Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2502004 C 2012/10/16

(11)(21) 2 502 004

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(86) Date de dépôt PCT/PCT Filing Date: 2003/08/07
(87) Date publication PCT/PCT Publication Date: 2004/06/24
(45) Date de délivrance/Issue Date: 2012/10/16
(85) Entrée phase nationale/National Entry: 2005/04/11
(86) N° demande PCT/PCT Application No.: US 2003/024953
(87) N° publication PCT/PCT Publication No.: 2004/054257
(30) Priorité/Priority: 2002/12/06 (US10/313,773)

(51) Cl.Int./Int.Cl. *H04N 7/12* (2006.01), *G06K 9/36* (2006.01), *G06T 9/00* (2006.01), *H04N 7/26* (2006.01), *H04N 7/36* (2006.01), *H04N 9/74* (2006.01)

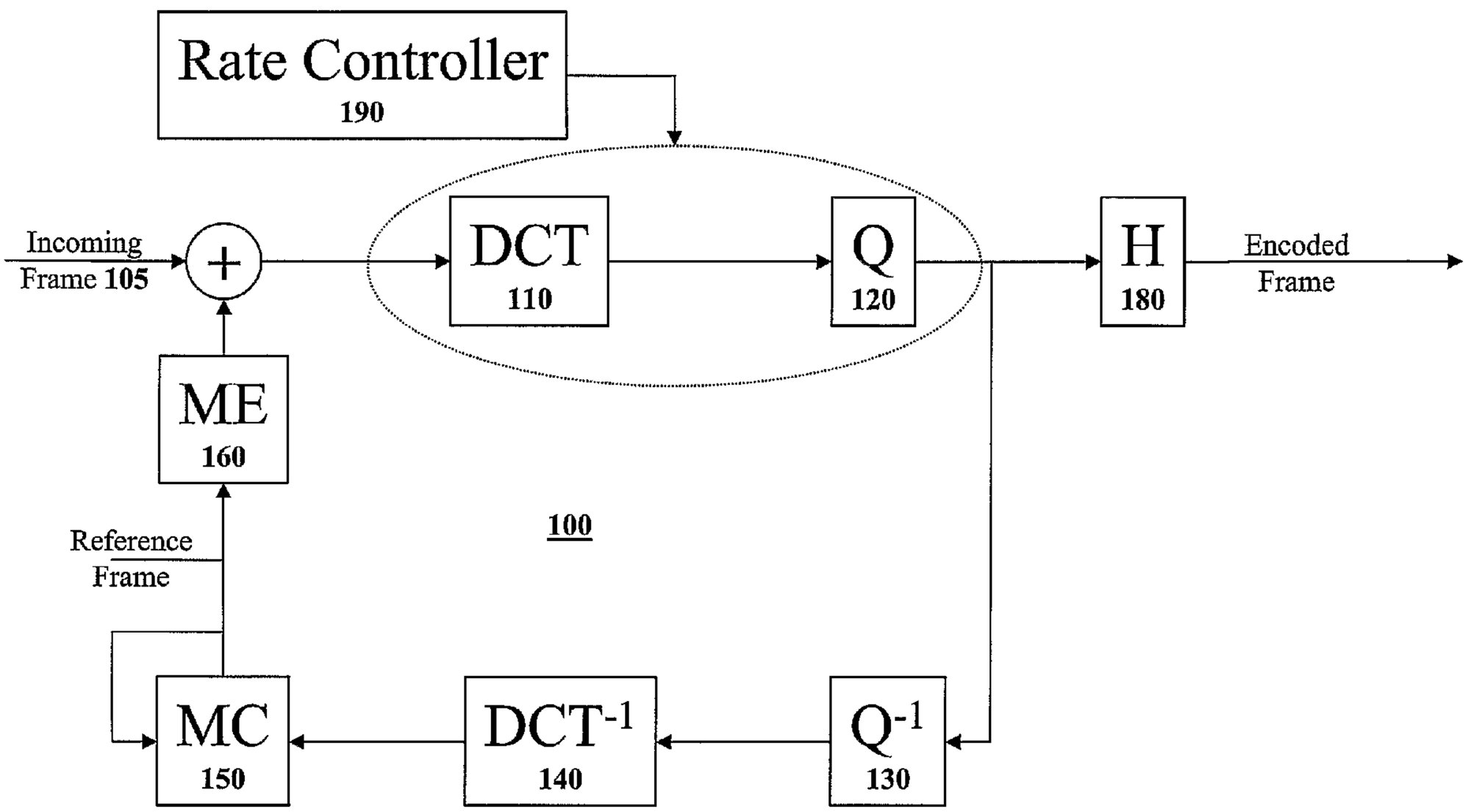
(72) Inventeurs/Inventors:
HASKELL, BARIN, G., US;
SINGER, DAVID, W., US;
DUMITRAS, ADRIANA, US;
PURI, ATUL, US

(73) Propriétaire/Owner:
APPLE INC., US

(74) Agent: RICHES, MCKENZIE & HERBERT LLP

(54) Titre : PROCEDE ET APPAREIL DE SPECIFICATION DE MINUTAGE ENTRE IMAGES A PRECISION VARIABLE POUR CODAGE VIDEO NUMERIQUE A EXIGENCES REDUITES POUR DES OPERATIONS DE DIVISION

(54) Title: METHOD AND APPARATUS FOR VARIABLE ACCURACY INTER-PICTURE TIMING SPECIFICATION FOR DIGITAL VIDEO ENCODING WITH REDUCED REQUIREMENTS FO DIVISION OPERATIONS



(57) Abrégé/Abstract:
A method and apparatus for performing motion estimation in a digital video system is disclosed (Fig.1). Specifically, the present invention discloses a system that quickly calculates estimated motion vectors in a very efficient manner (Fig.1, item 160). In one embodiment, a first multiplicand is determined by multiplying a first display time difference between a first video picture and a second video picture by a power of two scale value (Fig.1, items 150, 160). This step scales up a numerator for a ratio (Fig.1, item 120). Next, the system determines a scaled ratio by divi ding that scaled numerator by a second first display time difference between the second video picture and a third video picture. The scaled ratio is then stored calculating motion vector estimations. By storing the scaled ratio, all the estimated motion vectors can be calculated quickly with good precision since the scaled ratio saves significant bits and reducing the scale is performed by simple shifts (Fig.1, item 180).

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 June 2004 (24.06.2004)

PCT

(10) International Publication Number
WO 2004/054257 A1

(51) International Patent Classification[7]: H04N 7/12, 9/74, G06K 9/36

(21) International Application Number: PCT/US2003/024953

(22) International Filing Date: 7 August 2003 (07.08.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/313,773 6 December 2002 (06.12.2002) US

(71) Applicant: **APPLE COMPUTER, INC.** [US/US]; One Infinite Loop, Mail Stop: 38-PAT, Cupertino, CA 95014 (US).

(72) Inventors: **HASKELL, Barin, G.**; 1190 Fairbrook Dr., Mountain View, CA 94040-3960 (US). **SINGER, David, W.**; 268 Wawona Street, San Francisco, CA 94127-1328 (US). **DUMITRAS, Adriana**; 250 W. El Camino Real, Apt. 2403, Sunnyvale, CA 94087 (US). **PARI, Atul**; 19500 Pruneridge Ave, #4203, Cupertino, CA 95014-0632 (US).

(74) Agent: **JOHANSEN, Dag**; Stattler Johansen & Adeli LLP, P.O. Box 51860, Palo Alto, CA 94303-0728 (US).

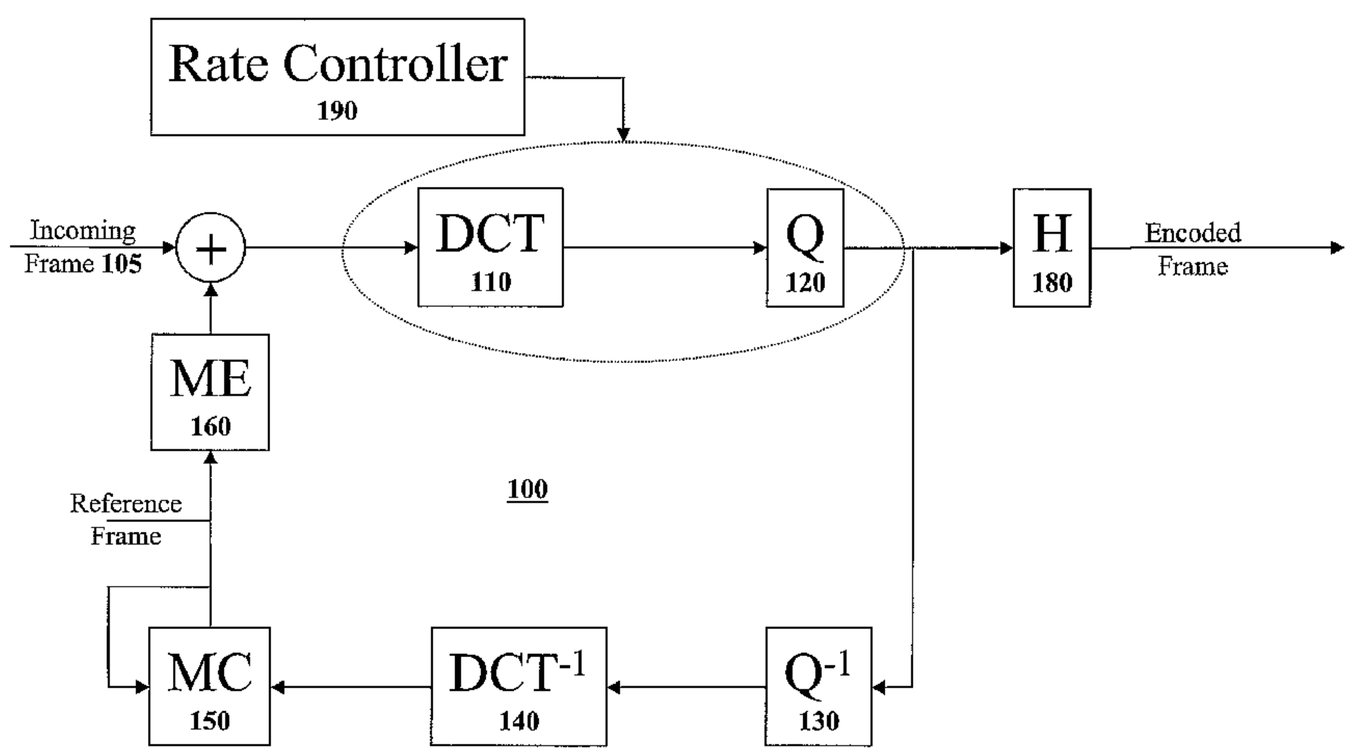
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHOD AND APPARATUS FOR VARIABLE ACCURACY INTER-PICTURE TIMING SPECIFICATION FOR DIGITAL VIDEO ENOCODING WITH REDUCED REQUIREMENTS FOR DIVISION OPERATIONS



(57) Abstract: A method and apparatus for performing motion estimation in a digital video system is disclosed (Fig.1). Specifically, the present invention discloses a system that quickly calculates estimated motion vectors in a very efficient manner (Fig.1, item 160). In one embodiment, a first multiplicand is determined by multiplying a first display time difference between a first video picture and a second video picture by a power of two scale value (Fig.1, items 150, 160). This step scales up a numerator for a ratio (Fig.1, item 120). Next, the system determines a scaled ratio by divi ding that scaled numerator by a second first display time difference between the second video picture and a third video picture. The scaled ratio is then stored calculating motion vector estimations. By storing the scaled ratio, all the estimated motion vectors can be calculated quickly with good precision since the scaled ratio saves significant bits and reducing the scale is performed by simple shifts (Fig.1, item 180).

WO 2004/054257 A1

Method and Apparatus for Variable Accuracy Inter-Picture Timing Specification for Digital Video Encoding With Reduced Requirements fo Division Operations

RELATED APPLICATIONS

This patent application claims the benefit of an earlier filing date under title 35, United States Code, Section 120 to the United States Patent Application having serial number 10/313,773 filed on December 6, 2002.

FIELD OF THE INVENTION

The present invention relates to the field of multimedia compression systems. In particular the present invention discloses methods and systems for specifying variable accuracy inter-picture timing with reduced requirements for processor intensive division operation.

BACKGROUND OF THE INVENTION

Digital based electronic media formats are finally on the cusp of largely replacing analog electronic media formats. Digital compact discs (CDs) replaced analog vinyl records long ago. Analog magnetic cassette tapes are becoming increasingly rare. Second and third generation digital audio systems such as Mini-discs and MP3 (MPEG Audio - layer 3) are now taking market share from the first generation digital audio format of compact discs.

The video media formats have been slower to move to digital storage and digital transmission formats than audio media. The reason for this slower digital adoption has been largely due to the massive amounts of digital information required to accurately represent acceptable quality video in digital form and the fast processing capabilities needed to encode compressed video. The massive amounts of digital information needed to accurately represent video require very high-capacity digital storage systems and high-bandwidth transmission systems.

However, video is now rapidly moving to digital storage and transmission formats. Faster computer processors, high-density storage systems, and new efficient compression and encoding algorithms have finally made digital video transmission and storage practical at consumer price points. The DVD (Digital Versatile Disc), a digital video system, has been one of the fastest selling consumer electronic products in years. DVDs have been rapidly supplanting Video-Cassette Recorders (VCRs) as the pre-recorded video playback system of choice due to their high video quality, very high audio quality, convenience, and extra features. The antiquated analog NTSC (National Television Standards Committee) video transmission system is currently in the process of being replaced with the digital ATSC (Advanced Television Standards Committee) video transmission system.

Computer systems have been using various different digital video encoding formats for a number of years. Specifically, computer systems have employed different video coder/decoder methods for compressing and encoding or decompressing

and decoding digital video, respectively. A video coder/decoder method, in hardware or software implementation, is commonly referred to as a "CODEC".

Among the best digital video compression and encoding systems used by computer systems have been the digital video systems backed by the Motion Pictures Expert Group commonly known by the acronym MPEG. The three most well known and highly used digital video formats from MPEG are known simply as MPEG-1, MPEG-2, and MPEG-4. VideoCDs (VCDs) and early consumer-grade digital video editing systems use the early MPEG-1 digital video encoding format. Digital Versatile Discs (DVDs) and the Dish Network brand Direct Broadcast Satellite (DBS) television broadcast system use the higher quality MPEG-2 digital video compression and encoding system. The MPEG-4 encoding system is rapidly being adapted by the latest computer based digital video encoders and associated digital video players.

The MPEG-2 and MPEG-4 standards compress a series of video frames or video fields and then encode the compressed frames or fields into a digital bitstream. When encoding a video frame or field with the MPEG-2 and MPEG-4 systems, the video frame or field is divided into a rectangular grid of pixelblocks. Each pixelblock is independently compressed and encoded.

When compressing a video frame or field, the MPEG-4 standard may compress the frame or field into one of three types of compressed frames or fields: Intra-frames (I-frames), Unidirectional Predicted frames (P-frames), or Bi-Directional Predicted frames (B-frames). Intra-frames completely independently encode an independent video frame with no reference to other video frames. P-frames define a

video frame with reference to a single previously displayed video frame. B-frames define a video frame with reference to both a video frame displayed before the current frame and a video frame to be displayed after the current frame. Due to their efficient usage of redundant video information, P-frames and B-frames generally provide the best compression.

SUMMARY OF THE INVENTION

A method and apparatus for performing motion estimation in a video codec is disclosed. Specifically, the present invention discloses a system that quickly calculates estimated motion vectors in a very efficient manner without requiring an excessive number of division operations.

In one embodiment, a first multiplicand is determined by multiplying a first display time difference between a first video picture and a second video picture by a power of two scale value. This step scales up a numerator for a ratio. Next, the system determines a scaled ratio by dividing that scaled numerator by a second first display time difference between said second video picture and a third video picture. The scaled ratio is then stored to be used later for calculating motion vector estimations. By storing the scaled ratio, all the estimated motion vectors can be calculated quickly with good precision since the scaled ratio saves significant bits and reducing the scale is performed by simple shifts thus eliminating the need for time consuming division operations.

In another aspect, the present invention provides for a sequence of video pictures comprising a first video picture, a second video picture, and a third video picture, a method comprising: computing a scaling value that is based on (i) a first order difference value between an order value for the third video picture and an order value for the first video picture, and (ii) a second order difference value between an order value for the second video picture and the order value for the first video picture, wherein an order value for a video picture specifies a display order for the video picture; and computing a motion vector

for the second video picture based on the scaling value and a motion vector for the third video picture, wherein computing the motion vector for the second video picture comprises performing a bit shifting operation.

In a further aspect, the present invention provides for a sequence of video pictures comprising a first video picture, a second video picture, and a third video picture, a method comprising: computing a scaling value that is based on (i) a power of two value, (ii) a first order difference value between an order value for the third video picture and an order value for the first video picture, and (iii) a second order difference value between an order value for the second video picture and the order value for the first video picture wherein the order value for the second video picture is encoded in a slice header associated with the second video picture; and computing a motion vector for the second video picture based on the scaling value and a motion vector for the third video picture, wherein computing the motion vector for the second video picture comprises performing a bit shifting operation.

In a still further aspect, the present invention provides a method comprising: computing a scaling value based on (i) a first order difference value between an order value for a first video picture and an order value for a second video picture, and (ii) a particular value that is based on a power of two value and a second order difference value between an order value for a third video picture and the order value for the second video picture, wherein the order value for the second video picture is encloded in a slice header of a bitstream associated with the second video picture; and using said scaling value to compute a motion vector for the second video picture by performing a bit shifting operation.

In a further aspect, the present invention provides a method comprising: a. receiving a plurality of video pictures and at least one order value; and b. computing an implicit B prediction block weighting for a video picture based on said at least one order value.

In a still further aspect, the present invention provides a method comprising: a. decoding a plurality of video pictures by using at least one order value, said order value is for establishing an ordering for reference video picture selection; and c. outputting the decoded video pictures based on the order value.

In a further aspect, the present invention provides a method of decoding coded video data, comprising: calculating, for a current frame of video data to be decoded, a scale factor (Z) based on a ratio of two time differences ($\Delta t_1/\Delta t_2$), the first time difference (Δt_1) representing a temporal difference between the current frame and a first reference frame and the second time difference (Δt_2) representing a temporal difference between the first reference frame and a second reference frame, the scale factor representing the ratio multiplied by a power of two ($Z=2^N*(\Delta t_1/\Delta t_2)$) subject to rounding, predicting data of pixelblocks of the current frame from data of the reference frames according to predictive coding techniques, further comprising, as part of the prediction, interpolating a motion

vector (mv_{PB}) of the pixelblocks from a motion vector (mv_{REF}) of a co-located pixelblock in one of the reference frames as $mv_{PB} = mv_{REF}*Z/2^N$.

In a still further aspect, the present invention provides a method of decoding coded video data, comprising: calculating, for a current frame of video data to be decoded, a scale factor (Z) based on a ratio of two time differences ($\Delta t_1/\Delta t_2$), the first time difference (Δt_1) representing a temporal difference between the current frame and a first reference frame and the second time difference (Δt_2) representing a temporal difference between the first reference frame and a second reference frame, the scale factor representing the ratio multiplied by a power of two ($Z=2^N*(\Delta t_1/\Delta t_2)$) subject to rounding, predicting data of pixelblocks of the current frame from data of the reference frames according to predictive coding techniques, as part of the prediction, determining whether motion vectors for the pixelblock are to be interpolated from motion vectors of the reference frames, if so, interpolating motion vectors ($mv_{PB[i]}$) of the respective pixelblocks from motion vector ($mv_{REF[i]}$) of a co-located pixelblock in one of the reference frames as $mv_{PB} = mv_{REF}*Z/2^N$ wherein the scale factor Z is common to motion vector derivations of all pixelblocks i in the current frame.

In a further aspect, the present invention provides a video decoder, comprising: a motion estimator to calculate, for a current frame of video data to be decoded, a scale factor (Z) based on a ratio of two time differences ($\Delta t_1/\Delta t_2$), the first time difference (Δt_1)

representing a temporal difference between the current frame and a first reference frame and the second time difference (Δt_2) representing a temporal difference between the first reference frame and a second reference frame, the scale factor representing the ratio multiplied by a power of two ($Z=2^N*(\Delta t_1/\Delta t_2)$) subject to rounding, a predictor to predict data of pixelblocks of the current frame from data of the reference frames according to predictive coding techniques, wherein the estimator further interpolates a motion vector (mv_{PB}) of the pixelblocks from a motion vector (mv_{REF}) of a co-located pixelblock in one of the reference frames as $mv_{PB} = mv_{REF}*Z/2^N$.

In a still further aspect, the present invention provides a video decoder, comprising: a motion estimator to calculate, for a current frame of video data to be decoded, a scale factor (Z) based on a ratio of two time differences ($\Delta t_1/\Delta t_2$), the first time difference (Δt_1) representing a temporal difference between the current frame and a first reference frame and the second time difference (Δt_2) representing a temporal difference between the first reference frame and a second reference frame, the scale factor representing the ratio multiplied by a power of two ($Z=2^N*(\Delta t_1/\Delta t_2)$) subject to rounding, a predictor to predict data of pixelblocks of the current frame from data of the reference frames according to predictive coding techniques, as part of the prediction, determining whether motion vectors for the pixelblock are to be interpolated from motion vectors of the reference frames, wherein the estimator further interpolates motion vectors ($mv_{PB[i]}$) of the respective pixelblocks from motion vector ($mv_{REF[i]}$) of a co-located pixelblock in one of the reference

frames as $mv_{PB} = mv_{REF}*Z/2^N$, wherein the scale factor Z is common to motion vector derivations of all pixelblocks i in the current frame.

In a further aspect, the present invention provides computer readable medium storing program instructions that, when executed by a processing device, cause the device to: calculate, for a current frame of video data to be decoded, a scale factor (Z) based on a ratio of two time differences ($\Delta t_1/\Delta t_2$), the first time difference (Δt_1) representing a temporal difference between the current frame and a first reference frame and the second time difference (Δt_2) representing a temporal difference between the first reference frame and a second reference frame, the scale factor representing the ratio multiplied by a power of two ($Z=2^N*(\Delta t_1/\Delta t_2)$) subject to rounding, predict data of pixelblocks of the current frame from data of the reference frames according to predictive coding techniques, and as part of the prediction, interpolate a motion vector (mv_{PB}) of the pixelblocks from a motion vector (mv_{REF}) of a co-located pixelblock in one of the reference frames as $mv_{PB} = mv_{REF}*Z/2^N$.

In a still further aspect, the present invention provides a method of coding a sequence of video data: interpolating motion vectors for a pixelblock in a current frame, the pixelblock to be coded according to bi-directional prediction with reference to a pair of reference frames in the sequence, wherein the interpolation includes for at least one motion vector: generating a scale factor (Z) based on a ratio of time differences ($\Delta t_1/\Delta t_2$) among

the current frame and the pair of reference frames, the first time difference (Δt_1) representing a temporal difference between the current frame and a first of the reference frame and the second time difference (Δt_2) representing a temporal difference between the two reference frames, the scale factor representing the ratio multiplied by a power of two ($Z=2^N*(\Delta t_1/\Delta t_2)$) subject to rounding, generating a first motion vector for the pixelblock (mv_{PB1}) according to a first derivation based on a motion vector extending between the reference frames (mv_{REF}), the first derivation being $mv_{PB1} = mv_{REF}*Z/2^N$, generating a second motion vector for the pixelblock (mv_{PB2}) based on a second derivation mv_{REF}, predicting video data for the pixelblock from video data of the first and second reference frames according to the generated motion vectors mv_{PB1} and mv_{PB2}, coding a difference between actual video data of the pixelblock and predicted video data of the pixel block, and outputting the coded difference to a channel as coded video data of the pixelblock.

In a further aspect, the present invention provides a method of coding a sequence of video data: coding a first reference frame at a first temporal location in the sequence, coding a second reference frame at a second temporal location in the sequence, interpolating motion vectors for a pixelblock in a third, current frame, the pixelblock to be coded according to bi-directional prediction with reference to the pair of reference frames, wherein the interpolation includes for at least one motion vector: generating a scale factor (Z) based on a ratio of time differences ($\Delta t_1/\Delta t_2$) among the current frame and the reference frames, the first time difference (Δt_1) representing a temporal difference between the

current frame and the first reference frame and the second time difference (Δt_2) representing a temporal difference between first and second reference frames, the scale factor representing the ratio multiplied by a power of two ($Z=2^N*(\Delta t_1/\Delta t_2)$) subject to rounding, generating a first motion vector for the pixelblock (mv_{PB1}) according to a first derivation based on a motion vector extending between the reference frames (mv_{REF}), the first derivation being $mv_{PB1} = mv_{REF}*Z/2^N$, generating a second motion vector for the pixelblock (mv_{PB2}) based on a second derivation of mv_{REF}, predicting video data for the pixelblock from video data of the first and second reference frames according to the generated motion vectors mv_{PB1} and mv_{PB2}, coding a difference between actual video data of the pixelblock and predicted video data of the pixel block, and outputting the coded difference to a channel as coded video data of the pixelblock.

In a still further aspect, the present invention provides a video coder, comprising: coding unit comprising a discrete cosine transform unit, a quantization unit to code an output of the discrete cosine transform unit, and an entropy coder to code an output of the quantization unit; motion estimator to interpolate motion vectors (mv_{PB1}, mv_{PB2}) for a pixelblock in a current frame, the motion vector estimator interpolating the motion vectors mv_{PB1}, mv_{PB2} with reference to a motion vector (mv_{REF}) extending from between previously-coded pair of reference frames by: generating a scale factor (Z) based on a ratio of time differences ($\Delta t_1/\Delta t_2$) among the current frame and the pair of reference frames, the first time difference (Δt_1) representing a temporal difference between the current frame and

a first of the reference frames and the second time difference (Δt_2) representing a temporal difference between the two reference frames, the scale factor representing the ratio multiplied by a power of two ($Z=2^N*(\Delta t_1/\Delta t_2)$) subject to rounding, generating a first motion vector for the pixelblock (mv_{PB1}) according to a first derivation based on a motion vector extending between the reference frames (mv_{REF}), the first derivation being $mv_{PB1} = mv_{REF}*Z/2^N$, generating a second motion vector for the pixelblock (mv_{PB2}) based on a second derivation mv_{REF}; and a video data predictor to predict video data for the pixelblock from video data of the first and second reference frames according to the generated motion vectors mv_{PB1} and mv_{PB2}; wherein the coding unit codes a difference between actual video data of the pixelblock and predicted video data of the pixel block, and outputs the coded difference to a channel as coded video data of the pixelblock.

In a further aspect, the present invention provides computer readable medium storing program instructions that, when executed by a processing device, cause the device to: interpolate motion vectors for a pixelblock in a current frame, the pixelblock to be coded according to bi-directional prediction with reference to a pair of reference frames in the sequence, wherein the interpolation includes for at least one motion vector: generating a scale factor (Z) based on a ratio of time differences ($\Delta t_1/\Delta t_2$) among the current frame and the pair of reference frames, the first time difference (Δt_1) representing a temporal difference between the current frame and a first of the reference frame and the second time difference (Δt_2) representing a temporal difference between the two reference frames, the

scale factor representing the ratio multiplied by a power of two ($Z=2^N*(\Delta t_1/\Delta t_2)$) subject to rounding, generating a first motion vector for the pixelblock (mv_{PB1}) according to a first derivation based on a motion vector extending between the reference frames (mv_{REF}), the first derivation being $mv_{PB1} = mv_{REF}*Z/2^N$, generating a second motion vector for the pixelblock (mv_{PB2}) based on a second derivation mv_{REF}, predict video data for the pixelblock from video data of the first and second reference frames according to the generated motion vectors mv_{PB1} and mv_{PB2}, code a difference between actual video data of the pixelblock and predicted video data of the pixel block, and output the coded difference to a channel as coded video data of the pixelblock.

In a still further aspect, the present invention provides computer readable medium storing program instructions that, when executed by a processing device, cause the device to: code a first reference frame at a first temporal location in the sequence, code a second reference frame at a second temporal location in the sequence, interpolate motion vectors for a pixelblock in a third, current frame, the pixelblock to be coded according to bi-directional prediction with reference to the pair of reference frames, wherein the interpolation includes for at least one motion vector: generating a scale factor (Z) based on a ratio of time differences ($\Delta t_1/\Delta t_2$) among the current frame and the reference frames, the first time difference (Δt_1) representing a temporal difference between the current frame and the first reference frame and the second time difference (Δt_2) representing a temporal difference between first and second reference frames, the scale factor representing the ratio

multiplied by a power of two ($Z=2^N*(\Delta t_1/\Delta t_2)$) subject to rounding, generating a first motion vector for the pixelblock (mv_{PB1}) according to a first derivation based on a motion vector extending between the reference frames (mv_{REF}), the first derivation being $mv_{PB1} = mv_{REF}*Z/2^N$, generating a second motion vector for the pixelblock (mv_{PB2}) based on a second derivation of mv_{REF}, predict video data for the pixelblock from video data of the first and second reference frames according to the generated motion vectors mv_{PB1} and mv_{PB2}, code a difference between actual video data of the pixelblock and predicted video data of the pixel block, and output the coded difference to a channel as coded video data of the pixelblock.

In a further aspect, the present invention provides a coded video signal created by a method, comprising: interpolating motion vectors for a pixelblock in a current frame, the pixelblock to be coded according to bi-directional prediction with reference to a pair of reference frames in the sequence, wherein the interpolation includes, for at least one motion vector: generating a scale factor (Z) based on a ratio of time differences ($\Delta t_1/\Delta t_2$) among the current frame and the pair of reference frames, the first time difference (Δt_1) representing a temporal difference between the current frame and a first of the reference frame and the second time difference (Δt_2) representing a temporal difference between the two reference frames, the scale factor representing the ratio multiplied by a power of two ($Z=2^N*(\Delta t_1/\Delta t_2)$) subject to rounding, generating a first motion vector for the pixelblock (mv_{PB1}) according to a first derivation based on a motion vector extending between the

reference frames (mv_{REF}), the first derivation being $mv_{PB1} = mv_{REF}*Z/2^N$, generating a second motion vector for the pixelblock (mv_{PB2}) based on a second derivation mv_{REF}, predicting video data for the pixelblock from video data of the first and second reference frames according to the generated motion vectors mv_{PB1} and mv_{PB2}, coding a difference between actual video data of the pixelblock and predicted video data of the pixel block, and outputting the coded difference to a channel as coded video data of the pixelblock.

In a still further aspect, the present invention provides an apparatus comprising: a storage for storing a stream comprising a first video picture, a second video picture, and a third video picture; and at least one processor for: (i) computing a scaling value that is based on (i) a first order difference value between an order value for the third video picture and an order value for the first video picture, and (ii) a second order difference value between an order value for the second video picture and the order value for the first video picture; and computing a particular motion vector for the second video picture based on the scaling value and a motion vector for the third video picture, wherein computing the particular motion vector comprises performing a bit shifting operation.

In a further aspect, the present invention provides an apparatus comprising: a storage for storing a stream comprising a first video picture, a second video picture, and a third video picture; and at least one processor for: computing a scaling value that is based on (i) a first order difference value between an order value for the third video picture and

an order value for the first video picture, and (ii) a second order difference value between an order value for the second video picture and the order value for the first video picture; and computing a particular motion vector for the second video picture based on the scaling value and a motion vector for the third video picture, wherein computing the particular motion vector comprises performing a division by a power of two value.

In a still further aspect, the present invention provides an apparatus comprising: at least one processor for: computing a scaling value based on (i) a first order difference value between an order value for a first video picture and an order value for a second video picture, and (ii) a particular value that is based on a power of two value and a second order difference value between an order value for a third video picture and the order value for the second video picture; and using said scaling value to compute a motion vector.

In a further aspect, the present invention provides an apparatus comprising: at least one processor for decoding a plurality of video pictures by using at least one order value, said order value is for establishing an ordering for reference video picture selection; and a storage for outputting the decoded video pictures based on the order value.

In a further aspect, the present invention provides for a sequence of video pictures comprising a first video picture, a second video picture, and a third video picture, the method comprising: computing a scaling value that is (i) inversely proportional to a first order difference value between an order value for the third video picture and an order value

for the first video picture and (ii) directly proportional to a second order difference value between an order value for the second video picture and the order value for the first video picture, wherein the order value for the second picture is encoded in a bitstream more than once; and computing a motion vector for the second video picture by multiplying the scaling value by a motion vector for the third video picture and performing a division operation that is based on a power of two value.

In a still further aspect, the present invention provides for a sequence of video pictures comprising a first video picture, a second video picture, and a third video picture, a method comprising: computing a scaling value that is (i) inversely proportional to a first order difference value between an order value for the third video picture and an order value for the first video picture, (ii) directly proportional to a second order difference value between an order value for the second video picture and the order value for the first video picture, and (iii) directly proportional to a power of two value, wherein an order value for a video picture specifies a display order for the video picture; and computing a motion vector for the second video picture by multiplying the scaling value and a motion vector for the third video picture and performing a division operation based on said power of two value.

In a further aspect, the present invention provides a method of decoding a bitstream comprising encoded first, second, and third video pictures, the method comprising: receiving an integer value representing an exponent of a particular power of two integer; computing a scaling value that is based on (i) a first order difference value between an

order value for the third video picture and an order value for the first video picture, and (ii) a second order difference value between an order value for the second video picture and the order value for the first video picture, wherein the order value for the second video picture is derived from the particular power of two integer; computing a motion vector for the second video picture by multiplying the scaling value by a motion vector for the third video picture and performing a bit shifting operation; and decoding the second video picture by using the computed motion vector.

In a still further aspect, the present invention provides a method for encoding a sequence of video pictures comprising a first video picture, a second video picture, and a third video picture, the method comprising: computing a scaling value that is based on (i) a first order difference value between an order value for the third video picture and an order value for the first video picture, and (ii) a second order difference value between an order value for the second video picture and the order value for the first video picture; computing a particular motion vector for the second video picture based on the scaling value and a motion vector for the third video picture, wherein computing the particular motion vector for the second video picture comprises performing a bit shifting operation; encoding the second video picture in a bitstream by using the computed motion vector; and encoding the order value for the second video picture in the bitstream by using an exponent of a power of integer.

Other objects, features, and advantages of present invention will be apparent from the company drawings and from the following detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

The objects, features, and advantages of the present invention will be apparent to one skilled in the art, in view of the following detailed description in which:

Figure 1 illustrates a high-level block diagram of one possible digital video encoder system.

Figure 2 illustrates a series of video pictures in the order that the pictures should be displayed wherein the arrows connecting different pictures indicate inter-picture dependency created using motion compensation.

Figure 3 illustrates the video pictures from **Figure 2** listed in a preferred transmission order of pictures wherein the arrows connecting different pictures indicate inter-picture dependency created using motion compensation.

Figure 4 graphically illustrates a series of video pictures wherein the distances between video pictures that reference each other are chosen to be powers of two.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A method and system for specifying Variable Accuracy Inter-Picture Timing in a multimedia compression and encoding system with reduced requirements for division operations is disclosed. In the following description, for purposes of explanation, specific nomenclature is set forth to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that these specific details are not required in order to practice the present invention. For example, the present invention has been described with reference to the MPEG multimedia compression and encoding system. However, the same techniques can easily be applied to other types of compression and encoding systems.

Multimedia Compression and Encoding Overview

Figure 1 illustrates a high-level block diagram of a typical digital video encoder **100** as is well known in the art. The digital video encoder **100** receives an incoming video stream of video frames **105** at the left of the block diagram. The digital video encoder **100** partitions each video frame into a grid of pixelblocks. The pixelblocks are individually compressed. Various different sizes of pixelblocks may be used by different video encoding systems. For example, different pixelblock resolutions include 8x8, 8x4, 16x8, 4x4, etc. Furthermore, pixelblocks are occasionally referred to as 'macroblocks.' This document will use the term pixelblock to refer to any block of pixels of any size.

A Discrete Cosine Transformation (DCT) unit **110** processes each pixelblock in the video frame. The frame may be processed independently (an intra-frame) or with reference to information from other frames received from the motion compensation unit (an inter-frame). Next, a Quantizer (Q) unit **120** quantizes the information from the Discrete Cosine Transformation unit **110**. Finally, the quantized video frame is then encoded with an entropy encoder (H) unit **180** to produce an encoded bitstream. The entropy encoder (H) unit **180** may use a variable length coding (VLC) system.

Since an inter-frame encoded video frame is defined with reference to other nearby video frames, the digital video encoder **100** needs to create a copy of how each decoded frame will appear within a digital video decoder such that inter-frames may be encoded. Thus, the lower portion of the digital video encoder **100** is actually a digital video decoder system. Specifically, an inverse quantizer (Q^{-1}) unit **130** reverses the quantization of the video frame information and an inverse Discrete Cosine Transformation (DCT^{-1}) unit **140** reverses the Discrete Cosine Transformation of the video frame information. After all the DCT coefficients are reconstructed from inverse Discrete Cosine Transformation (DCT^{-1}) unit **140**, the motion compensation unit will use that information, along with the motion vectors, to reconstruct the encoded video frame. The reconstructed video frame is then used as the reference frame for the motion estimation of the later frames.

The decoded video frame may then be used to encode inter-frames (P-frames or B-frames) that are defined relative to information in the decoded video frame. Specifically, a motion compensation (MC) unit **150** and a motion estimation (ME) unit

160 are used to determine motion vectors and generate differential values used to encode inter-frames.

A rate controller **190** receives information from many different components in a digital video encoder **100** and uses the information to allocate a bit budget for each video frame. The rate controller **190** should allocate the bit budget in a manner that will generate the highest quality digital video bit stream that that complies with a specified set of restrictions. Specifically, the rate controller **190** attempts to generate the highest quality compressed video stream without overflowing buffers (exceeding the amount of available memory in a video decoder by sending more information than can be stored) or underflowing buffers (not sending video frames fast enough such that a video decoder runs out of video frames to display).

Digital Video Encoding With Pixelblocks

In some video signals the time between successive video pictures (frames or fields) may not be constant. (Note: This document will use the term video pictures to generically refer to video frames or video fields.) For example, some video pictures may be dropped because of transmission bandwidth constraints. Furthermore, the video timing may also vary due to camera irregularity or special effects such as slow motion or fast motion. In some video streams, the original video source may simply have non-uniform inter-picture times by design. For example, synthesized video such as computer graphic animations may have non-uniform timing since no arbitrary video timing is imposed by a uniform timing video capture system such as a video camera system. A

flexible digital video encoding system should be able to handle non-uniform video picture timing.

As previously set forth, most digital video encoding systems partition video pictures into a rectangular grid of pixelblocks. Each individual pixelblock in a video picture is independently compressed and encoded. Some video coding standards, e.g., ISO MPEG or ITU H.264, use different types of predicted pixelblocks to encode video pictures. In one scenario, a pixelblock may be one of three types:

1. I-pixelblock – An Intra (I) pixelblock uses no information from any other video pictures in its coding (it is completely self-defined);
2. P-pixelblock – A unidirectionally predicted (P) pixelblock refers to picture information from one preceding video picture; or
3. B-pixelblock – A bi-directional predicted (B) pixelblock uses information from one preceding picture and one future video picture.

If all the pixelblocks in a video picture are Intra-pixelblocks, then the video picture is an Intra-frame. If a video picture only includes unidirectional predicted macro blocks or intra-pixelblocks, then the video picture is known as a P-frame. If the video picture contains any bi-directional predicted pixelblocks, then the video picture is known as a B-frame. For the simplicity, this document will consider the case where all pixelblocks within a given picture are of the same type.

An example sequence of video pictures to be encoded might be represented as:

$I_1\ B_2\ B_3\ B_4\ P_5\ B_6\ B_7\ B_8\ B_9\ P_{10}\ B_{11}\ P_{12}\ B_{13}\ I_{14}\ldots$

where the letter (I, P, or B) represents if the video picture is an I-frame, P-frame, or B-frame and the number represents the camera order of the video picture in the sequence of video pictures. The camera order is the order in which a camera recorded the video pictures and thus is also the order in which the video pictures should be displayed (the display order).

The previous example series of video pictures is graphically illustrated in **Figure 2**. Referring to **Figure 2**, the arrows indicate that pixelblocks from a stored picture (I-frame or P-frame in this case) are used in the motion compensated prediction of other pictures.

In the scenario of **Figure 2**, no information from other pictures is used in the encoding of the intra-frame video picture I_1. Video picture P_5 is a P-frame that uses video information from previous video picture I_1 in its coding such that an arrow is drawn from video picture I_1 to video picture P_5. Video picture B_2, video picture B_3, video picture B_4 all use information from both video picture I_1 and video picture P_5 in their coding such that arrows are drawn from video picture I_1 and video picture P_5 to video picture B_2, video picture B_3, and video picture B_4. As stated above the inter-picture times are, in general, not the same.

Since B-pictures use information from future pictures (pictures that will be displayed later), the transmission order is usually different than the display order. Specifically, video pictures that are needed to construct other video pictures should be transmitted first. For the above sequence, the transmission order might be:

$$I_1\ P_5\ B_2\ B_3\ B_4\ P_{10}\ B_6\ B_7\ B_8\ B_9\ P_{12}\ B_{11}\ I_{14}\ B_{13}\ \ldots$$

Figure 3 graphically illustrates the preceding transmission order of the video pictures from **Figure 2**. Again, the arrows in the figure indicate that pixelblocks from a stored video picture (I or P in this case) are used in the motion compensated prediction of other video pictures.

Referring to **Figure 3**, the system first transmits I-frame I_1 which does not depend on any other frame. Next, the system transmits P-frame video picture P_5 that depends upon video picture I_1. Next, the system transmits B-frame video picture B_2 after video picture P_5 even though video picture B_2 will be displayed before video picture P_5. The reason for this is that when it comes time to decode video picture B_2, the decoder will have already received and stored the information in video pictures I_1 and P_5 necessary to decode video picture B_2. Similarly, video pictures I_1 and P_5 are ready to be used to decode subsequent video picture B_3 and video picture B_4. The receiver/decoder reorders the video picture sequence for proper display. In this operation I and P pictures are often referred to as stored pictures.

The coding of the P-frame pictures typically utilizes Motion Compensation, wherein a Motion Vector is computed for each pixelblock in the picture. Using the computed motion vector, a prediction pixelblock (P-pixelblock) can be formed by translation of pixels in the aforementioned previous picture. The difference between the actual pixelblock in the P-frame picture and the prediction pixelblock is then coded for transmission.

P-Pictures

The coding of P-Pictures typically utilize Motion Compensation (MC), wherein a Motion Vector (MV) pointing to a location in a previous picture is computed for each pixelblock in the current picture. Using the motion vector, a prediction pixelblock can be formed by translation of pixels in the aforementioned previous picture. The difference between the actual pixelblock in the P-Picture and the prediction pixelblock is then coded for transmission.

Each motion vector may also be transmitted via predictive coding. For example, a motion vector prediction may be formed using nearby motion vectors. In such a case, then the difference between the actual motion vector and the motion vector prediction is coded for transmission.

B-Pictures

Each B-pixelblock uses two motion vectors: a first motion vector referencing the aforementioned previous video picture and a second motion vector referencing the future video picture. From these two motion vectors, two prediction pixelblocks are computed. The two predicted pixelblocks are then combined together, using some function, to form a final predicted pixelblock. As above, the difference between the actual pixelblock in the B-frame picture and the final predicted pixelblock is then encoded for transmission.

As with P-pixelblocks, each motion vector (MV) of a B-pixelblock may be transmitted via predictive coding. Specifically, a predicted motion vector is formed using

nearby motion vectors. Then, the difference between the actual motion vector and the predicted is coded for transmission.

However, with B-pixelblocks the opportunity exists for interpolating motion vectors from motion vectors in the nearest stored picture pixelblock. Such motion vector interpolation is carried out both in the digital video encoder and the digital video decoder.

This motion vector interpolation works particularly well on video pictures from a video sequence where a camera is slowly panning across a stationary background. In fact, such motion vector interpolation may be good enough to be used alone. Specifically, this means that no differential information needs be calculated or transmitted for these B-pixelblock motion vectors encoded using interpolation.

To illustrate further, in the above scenario let us represent the inter-picture display time between pictures i and j as $D_{i,j}$, i.e., if the display times of the pictures are T_i and T_j, respectively, then

$D_{i,j} = T_i - T_j$ from which it follows that

$D_{i,k} = D_{i,j} + D_{j,k}$

$D_{i,k} = -D_{k,i}$

Note that $D_{i,j}$ may be negative in some cases.

Thus, if $MV_{5,1}$ is a motion vector for a P_5 pixelblock as referenced to I_1, then for the corresponding pixelblocks in B_2, B_3 and B_4 the motion vectors as referenced to I_1 and P_5, respectively would be interpolated by

$$MV_{2,1} = MV_{5,1}*D_{2,1}/D_{5,1}$$
$$MV_{5,2} = MV_{5,1}*D_{5,2}/D_{5,1}$$

$$MV_{3,1} = MV_{5,1}*D_{3,1}/D_{5,1}$$
$$MV_{5,3} = MV_{5,1}*D_{5,3}/D_{5,1}$$

$$MV_{4,1} = MV_{5,1}*D_{4,1}/D_{5,1}$$
$$MV_{5,4} = MV_{5,1}*D_{5,4}/D_{5,1}$$

Note that since ratios of display times are used for motion vector prediction, absolute display times are not needed. Thus, relative display times may be used for $D_{i,j}$ inter-picture display time values.

This scenario may be generalized, as for example in the H.264 standard. In the generalization, a P or B picture may use any previously transmitted picture for its motion vector prediction. Thus, in the above case picture B_3 may use picture I_1 and picture B_2 in its prediction. Moreover, motion vectors may be extrapolated, not just interpolated. Thus, in this case we would have:

$$MV_{3,1} = MV_{2,1}*D_{3,1}/D_{2,1}$$

Such motion vector extrapolation (or interpolation) may also be used in the prediction process for predictive coding of motion vectors.

Encoding Inter-Picture Display Times

The variable inter-picture display times of video sequences should be encoded and transmitted in a manner that renders it possible to obtain a very high coding efficiency and has selectable accuracy such that it meets the requirements of a video decoder. Ideally, the encoding system should simplify the tasks for the decoder such that relatively simple computer systems can decode the digital video.

The variable inter-picture display times are potentially needed in a number of different video encoding systems in order to compute differential motion vectors, Direct Mode motion vectors, and/or Implicit B Prediction Block Weighting.

The problem of variable inter-picture display times in video sequences is intertwined with the use of temporal references. Ideally, the derivation of correct pixel values in the output pictures in a video CODEC should be independent of the time at which that picture is decoded or displayed. Hence, timing issues and time references should be resolved outside the CODEC layer.

There are both coding-related and systems-related reasons underlying the desired time independence. In a video CODEC, time references are used for two purposes:

(1) To establish an ordering for reference picture selection; and

(2) To interpolate motion vectors between pictures.

To establish an ordering for reference picture selection, one may simply send a relative position value. For example, the difference between the frame position N in decode order

and the frame position M in the display order, i.e., N-M. In such an embodiment, time-stamps or other time references would not be required. To interpolate motion vectors, temporal distances would be useful if the temporal distances could be related to the interpolation distance. However, this may not be true if the motion is non-linear. Therefore, sending parameters other than temporal information for motion vector interpolation seems more appropriate.

In terms of systems, one can expect that a typical video CODEC is part of a larger system where the video CODEC coexists with other video (and audio) CODECs. In such multi-CODEC systems, good system layering and design requires that general functions, which are logically CODEC-independent such as timing, be handled by the layer outside the CODEC. The management of timing by the system and not by each CODEC independently is critical to achieving consistent handling of common functions such as synchronization. For instance in systems that handle more than one stream simultaneously, such as a video/audio presentation, timing adjustments may sometimes be needed within the streams in order to keep the different streams synchronized. Similarly, in a system that handles a stream from a remote system with a different clock timing adjustments may be needed to keep synchronization with the remote system. Such timing adjustments may be achieved using time stamps. For example, time stamps that are linked by means of "Sender Reports" from the transmitter and supplied in RTP in the RTP layer for each stream may be used for synchronization. These sender reports may take the form of:

Video RTP TimeStamp X is aligned with reference timestamp Y

Audio RTP TimeStamp W is aligned with reference timestamp Z

Wherein the wall-clock rate of the reference timestamps is known, allowing the two streams to be aligned. However, these timestamp references arrive both periodically and separately for the two streams, and they may cause some needed re-alignment of the two streams. This is generally achieved by adjusting the video stream to match the audio or vice-versa. System handling of time stamps should not affect the values of the pixels being displayed. More generally, system handling of temporal information should be performed outside the CODEC.

A Specific Example

As set forth in the previous section, the problem in the case of non uniform inter-picture times is to transmit the inter-picture display time values $D_{i,j}$ to the digital video receiver in an efficient manner. One method of accomplishing this goal is to have the system transmit the display time difference between the current picture and the most recently transmitted stored picture for each picture after the first picture. For error resilience, the transmission could be repeated several times within the picture. For example, the display time difference may be repeated in the slice headers of the MPEG or H.264 standards. If all slice headers are lost, then presumably other pictures that rely on the lost picture for decoding information cannot be decoded either.

Thus, with reference to the example of the preceding section, a system would transmit the following inter-picture display time values:

$$D_{5,1}\ D_{2,5}\ D_{3,5}\ D_{4,5}\ D_{10,5}\ D_{6,10}\ D_{7,10}\ D_{8,10}\ D_{9,10}\ D_{12,10}\ D_{11,12}\ D_{14,12}\ D_{13,14}\ \ldots$$

For the purpose of motion vector estimation, the accuracy requirements for the inter-picture display times $D_{i,j}$ may vary from picture to picture. For example, if there is only a

single B-frame picture B_6 halfway between two P-frame pictures P_5 and P_7, then it suffices to send only:

$$D_{7,5} = 2 \text{ and } D_{6,7} = -1$$

where the $D_{i,j}$ inter-picture display time values are relative time values.

If, instead, video picture B_6 is only one quarter the distance between video picture P_5 and video picture P_7 then the appropriate $D_{i,j}$ inter-picture display time values to send would be:

$$D_{7,5} = 4 \text{ and } D_{6,7} = -1$$

Note that in both of the preceding examples, the display time between the video picture B_6 and video picture video picture P_7 (inter-picture display time $D_{6,7}$) is being used as the display time "unit" value . In the most recent example, the display time difference between video picture P_5 and picture video picture P_7 (inter-picture display time $D_{6,7}$) is four display time "units" ($4 * D_{6,7}$) .

Improving Decoding Efficiency

In general, motion vector estimation calculations are greatly simplified if divisors are powers of two. This is easily achieved in our embodiment if $D_{i,j}$ (the inter-picture time) between two stored pictures is chosen to be a power of two as graphically illustrated in **Figure 4**. Alternatively, the estimation procedure could be defined to truncate or round all divisors to a power of two.

In the case where an inter-picture time is to be a power of two, the number of data bits can be reduced if only the integer power (of two) is transmitted instead of the full value of the inter-picture time. **Figure 4** graphically illustrates a case wherein the

distances between pictures are chosen to be powers of two. In such a case, the $D_{3,1}$ display time value of 2 between video picture P_1 and picture video picture P_3 is transmitted as 1 (since $2^1 = 2$) and the $D_{7,3}$ display time value of 4 between video picture P_7 and picture video picture P_3 can be transmitted as 2 (since $2^2 = 4$).

Alternatively, the motion vector interpolation of extrapolation operation can be approximated to any desired accuracy by scaling in such a way that the denominator is a power of two. (With a power of two in the denominator division may be performed by simply shifting the bits in the value to be divided.) For example,

$$D_{5,4}/D_{5,1} \sim Z_{5,4}/P$$

Where the value P is a power of two and $Z_{5,4} = P*D_{5,4}/D_{5,1}$ is rounded or truncated to the nearest integer. The value of P may be periodically transmitted or set as a constant for the system. In one embodiment, the value of P is set as $P = 2^8 = 256$.

The advantage of this approach is that the decoder only needs to compute $Z_{5,4}$ once per picture or in many cases the decoder may pre-compute and store the Z value. This allows the decoder to avoid having to divide by $D_{5,1}$ for every motion vector in the picture such that motion vector interpolation may be done much more efficiently. For example, the normal motion vector calculation would be:

$$MV_{5,4} = MV_{5,1}*D_{5,4}/D_{5,1}$$

But if we calculate and store $Z_{5,4}$ wherein $Z_{5,4} = P*D_{5,4}/D_{5,1}$ then

$$MV_{5,4} = MV_{5,1}*Z_{5,4}/P$$

But since the P value has been chosen to be a power of two, the division by P is merely a simple shift of the bits. Thus, only a single multiplication and a single shift are required to calculate motion vectors for subsequent pixelblocks once the Z value has been

calculated for the video picture. Furthermore, the system may keep the accuracy high by performing all divisions last such that significant bits are not lost during the calculation. In this manner, the decoder may perform exactly the same as the motion vector interpolation as the encoder thus avoiding any mismatch problems that might otherwise arise.

Since division (except for division by powers of two) is a much more computationally intensive task for a digital computer system than addition or multiplication, this approach can greatly reduce the computations required to reconstruct pictures that use motion vector interpolation or extrapolation.

In some cases, motion vector interpolation may not be used. However, it is still necessary to transmit the display order of the video pictures to the receiver/player system such that the receiver/player system will display the video pictures in the proper order. In this case, simple signed integer values for $D_{i,j}$ suffice irrespective of the actual display times. In some applications only the sign (positive or negative) may be needed to reconstruct the picture ordering.

The inter-picture times $D_{i,j}$ may simply be transmitted as simple signed integer values. However, many methods may be used for encoding the $D_{i,j}$ values to achieve additional compression. For example, a sign bit followed by a variable length coded magnitude is relatively easy to implement and provides coding efficiency.

One such variable length coding system that may be used is known as UVLC (Universal Variable Length Code). The UVLC variable length coding system is given by the code words:

```
1 =         1
2 =       0 1 0
3 =       0 1 1
4 =     0 0 1 0 0
5 =     0 0 1 0 1
6 =     0 0 1 1 0
7 =     0 0 1 1 1
8 = 0 0 0 1 0 0 0...
```

Another method of encoding the inter-picture times may be to use arithmetic coding. Typically, arithmetic coding utilizes conditional probabilities to effect a very high compression of the data bits.

Thus, the present invention introduces a simple but powerful method of encoding and transmitting inter-picture display times and methods for decoding those inter-picture display times for use in motion vector estimation. The encoding of inter-picture display times can be made very efficient by using variable length coding or arithmetic coding. Furthermore, a desired accuracy can be chosen to meet the needs of the video codec, but no more.

The foregoing has described a system for specifying variable accuracy inter-picture timing in a multimedia compression and encoding system. It is

contemplated that changes and modifications may be made by one of ordinary skill in the art, to the materials and arrangements of elements of the present invention without departing from the scope of the invention.

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. For a sequence of video pictures comprising a first video picture, a second video picture, and a third video picture, a method comprising:

computing a scaling value that is based on (i) a first order difference value between an order value for the third video picture and an order value for the first video picture, and (ii) a second order difference value between an order value for the second video picture and the order value for the first video picture, wherein an order value for a video picture specifies a display order for the video picture; and

computing a motion vector for the second video picture based on the scaling value and a motion vector for the third video picture, wherein computing the motion vector for the second video picture comprises performing a bit shifting operation.

2. The method of claim 1 further comprising decoding the second video picture by using the computed motion vector.

3. The method of claim 1 further comprising:

encoding the second video picture by using the computed motion vector; and

storing the encoded second video picture in a bitstream.

4. The method of claim 1, wherein an order value for a video picture is representative of an output position for the video picture in the sequence of video pictures.

5. The method of claim 1, wherein performing a bit shifting operation comprises shifting a binary value by 8 bit positions.

6. The method of claim 1 further comprising decoding the second video picture by using the computed motion vector.

7. The method of claim 1, wherein the order value for the second video picture is encoded in a bitstream by using an exponent of a power of two integer.

8. The method of claim 1, wherein the scaling value is for computing a plurality of motion vectors for the second video picture.

9. The method of claim 1, wherein the order value for the second video picture is derived from a value that is stored in a slice header associated with the second video picture.

10. The method of claim 1, wherein the bit shifting operation performs a division by a power of two value.

11. The method of claim 1, wherein the first video picture is an I video picture that does not comprise any unidirectional or bidirectional predicted macroblock.

12. The method of claim 1, wherein the second video picture is a B video picture that comprises at least one bidirectional predicted macroblock.

13. The method of claim 1, wherein computing the motion vector comprises performing an interpolation operation.

14. The method of claim 1, wherein computing the motion vector comprises performing an extrapolation operation.

15. The method of claim 1 further comprising encoding the second video picture using the computed motion vector.

16. The method of claim 1, wherein the order value for the second video picture is stored in a bitstream in a compressed format.

17. The method of claim 16, wherein the order value for the second video picture is compressed by using variable length coding.

18. The method of claim 16, wherein the order value for the second video picture is compressed by using arithmetic coding.

19. The method of claim 1, wherein a particular order value is derived from a value that is encoded more than once in a bitstream.

20. For a sequence of video pictures comprising a first video picture, a second video picture, and a third video picture, a method comprising:

computing a scaling value that is based on (i) a power of two value, (ii) a first order difference value between an order value for the third video picture and an order value for the first video picture, and (iii) a second order difference value between an order value for the second video picture and the order value for the first video picture, wherein the order value for the second video picture is encoded in a slice header associated with the second video picture; and

computing a motion vector for the second video picture based on the scaling value and a motion vector for the third video picture, wherein computing the motion vector for the second video picture comprises performing a bit shifting operation.

21. The method of claim 20 further comprising decoding the second video picture by using the computed motion vector.

22. The method of claim 20, wherein the scaling value is proportional to the second order difference value and inversely proportional to the first order difference value.

23. The method of claim 20, wherein computing the scaling value is subject to a truncation operation.

24. The method of claim 20, wherein the bit shifting operation performs a division operation that is based on said power of two value.

25. The method of claim 20 further comprising encoding the second video picture using the computed motion vector for the second video picture.

26. For a sequence of video pictures comprising a first video picture, a second video picture, and a third video picture, a method comprising:

computing a scaling value that is based on (i) a first order difference value between an order value for the third video picture and an order value for the first video picture, and (ii) a second order difference value between an order value for the second video picture and the order value for the first video picture, wherein an order value for a video picture is encoded in a slice header of a bitstream associated with the video picture; and

computing a particular motion vector for the second video picture based on the scaling value and a motion vector for the third video picture, wherein computing the particular motion vector comprises performing a division by a power of two value.

27. The method of claim 26, wherein the order value for the second video picture is encoded in the bitstream by using an exponent of a power of two integer.

28. The method of claim 26, wherein an order value for a video picture specifies a position for the video picture in the sequence of video pictures.

29. The method of claim 28, wherein the position for the video picture is an output position.

30. The method of claim 26 further comprising decoding the second video picture by using the computed motion vector for the second video picture.

31. The method of claim 26 further comprising encoding the second video picture by using the computed motion vector for the second video picture.

32. A method comprising:

computing a scaling value based on (i) a first order difference value between an order value for a first video picture and an order value for a second video picture, and (ii) a particular value that is based on a power of two value and a second order difference value between an order value for a third video picture and the order value for the second video picture, wherein the order value for the second video picture is encoded in a slice header of a bitstream associated with the second video picture; and

using said scaling value to compute a motion vector for the second video picture by performing a bit shifting operation.

33. The method of claim 32, wherein computing the scaling value comprises is subject to a truncation operation.

34. The method of claim 32, wherein using the scaling value to compute the motion vector for the second video picture comprises computing a scaled motion vector by multiplying said scaling value by a motion vector associated with said third video picture.

35. The method of claim 32, wherein the bit shifting operation performs a division that is based on said power of two value.

36. The method of claim 32, wherein the order value for the second video picture is encoded in the bitstream by using an exponent of a power of two integer.

37. The method of claim 32, wherein the bit shifting operation shifts a binary value by 8 bit positions.

38. For a sequence of video pictures comprising a first video picture, a second video picture, and a third video picture, the method comprising:

computing a scaling value that is (i) inversely proportional to a first order difference value between an order value for the third video picture and an order value for the first video picture and (ii) directly proportional to a second order difference value between an order value for the second video picture and the order value for the first video picture, wherein the order value for the second picture is encoded in a bitstream more than once; and

computing a motion vector for the second video picture by multiplying the scaling value by a motion vector for the third video picture and performing a division operation that is based on a power of two value.

39. The method of claim 38 further comprising decoding the second video picture by using the computed motion vector.

40. The method of claim 38 further comprising:

encoding the second video picture by using the computed motion vector; and

storing the encoded second video picture in the bitstream.

41. The method of claim 38, wherein the order value for the second video picture is encoded in the bitstream by using an exponent of a power of two integer.

42. The method of claim 38, wherein an order value for a video picture specifies a display order for the video picture.

43. The method of 38, wherein the order value is compressed in the bitstream using variable length coding.

44. The method of claim 38, wherein the division operation divides by two hundred and fifty-six (256).

45. For a sequence of video pictures comprising a first video picture, a second video picture, and a third video picture, a method comprising:

computing a scaling value that is (i) inversely proportional to a first order difference value between an order value for the third video picture and an order value for the first video picture, (ii) directly proportional to a second order difference value between an order value for the second video picture and the order value for the first video picture, and (iii) directly proportional to a power of two value, wherein an order value for a video picture specifies a display order for the video picture; and

computing a motion vector for the second video picture by multiplying the scaling value and a motion vector for the third video picture and performing a division operation based on said power of two value.

46. The method of claim 45 further comprising decoding the second video picture by using the computed motion vector.

47. The method of claim 45 further comprising:

encoding the second video picture by using the computed motion vector; and

storing the encoded second video picture in a bitstream.

48. The method of claim 45, wherein computing the scaling value is subject to a truncation operation.

49. The method of claim 45, wherein the order value for the second video picture is encoded in a slice header associated with the second video picture.

50. The method of claim 45, wherein the order value for the second video picture is encoded in a bitstream by using an exponent of a power of two integer.

51. The method of claim 45, wherein the division operation divides by two hundred and fifty-six (256).

52. For a sequence of video pictures comprising a first video picture, a second video picture, and a third video picture, a method comprising:

computing a scaling value that is based on (i) a first order difference value between an order value for the third video picture and an order value for the first video picture, and (ii) a second order difference value between an order value for the second video picture and the order value for the first video picture, wherein computing the scaling value is subject to a truncation operation, wherein the order value for the second video picture specifies a display order for the second video picture;

computing a particular motion vector for the second video picture based on the scaling value and a motion vector for the third video picture, wherein computing the particular motion vector for the second video picture comprises performing a bit shifting operation.

53. The method of claim 52 further comprising decoding the second video picture by using the computed motion vector.

54. The method of claim 52 further comprising:

encoding the second video picture by using the computed motion vector; and

storing the encoded second video picture in a bitstream.

55. The method of claim 52, wherein computing the scaling value comprise performing an interpolation operation.

56. The method of claim 52, wherein an order value for a video picture specifies a display order for the video picture.

57. The method of claim 52, wherein the bit shifting operation shifts a binary value by 8 bit positions.

58. The method of claim 52, wherein computing the motion vector for the second video picture comprises multiplying the scaling value with the motion vector for the third video picture.

59. For a sequence of video pictures comprising a first video picture, a second video picture, and a third video picture, a method comprising:

subject to a truncation operation, computing a scaling value that is (i) inversely proportional to a first order difference value between an order value for the third video picture and an order value for the first video picture and (ii) directly proportional to a second order difference value between an order value for the second video picture and the order value for the first video picture, wherein the order value for the second video picture is encoded in a slice header of a bitstream associated with the second video picture; and

computing a motion vector for the second video picture by multiplying the scaling value by a motion vector for the third video picture and performing a bit shifting operation.

60. The method of claim 59 further comprising decoding the second video picture by using the computed motion vector.

61. The method of claim 59 further comprising:

encoding the second video picture by using the computed motion vector; and

storing the encoded second video picture in the bitstream.

62. The method of claim 59, wherein computing the scaling value comprise performing an interpolation operation.

63. The method of claim 59, wherein the order value for the second video picture is encoded in the bitstream by using an exponent of a power of two integer.

64. The method of claim 59, wherein the bit shifting operation shifts a binary value by 8 bit positions.

65. For a sequence of video pictures comprising a first video picture, a second video picture, and a third video picture, the method comprising:

computing a scaling value that is (i) inversely proportional to a first order difference value between an order value for the third video picture and an order value for the first video picture and (ii) directly proportional to a second order difference value between an order value for the second video picture and the order value for the first video picture, wherein the order value for the second video picture is encoded in a bitstream by using an exponent of a power of two integer; and

computing a motion vector for the second video picture by multiplying the scaling value by a motion vector for the third video picture and performing a bit shifting operation.

66. The method of claim 65 further comprising decoding the second video picture by using the computed motion vector.

67. Te method of claim 65 further comprising:

encoding the second video picture by using the computed motion vector; and

storing the encoded second video picture in the bitstream.

68. The method of claim 65, wherein computing the scaling value comprise performing an interpolation operation.

69. The method of claim 65, wherein the order value for the second video picture is encoded in a slice header associated with the second video picture.

70. The method of 69, wherein the order value is compressed in the bitstream using variable length coding.

71. The method of claim 65, wherein the bit shifting operation shifts a binary value by 8 bit positions.

72. A method of decoding a bitstream comprising encoded first, second, and third video pictures, the method comprising:

receiving an integer value representing an exponent of a particular power of two integer;

computing a scaling value that is based on (i) a first order difference value between an order value for the third video picture and an order value for the first video picture, and (ii) a second order difference value between an order value for the second video picture and the order value for the first video picture, wherein the order value for the second video picture is derived from the particular power of two integer;

computing a motion vector for the second video picture by multiplying the scaling value by a motion vector for the third video picture and performing a bit shifting operation; and

decoding the second video picture by using the computed motion vector.

73. The method of claim 72 wherein computing the scaling value is subject to a truncation operation.

74. The method of claim 72 wherein the scaling value is inversely proportional to the first order difference value and directly proportional to the second order difference value.

75. The method of claim 72, wherein the scaling value is for computing a plurality of motion vectors for the second video pictures.

76. The method of claim 72, wherein the bit shifting operation shifts a binary value by 8 bit positions.

77. The method of claim 72, wherein computing the motion vector for the second video picture comprises multiplying the scaling value with the motion vector for the third video picture.

78. The method of claim 72, wherein an order value for a particular video picture specifies a display order for the particular video picture in a sequence of video pictures.

79. The method of claim 72, wherein the order value for the second video picture is encoded in a slice header associated with the second video picture.

80. The method of claim 72, wherein the order value for the second video picture is encoded more than once in the bitstream.

81. A method for encoding a sequence of video pictures comprising a first video picture, a second video picture, and a third video picture, the method comprising:

computing a scaling value that is based on (i) a first order difference value between an order value for the third video picture and an order value for the first video picture, and (ii) a second order difference value between an order value for the second video picture and the order value for the first video picture;

computing a particular motion vector for the second video picture based on the scaling value and a motion vector for the third video picture, wherein computing the particular motion vector for the second video picture comprises performing a bit shifting operation;

encoding the second video picture in a bitstream by using the computed motion vector; and

encoding the order value for the second video picture in the bitstream by using an exponent of a power of integer.

82. The method of claim 81, wherein computing the scaling value is subject to a truncation operation.

83. The method of claim 81, wherein the scaling value is inversely proportional to the first order difference value and directly proportional to the second order difference value.

84. The method of claim 81, wherein the scaling value is for computing a plurality of motion vectors for the second video picture.

85. The method of claim 81, wherein the bit shifting operation shifts a binary value by 8 bit positions.

86. The method of claim 81, wherein computing the motion vector for the second video picture comprises multiplying the scaling value with the motion vector for the third video picture.

87. The method of claim 81, wherein an order value for a particular video picture specifies a display order for the particular video picture.

88. The method of claim 81, wherein the order value for the second video picture is encoded in a slice header associated with the second video picture.

89. The method of claim 81, wherein the order value for the second video picture is encoded more than once in the bitstream comprising the encoded first, second, and third video pictures.

90. A computer readable medium storing a computer program that is executable by at least one processor, the computer program comprising sets of instructions for implementing the method according to any one of claims 1 to 37 and 38 to 89.

91. A computer system comprising means for implementing steps according to any one of claims 1 to 37 and 38 to 89.

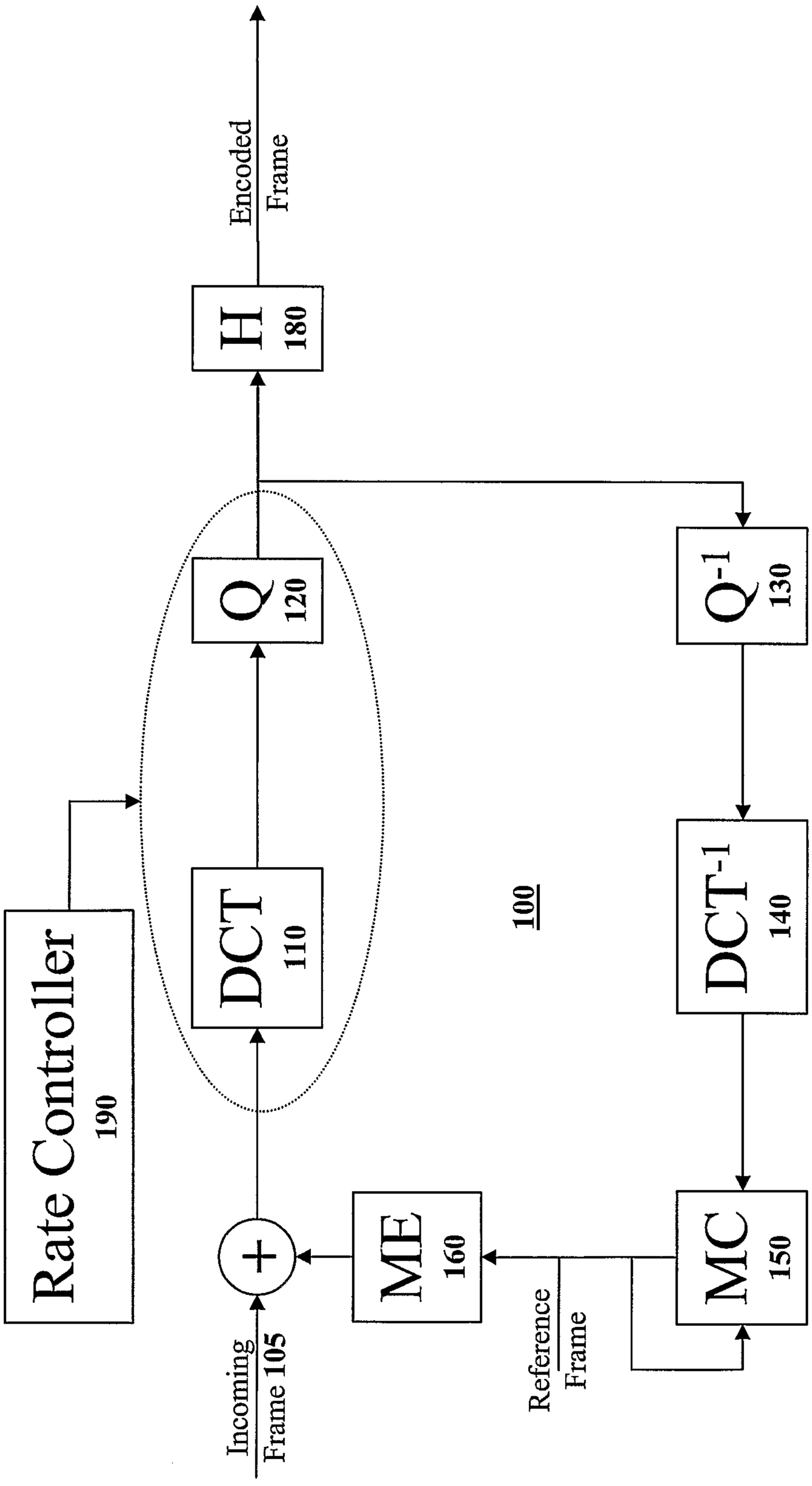

Encoded Frame
H 180
Q 120
Q-1 130
DCT 110
100
DCT-1 140
Rate Controller 190
ME 160
MC 150
Incoming Frame 105
Reference Frame

Figure 1

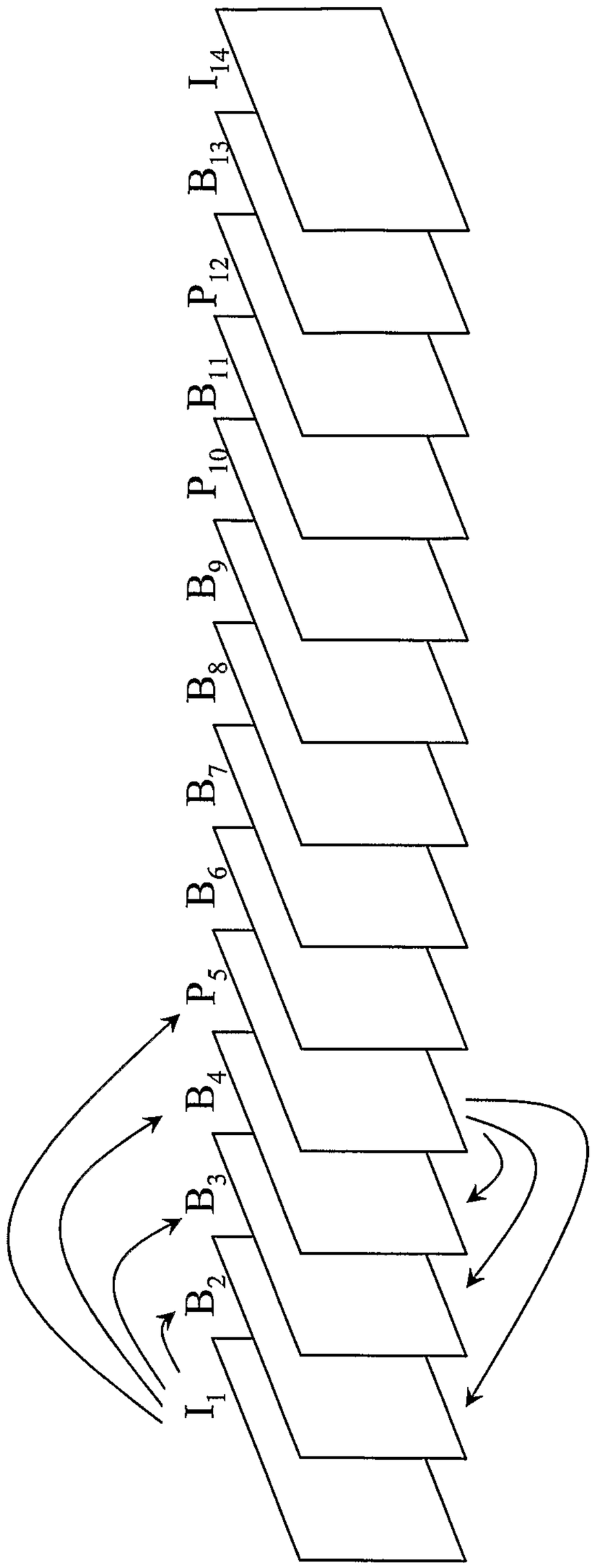


Figure 2

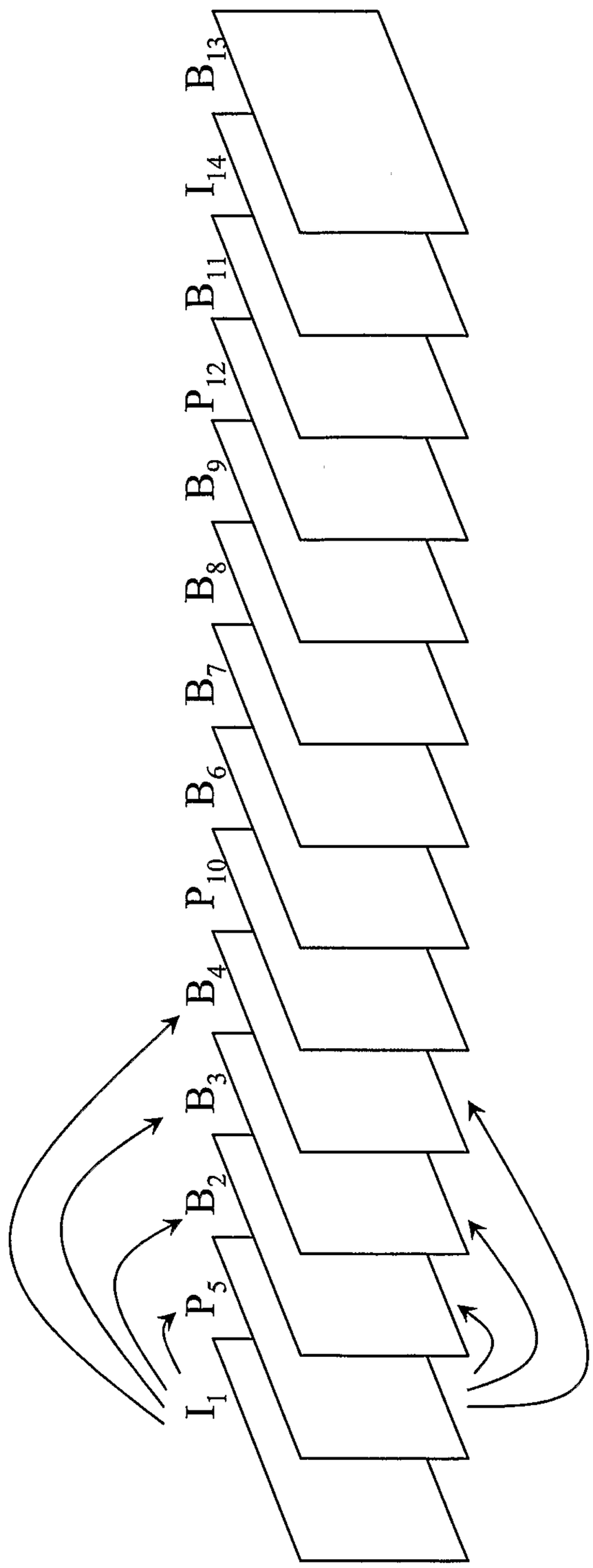


Figure 3

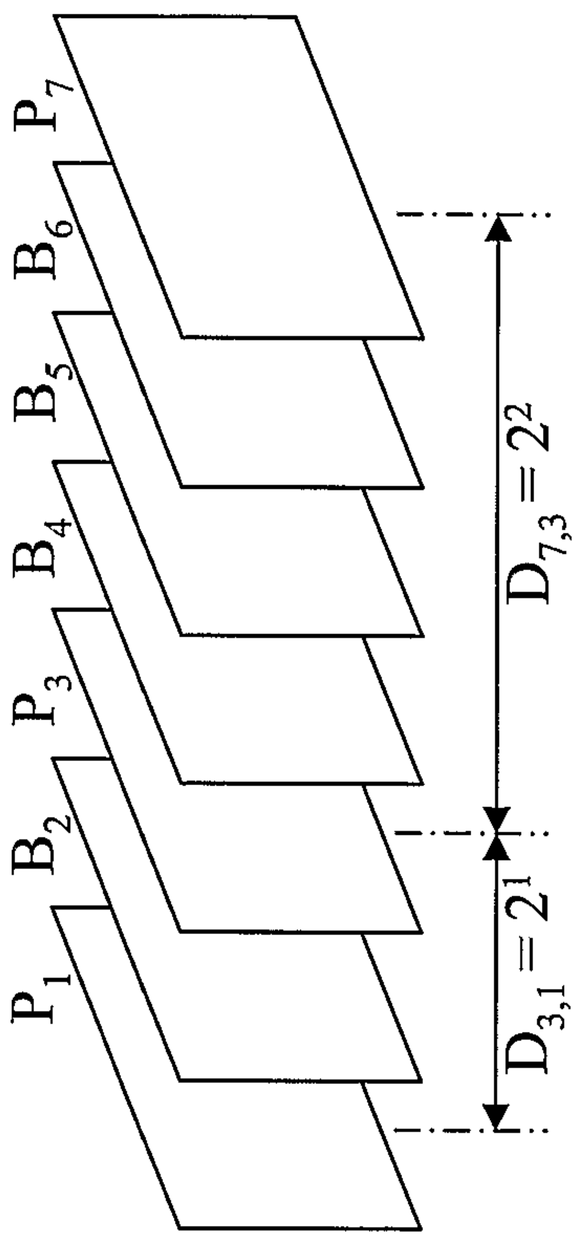
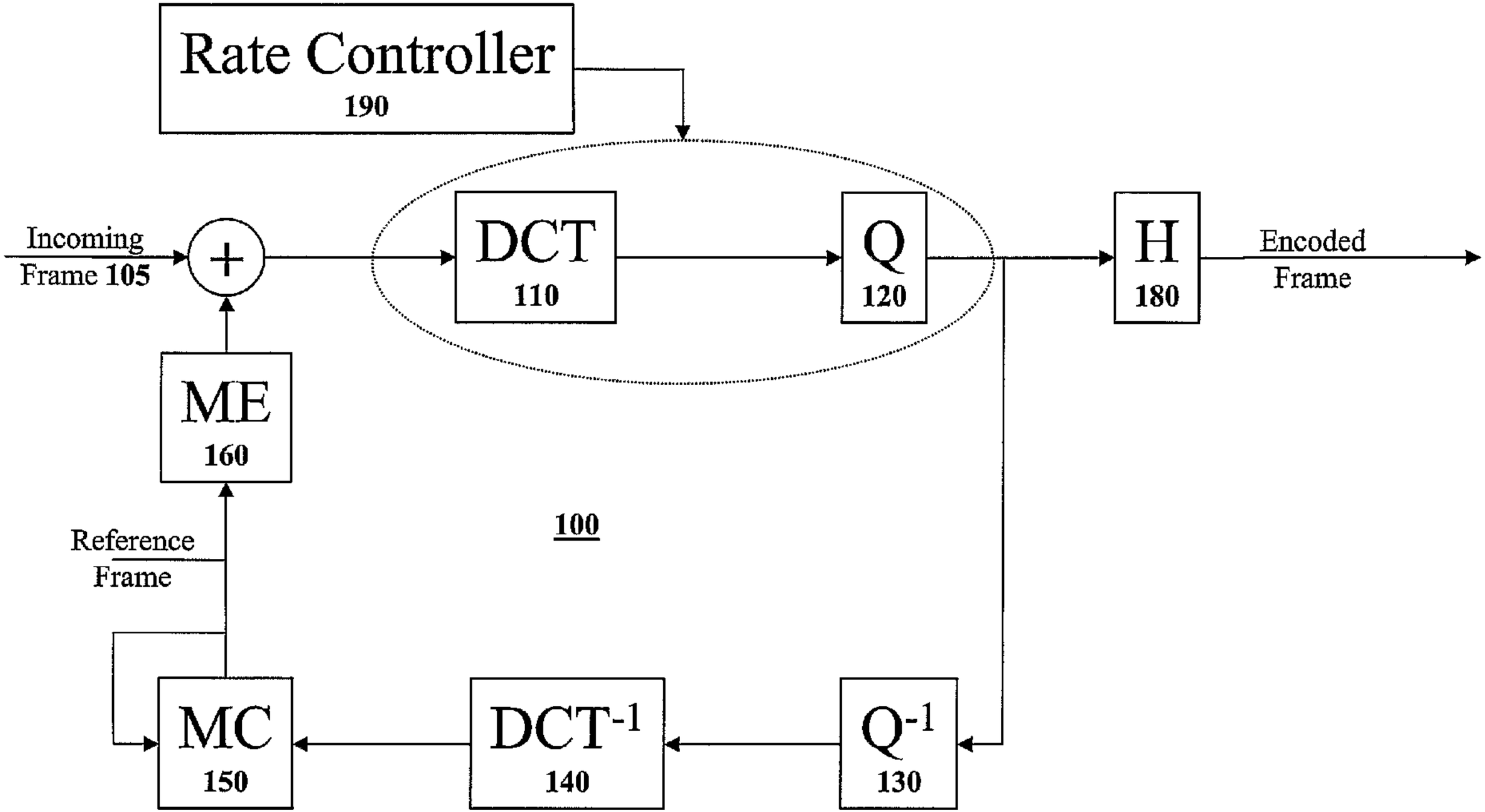


Figure 4


Rate Controller
190
Incoming
Frame 105
+
DCT
110
Q
120
H
180
Encoded
Frame
ME
160
100
Reference
Frame
MC
150
DCT-1
140
Q-1
130