion information from a first motion resolution evaluation to speed up a second motion resolution evaluation.

[0108] 2c) Using previously computed sub-pixel values from a first motion resolution evaluation to speed up a second motion resolution evaluation.

[0109] 3) Using a search range with different directional resolution for motion estimation. For example, the horizontal resolution of the search range is quarter pixel and the vertical resolution is half pixel. This speeds up motion estimation by skipping certain quarter-pixel locations.

[0110] 4) Using a motion information representation with different bit allocation for horizontal and vertical motion. For example, a video encoder uses an additional bit for motion information in the horizontal direction, compared to the vertical direction.

[0111] 5a) Using a resolution bit with a motion information representation for additional resolution of motion estimation/compensation. For example, a video encoder adds a bit to half-pixel motion information to differentiate between a half-pixel increment and a quarter-pixel increment. A video decoder receives the resolution bit.

[0112] 5b) Selectively using a resolution bit with a motion information representation for additional resolution of motion estimation/compensation. For example, a video encoder adds a bit to half-pixel motion information to differentiate between a half-pixel increment and a quarter-pixel increment only for half-pixel motion information, not integer-pixel motion information. A video decoder selectively receives the resolution bit.

[0113] For motion estimation, the video encoder establishes a search range within the reference frame. The video encoder can center the search range around a predicted location that is set based upon the motion information for neighboring sets of pixels. In some embodiments, the encoder uses a reduced coverage range for the higher resolution motion estimation (e.g., quarter-pixel motion estimation) to balance between the bits used to signal the higher resolution motion information and distortion reduction due to the higher resolution motion estimation. Most motions observed in TV and movie content tends to be dominated by finer horizontal motion than vertical motion. This is probably due to the fact that most camera movements tend to be

more horizontal, since rapid vertical motion seems to make viewers dizzy. Taking advantage of this characteristic, the encoder uses higher resolution motion estimation/compensation that covers more horizontal locations than vertical locations. This strikes a balance between rate and distortion, and lowers the computational complexity of the motion information search process as well. In alternative embodiments, the search range has the same resolution horizontally and vertically.

[0114] Within the search range, the encoder finds a motion vector that parameterizes the motion of a macroblock or other set of pixels in the predicted frame. In some embodiments, with an efficient and low complexity method, the encoder computes and switches between higher sub-pixel accuracy and lower sub-pixel accuracy. In alternative embodiments, the encoder does not switch between resolutions for motion estimation/compensation. Instead of motion vectors (translations), the encoder can compute other types motion information to parameterize motion of a set of pixels between frames.

[0115] In one implementation, the encoder switches between quarter-pixel accuracy using a combination of four taps/two taps filter, and half-pixel accuracy using a two-tap filter. The encoder switches resolution of motion estimation/compensation on a per frame basis, per sequence basis, or other basis. The rationale behind this is that quarter-pixel motion compensation works well for very clean video sources (i.e., no noise), while half-pixel motion compensation handles noisy video sources (e.g., video from a cable feed) much better. This is due to the fact that the two-tap filter of the half-pixel motion compensation acts as a low-pass filter and tends to attenuate the noise. In contrast, the four-tap filter of the quarter-pixel motion compensation has some highpass effects so it can preserve the edges, but, unfortunately, it also tends to accentuate the noise. Other implementations use different filters.

[0116] After the encoder finds a motion vector or other motion information, the encoder outputs the information. For example, the encoder outputs entropy-coded data for the motion vector, motion vector differentials, or other motion information. In some embodiments, the encoder uses a motion vector with different bit allocation for horizontal and vertical motion. An extra bit adds quarter-pixel resolution horizontally to a half-pixel motion vector. The encoder saves bits by coding vertical motion vector at half-pixel accuracy. The encoder can add the bit only for half-pixel motion vectors, not for integer-pixel motion vectors, which further reduces the overall bitrate. In alternative embodiments, the encoder uses the same bit allocation for horizontal and vertical motions.

[0117] 1. Resolution Switching

[0118] In some embodiments, a video encoder switches resolution of motion estimation/compensation. **FIG. 9** shows a technique for selecting a motion estimation resolution for a predicted video frame. The encoder selects between half-pixel resolution and quarter-pixel resolution for motion vectors on a per frame basis. For the sake of simplicity, **FIG. 9** does not show the various ways in which the technique (**900**) can be used in conjunction with other techniques. In alternative embodiments, the encoder switches between resolutions other than quarter and half-pixel and/or switches at a frequency other than per frame.

[0119] The encoder gets (**910**) a macroblock for a predicted frame and computes (**920**) a half-pixel motion vector for the macroblock. The encoder also computes (**930**) a quarter-pixel motion vector for the macroblock. The encoder evaluates (**940**) the motion vectors. For example, for each of the motion vectors, the encoder computes an error measure such as sum of absolute differences ["SAD"], mean square error ["MSE"], a perceptual distortion measure, or another measure for the prediction residual.

[0120] In one implementation, the encoder computes and evaluates motion vectors as shown in FIG **10***a*. For a macroblock, the encoder computes (**1010**) a half-pixel motion vector MV_h in integer-pixel accuracy. For example, the encoder finds a motion vector by searching at integer increments within the search range. The encoder then computes (**1020**) MV_h to half-pixel accuracy in a region around the first computed MV_h.

[0121] In a separate path, the encoder computes (**1050**) a quarter-pixel motion vector MV_q in integer-pixel accuracy and then computes (**1070**) MV_q to quarter-pixel accuracy in a region around the first computed MV_q. The encoder then evaluates (**1090**) the final MV_h and MV_q. Alternatively, the encoder evaluates the motion vectors later.

[0122] In another implementation, the encoder eliminates a computation of a motion vector at integer-pixel accuracy in many cases by computing motion vectors as shown in **FIG. 10***b*. The encoder computes (**1010**) MV_h to integer-pixel accuracy.

[0123] Most of the time the integer-pixel portion of the MV_q is the same as the integer-pixel portion of MV_h. Thus, instead of computing the MV_q to integer-pixel accuracy every time as in **FIG. 1O***a*, the encoder checks (**1030**) whether the integer-pixel accurate MV_h can be used for MV_q. Specifically, the encoder checks whether integer-pixel accurate MV_h lies within the motion vector search range for the set of quarter-pixel motion vectors. The motion vector search range for a given macroblock is set to be ±16 (R in **FIG. 10**) of a motion vector predictor for the quarter-pixel motion vector. The motion vector predictor for a macroblock is the component-wise median of the macroblock's left, top, and top-right neighboring macroblocks' motion vectors, and can be different for MV_h and MV_q. Alternatively, the range, motion vector predictor, or conditional bypass is computed differently.

[0124] If the integer-pixel MV_h lies within the range then the encoder skips the computation of the integer-pixel MV_q, and simply sets (**1040**) MV_q to MV_h. Otherwise, the encoder computes (**1050**) MV_q to integer-pixel accuracy. The encoder computes (**1020**) MV_h to half-pixel accuracy, computes (**1070**) MV_q to quarter-pixel accuracy, and evaluates (**1070**) the motion vectors. Alternatively, the encoder computes the quarter-pixel motion vector at integer-pixel accuracy first, and selectively bypasses the computation of the half-pixel motion vector at integer-pixel accuracy.

[0125] Returning to **FIG. 9**, the encoder determines (**950**) whether there are any more macroblocks in the frame. If so, the encoder gets (**960**) the next macroblock and computes motion vectors for it.

[0126] Otherwise, the encoder selects (**970**) the motion vector resolution for the predicted frame. In one implementation, the encoder uses a rate-distortion criterion to select

the set of MV_h's or the set of MV_q's. The encoder compares the cost of choosing half-pixel resolution versus quarter-pixel resolution and picks the minimum of the two.

[0127] The cost functions are defined as follows:

$$J_q=SAD_q+QP*iMvBitOverhead$$

$$J_h=SAD_h$$

[0128] where J_h and J_q are the cost of choosing half-pixel resolution and quarter-pixel resolution, respectively. SAD_h and SAD_q are the sums of the residual error from prediction using the half-pixel and quarter-pixel motion vectors, respectively. QP is a quantization parameter. The effect of QP is to bias the selection in favor of half-pixel resolution in cases where QP is high and distortion in residuals would offset gains in quality from the higher resolution motion estimation. iMvBitOverhead is the extra bits for coding quarter-pixel motion vectors compared to the half-pixel motion vectors. In an implementation in which half-pixel motion vectors (but not integer-pixel motion vectors) have an extra resolution bit, iMvBitOverhead is the number of non-integer-pixel motion vectors in the set of MV_qs. Alternatively, the encoder uses other costs functions, for example, cost functions that directly compare the bits spent for different resolutions of motion vectors.

[0129] 2. Different Horizontal and Vertical Resolutions

[0130] In some embodiments, a video encoder uses a search range with different horizontal and vertical resolutions. For example, the horizontal resolution of the search range is quarter pixel and the vertical resolution of the search range is half pixel.

[0131] The encoder finds an integer-pixel accurate motion vector in a search range, for example, by searching at integer increments within the search range. In a region around the integer-pixel accurate motion vector, the encoder computes a sub-pixel accurate motion vector by evaluating motion vectors at sub-pixel locations in the region.

[0132] **FIG. 11** shows a location I that is pointed to by an integer-pixel accurate motion vector. The encoder computes a half-pixel motion vector by searching for the best match among all eight half-pixel locations H_0 to H_7 surrounding the integer position I. On the other hand, the encoder computes the quarter-pixel motion vector by searching for the best match among the eight half-pixel locations H_0 to H_7 and eight quarter-pixel locations Q_0 to Q_7. The searched quarter-pixel locations are placed horizontally between adjacent half-pixel locations. The searched quarter-pixel locations are not placed vertically between adjacent half-pixel locations. Thus, the search density increases on horizontal quarter-pixel locations, but not vertical quarter-pixel locations. This feature improves performance by speeding up the motion estimation process compared to a search in each direction by quarter-pixel increments, which would also require the computation of values for additional quarter-pixel locations.

[0133] In an implementation in which quarter-pixel resolution is indicated by adding an extra bit to half-pixel motion vectors, the quarter-pixel location to the right of the integer-pixel location is not searched as a valid location for a quarter-pixel motion vector, although a sub-pixel value is computed there for matching purposes. In other implementations, that quarter-pixel location is also searched and a different scheme is used to represent quarter-pixel motion vectors. In alternative embodiments, the encoder uses a different search pattern for quarter-pixel motion vectors.

[0134] The encoder generates values for sub-pixel locations by interpolation. In one implementation, for each searched location, the interpolation filter differs depending on the resolution chosen. For half-pixel resolution, the encoder uses a two-tap bilinear filter to generate the match, while for quarter-pixel resolution, the encoder uses a combination of four-tap and two-tap filters to generate the match. **FIG. 12** shows sub-pixel locations H_0, H_1, $H_2$with values computed by interpolation of integer-pixel values a, b, c, . . . , p.

[0135] For half-pixel resolution, the interpolation used in the three distinct half-pixel locations H_0, H_1, H_2 is:

$$H_0=(f+g+1-iRndCtrl)>>1.$$

$$H_1=(f+j+1-iRndCtrl)>>1.$$

$$H_2=(f+g+j+k+2-iRndCtrl)>>2.$$

[0136] where iRndCtrl indicates rounding control and varies between 0 and 1 from frame to frame.

[0137] For quarter-pixel resolution, the interpolation used for the three distinct half-pixel locations H_0, H_1, H_2 is:

$$H_0=(-e+9f+9g-h+8)>>4.$$

$$H_1=(-b+9f+9j-n+8)>>4.$$

$$H_2=(-t_0+9t_1+9t_2-t_3+8)>>4.$$

[0138] where **t0**, **t1**, **t2**, **t3** are computed as follows:

$$t_0=(-a+9b+9c-d+8)>>4$$

$$t_1=(-e+9f+9g-h+8)>>4$$

$$t_2=(-i+9j+9k-l+8)>>4$$

$$t_3=(-m+9n+9o-p+8)>>4$$

[0139] For the quarter-pixel resolution, the encoder also searches some of the quarter-pixel locations, as indicated by Q_0 to Q_7 in **FIG. 11**. These quarter-pixel locations are situated horizontally in between either two half-pixel locations or an integer-pixel location and a half-pixel location. For these quarter-pixel locations, the encoder uses bilinear interpolation (i.e., (x+y+1)>>1) using the two horizontally neighboring half-pixel/integer-pixel locations without rounding control. Using bicubic interpolation followed by bilinear interpolation balances computational complexity and information preservation, giving good results for reasonable computational complexity.

[0140] Alternatively, the encoder uses filters with different numbers or magnitudes of taps. In general, bilinear interpolation smoothes the values, attenuating high frequency information, whereas bicubic interpolation preserves more high frequency information but can accentuate noise. Using two bilinear steps (one for half-pixel locations, the second for quarter-pixel locations) is simple, but can smooth the pixels too much for efficient motion estimation.

[0141] 3. Encoding and Decoding Motion Vector Information

[0142] In some embodiments, a video encoder uses different bit allocation for horizontal and vertical motion vectors. For example, the video encoder uses one or more extra bits to represent motion in one direction with finer resolution that motion in another direction. This allows the encoder to reduce bitrate for vertical resolution information

that is less useful for compression, compared to systems that code motion information at quarter-pixel resolution both horizontally and vertically.

[0143] In one implementation, a video encoder uses an extra bit for quarter-pixel resolution of horizontal component motion vectors for macroblocks. For vertical component motion vectors, the video encoder uses half-pixel vertical component motion vectors. The video encoder can also use integer-pixel motion vectors. For example, the encoder outputs one or more entropy codes or another representation for a horizontal component motion vector and a vertical component motion vector. The encoder also outputs an additional bit that indicates a quarter-pixel horizontal increment. A value of 0 indicates no quarter-pixel increment and a value of 1 indicates a quarter-pixel increment, or vice versa. In this implementation, the use of the extra bit avoids the use of separate entropy code tables for quarter-pixel MVs/DMVs and half-pixel MVs/DMVs, and also adds little to bitrate.

[0144] In another implementation, a video encoder selectively uses the extra bit for quarter-pixel resolution of horizontal component motion vectors for macroblocks. The encoder adds the extra bit only if 1) quarter-pixel resolution is used for the frame and 2) at least one of the horizontal or vertical component motion vectors for a macroblock has half-pixel resolution. Thus, the extra bit is not used when quarter-pixel resolution is not used for a frame or when the motion vector for the macroblock is integer-pixel resolution, which reduces overall bitrate. Alternatively, the encoder adds the extra bit based upon other criteria.

[0145] **FIG. 13** shows a technique for decoding information for motion vectors at selective resolution. For the sake of simplicity, **FIG. 13** does not show the various ways in which the technique (**1300**) can be used in conjunction with other techniques.

[0146] A decoder gets (**1310**) motion vector information for a macroblock, for example, receiving one or more entropy codes or other information for a motion vector, component motion vectors, differential motion vectors ("DMVs"), or differential component motion vectors.

[0147] The decoder determines (**1330**) whether it has received all of the motion vector information for the macroblock. For example, the decoder determines whether additional resolution is enabled for the macroblock (e.g., at a frame level). Or, the decoder determines from decoding of the already received motion vector information whether to expect additional information. Or, the encoder considers both whether the additional resolution is enabled and whether to expect it based upon previously decoded information.

[0148] If the decoder expects additional motion vector resolution information, the decoder gets (**1340**) the additional information. For example, the decoder gets one or more additional resolution bits for the motion vector information for the macroblock.

[0149] The decoder then reconstructs (**1350**) the macroblock using the motion vector information and determines (**1360**) whether there are other macroblocks in the frame. If not, the technique ends. Otherwise, the decoder gets (**1370**) the motion vector information for the next macroblock and continues.

[0150] B. Coding of Prediction Residuals

[0151] Motion estimation is rarely perfect, and the video encoder uses prediction residuals to represent the differences between the original video information and the video information predicted using motion estimation. In one or more embodiments, a video encoder exploits redundancies in prediction residuals in order to code the information using a smaller number of bits. For additional detail about coding of prediction residuals in some embodiments, see U.S. patent application Ser. No. aa/bbb,ccc, entitled "Sub-Block Transform Coding of Prediction Residuals," filed concurrently herewith.

[0152] C. Loop Filtering

[0153] Quantization and other lossy processing of prediction residuals can cause blocky artifacts in reference frames that are used for motion estimation/compensation for subsequent predicted frames. In one or more embodiments, a video encoder processes a reconstructed frame to reduce blocky artifacts prior to motion estimation using the reference frame. A video decoder processes the reconstructed frame to reduce blocky artifacts prior to motion compensation using the reference frame. With deblocking, a reference frame becomes a better reference candidate to encode the following frame. Thus, using the deblocking filter improves the quality of motion estimation/compensation, resulting in better prediction and lower bitrate for prediction residuals. For additional detail about using a deblocking filter in motion estimation/compensation in some embodiments, see U.S. patent application Ser. No. aa/bbb,ccc, entitled "Motion Compensation Loop With Filtering," filed concurrently herewith.

[0154] Having described and illustrated the principles of our invention with reference to various embodiments, it will be recognized that the various embodiments can be modified in arrangement and detail without departing from such principles. It should be understood that the programs, processes, or methods described herein are not related or limited to any particular type of computing environment, unless indicated otherwise. Various types of general purpose or specialized computing environments may be used with or perform operations in accordance with the teachings described herein. Elements of embodiments shown in software may be implemented in hardware and vice versa.

[0155] In view of the many possible embodiments to which the principles of our invention may be applied, 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. In a computer system, a computer-implemented method of exploiting temporal redundancy between plural video frames, the method comprising:

selecting a fractional pixel motion resolution from among plural different fractional pixel motion resolutions, wherein each of the plural different fractional pixel motion resolutions is less than single integer-pixel motion resolution, and wherein each of the plural different fractional pixel motion resolutions is associated with a different reference frame sub-pixel interpolation technique; and

applying one or more motion vectors at the selected fractional pixel motion resolution to predict one or more pixels in a current frame of the plural video frames relative to one or more corresponding pixels in a reference frame of the plural video frames.

2. The method of claim 1 wherein the plural different fractional pixel motion resolutions include a quarter-pixel motion resolution.

3. The method of claim 1 wherein each of the one or more motion vectors is for a macroblock.

4. The method of claim 1 wherein the selecting occurs on a per frame basis.

5. The method of claim 1 wherein the selecting occurs on a per sequence basis.

6. A computer-readable medium storing computer-executable instructions for causing the computer system to perform the method of claim 1 during video encoding.

7. The method of claim 1 wherein a video encoder performs the selecting based upon evaluation of the plural different fractional pixel motion resolutions.

8. The method of claim 1 wherein the selecting depends at least in part on a quantization factor.

9. A computer-readable medium storing computer-executable instructions for causing the computer system to perform the method of claim 1 during video decoding.

10. The method of claim 1 wherein a video decoder performs the selecting based upon information received from the encoder.

11. A computer-readable medium storing computer-executable instructions for causing a computer system programmed thereby to perform a method of exploiting temporal redundancy between video frames from a video source during video encoding, the method comprising:

selecting a motion resolution from among plural different motion resolutions, wherein the selecting depends at least in part upon evaluation of the plural different motion resolutions, wherein the evaluation includes a first motion estimation for a first motion resolution of the plural different motion resolutions and a second motion estimation for a second motion resolution of the plural different motion resolutions, and wherein the selecting further depends at least in part upon noise level of the video frames from the video source; and

applying motion information at the selected motion resolution to predict one or more pixels in a current frame of the plural video frames relative to one or more corresponding pixels in a reference frame of the plural video frames.

12. The computer-readable medium of claim 11 wherein the evaluation is a closed loop evaluation of the plural different motion resolutions.

13. The computer-readable medium of claim 11 wherein the evaluation is an open loop evaluation of the plural different motion resolutions.

14. The computer-readable medium of claim 11 wherein the selecting further depends at least in part upon a quantization factor.

15. A computer-readable medium storing computer-executable instructions for causing a computer system programmed thereby to perform a method of exploiting temporal redundancy between video frames during video encoding, the method comprising:

selecting a motion resolution from among plural different motion resolutions after different motion estimation for each of the plural different motion resolutions, wherein the selecting depends at least in part upon a quantization factor; and

applying motion information at the selected motion resolution to predict one or more pixels in a current frame of the plural video frames relative to one or more corresponding pixels in a reference frame of the plural video frames.

16. A computer-readable medium storing computer-executable instructions for causing a computer system programmed thereby to perform a method of exploiting temporal redundancy between video frames, the method comprising:

in a first evaluation for a first motion resolution, computing intermediate motion evaluation results for the first motion resolution; and

in a second evaluation for a second motion resolution, using the intermediate motion evaluation results in computation of final motion evaluation results for the second motion resolution.

17. The computer-readable medium of claim 16 wherein the first motion resolution is a half-pixel resolution, and wherein the second motion resolution is a quarter-pixel resolution.

18. The computer-readable medium of claim 16 wherein the intermediate motion evaluation results include an integer-pixel motion vector used as an intermediate motion vector result for the first motion resolution and for the second motion resolution.

19. The computer-readable medium of claim 18 wherein an encoder uses the integer-pixel motion vector when the integer-pixel motion vector falls within a search range for the second evaluation.

20. The computer-readable medium of claim 16 wherein the intermediate motion evaluation results include interpolated pixel values at half-pixel locations.

21. In a computer system, a computer-implemented method of exploiting temporal redundancy between plural video frames, the method comprising:

computing a pixel value at each of plural half-pixel sample positions in a reference frame of the plural video frames, the reference frame including plural pixel values at integer-pixel sample positions organized by row and column, wherein:

for each of the plural half-pixel sample positions in either an integer-pixel row or an integer-pixel column, the computed pixel value is a function of the pixel values at at least four integer-pixel sample positions, and

for each of the plural half-pixel sample positions in both a half-pixel row and a half-pixel column, the computed pixel value is a function of the pixel values at at least four half-pixel sample positions; and

computing a pixel value at each of plural quarter-pixel sample positions in the reference frame, wherein:

for each of the plural quarter-pixel sample positions, the computed pixel value is a function of the pixel values at two sample positions adjacent the quarter-

pixel sample position and on opposites sides of the quarter-pixel sample position, wherein each of the two sample positions is either an integer-pixel sample position or a half-pixel sample position.

22. The method of claim 21 wherein the pixel values are luminance values.

23. A computer-readable medium storing computer-executable instructions for causing the computer system to perform the method of claim 21 during video encoding.

24. A computer-readable medium storing computer-executable instructions for causing the computer system to perform the method of claim 21 during video decoding.

25. In a computer system, a computer-implemented method of exploiting temporal redundancy between plural video frames, the method comprising:

defining a motion prediction range in a reference frame of the plural video frames, wherein the defined motion prediction range has a horizontal motion resolution and a vertical motion resolution, and wherein the horizontal motion resolution is different than the vertical motion resolution; and

applying motion information for one or more pixels of a current frame of the plural video frames relative to one or more corresponding pixels in the defined motion prediction range in the reference frame.

26. The method of claim 25 wherein the horizontal motion resolution is finer than the vertical motion resolution.

27. The method of claim 26 wherein the horizontal motion resolution is quarter pixel and the vertical motion resolution is half pixel.

28. The method of claim 25 further comprising computing the motion information in a video encoder, wherein the video encoder interpolates pixel values at different numbers of sub-pixel locations horizontally and vertically.

29. The method of claim 25 wherein the motion information comprises motion vector information with different horizontal and vertical component resolutions.

30. A computer-readable medium storing computer-executable instructions for causing the computer system to perform the method of claim 25 during video encoding.

31. A computer-readable medium storing computer-executable instructions for causing the computer system to perform the method of claim 25 during video decoding.

32. In a computer system, a computer-implemented method of exploiting temporal redundancy between plural video frames, the method comprising:

defining a motion prediction range in a reference frame of the plural video frames, including using plural different sub-pixel interpolation filters in the reference frame; and

applying motion information for one or more pixels of a current frame of the plural video frames relative to one or more corresponding pixels in the defined motion prediction range in the reference frame, wherein the motion information includes a horizontal motion component and a vertical motion component, and wherein motion resolution of the vertical motion component is different than motion resolution of the horizontal motion component.

33. The method of claim 32 wherein an extra bit associated with the motion information in a bitstream increases the motion resolution of the horizontal motion component by a factor of 2.

34. The method of claim 32 further comprising checking whether differential motion resolution is enabled for the current frame and if so performing the applying.

35. The method of claim 32 further comprising checking whether differential motion resolution applies for the motion information and if so performing the applying.

36. A computer-readable medium storing computer-executable instructions for causing the computer system to perform the method of claim 32 during video encoding.

37. A computer-readable medium storing computer-executable instructions for causing the computer system to perform the method of claim 32 during video encoding.

* * * * *