COOLEY GODWARD KRONISH LLP
JOHN S. KYLE (199196) (jkyle@cooley.com)
4401 Eastgate Mall
San Diego, CA 92121
Telephone: (858) 550-6000
Facsimile: (858) 550-6420

STEPHEN C. NEAL (170085) (nealsc@cooley.com)
Five Palo Alto Square
3000 El Camino Real
Palo Alto, CA 94306-2155
Telephone: (650) 843-5000
Facsimile: (650) 857-0663

Attorneys for Plaintiff
MULTIMEDIA PATENT TRUST

IN THE UNITED STATES DISTRICT COURT
FOR THE SOUTHERN DISTRICT OF CALIFORNIA

Multimedia Patent Trust,

Plaintiff,

v.

Tandberg, Inc.

Defendant.

'09 CV 1377 JM WMc
Civil Action No. _____

DEMAND FOR JURY TRIAL

PLAINTIFF MULTIMEDIA PATENT TRUST'S COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff Multimedia Patent Trust ("Plaintiff"), by counsel, alleges as follows:

THE PARTIES

1. Plaintiff Multimedia Patent Trust is a Delaware statutory trust under the laws of the Delaware Statutory Trust Act, 12 Del. C. §§ 3801, *et seq.*

2. On information and belief, Defendant Tandberg is a corporation organized under the laws of the State of Delaware, and having a principal place of business at 1860 Michael Faraday Drive, Suite 100, Reston, Virginia 20190.

JURISDICTION AND VENUE

3. This is a civil action for patent infringement arising under the United States patent statutes, 35 U.S.C. § 1 *et seq.*

4. This Court has jurisdiction over the subject matter of this action under 28 U.S.C. §§ 1331 and 1338(a).

5. Defendant Tandberg, Inc. ("Tandberg") is subject to this Court's personal jurisdiction because it does and has done substantial business in this judicial District, including: (i) maintaining principal places of business in California (ii) selling and offering for sale to consumers within this District infringing video teleconferencing services and equipment; and (iii) regularly doing or soliciting business, engaging in other persistent courses of conduct, and/or deriving substantial revenue from products and/or services provided to individuals in this State and in this District. In addition, Defendant Tandberg has designated an agent for service of process in the State of California.

6. Venue is proper in this judicial District under 28 U.S.C. §§ 1391(b)-(c) and 1400(b).

BACKGROUND FACTS & PATENTS-IN-SUIT

7. The patents-in-suit are generally directed to systems and methods of encoding and decoding signals representative of moving images (i.e., "video compression").

8. Video compression techniques are used in many industries that involve either the transmission of video from one location to another and/or the manufacture and sale of devices to receive or store video signals. These industries include, for example, content providers, cable and satellite companies, teleconferencing providers, television manufacturers, television broadcasters and digital media providers.

9. Video compression reduces the amount of digital data needed to represent video so that it can be sent more efficiently over communications media, such as the Internet and satellites, or stored more efficiently on storage media such as DVDs and Blu-Ray discs. Video consists of a series of pictures, or frames, with each frame capturing a scene at an instant of time. When viewed one after another, the frames form the video sequences. Video compression involves

reducing the amount of digital data needed to represent information about the content of these pictures or frames while allowing a video to ultimately be reproduced from that information.

10. There are numerous benefits to video compression. For instance, it enables large amounts of video data to be stored on smaller memory devices and permits broadcasters to transmit greater numbers of programs using the same bandwidth over a particular transmission medium. For example, without video compression it would be impossible to store a feature-length film on a single DVD. Also, video retrieval via the internet would not be feasible due to the huge volume of uncompressed data that would need to be transmitted. The challenge that comes with video compression, however, is assuring that the video image ultimately reproduced from the reduced amount of digital data is of sufficient quality.

11. A video signal is encoded (compressed) prior to being transmitted over a medium or before it is stored on a medium. When the video signal is read off the storage medium or is received at the other end, it is decoded (decompressed) to recreate either the original signal or, in the case of a lossy compression technique (by which certain unnecessary bits of data are eliminated), a close approximation of the original signal. When encoding a video, the video signal is processed using a variety of techniques that reduce the amount of data, such as transformation, quantization, motion-compensated prediction and variable length encoding.

12. On September 18, 1990, the United States Patent and Trademark Office duly and legally issued United States Patent No. 4,958,226 ("the '226 Patent"), entitled "Conditional Motion Compensated Interpolation of Digital Motion Video," to Barin G. Haskell and Atul Puri. A copy of the '226 Patent is attached as Exhibit A.

13. On July 13, 1993, the United States Patent and Trademark Office duly and legally issued United States Patent No. 5,227,878 ("the '878 Patent"), entitled "Adaptive Coding and Decoding of Frames and Fields of Video," to Atul Puri and Rangarajan Aravind. The United States Patent and Trademark Office duly and legally issued a Certificate of Correction to the '878 Patent on October 25, 2005. A copy of the '878 Patent and its Certificate of Correction are attached as Exhibit B.

14. On March 19, 1996, the United States Patent and Trademark Office duly and legally issued United States Patent No. 5,500,678 ("the '678 Patent"), entitled "Optimized Scanning of Transform Coefficients in Video Coding," to Atul Puri. The United States Patent and Trademark Office duly and legally issued a Certificate of Correction to the '678 Patent on May 29, 2007. A copy of the '678 Patent and its Certificate of Correction are attached as Exhibit C.

15. On August 4, 1992, the United States Patent and Trademark Office duly and legally issued United States Patent No. 5,136,377 ("the '377 Patent"), entitled "Adaptive Non-Linear Quantizer," to James D. Johnston, et al. A copy of the '377 Patent is attached as Exhibit D.

16. On November 28, 2006, all rights, title and interest in and to the '226, '377, '878 and '678 Patents (collectively, the "Patents-in-Suit"), including the right to sue for past infringement, were assigned to the Multimedia Patent Trust. On November 30 and December 21, 2006, the United States Patent and Trademark Office issued Notices of the recordation of the assignments.

17. Plaintiff Multimedia Patent Trust is the sole holder of the entire right, title and interest in the Patents-in-Suit.

COUNT I

(PATENT INFRINGEMENT BY DEFENDANT TANDBERG)

18. Plaintiff Multimedia Patent Trust realleges and incorporates by reference paragraphs 1-18 of this Complaint as if fully set forth herein.

19. Plaintiff Multimedia Patent Trust is the sole holder of the entire right, title and interest in the '226, '377, '878 and '678 Patents.

20. Defendant Tandberg has directly and/or contributorily infringed and/or induced others to infringe one or more claims of each of the Patents-in-Suit by making, using, offering to sell, selling and/or importing within the United States infringing products and/or services, including products and services that are compliant with the H.263, H.264, and MPEG-4 Part 2 video compression standards.

21. Defendant Tandberg continues to infringe directly, contributorily infringe and/or induce others to infringe the Patents-in-Suit.

22. Although Plaintiff Multimedia Patent Trust notified Tandberg of its infringement of the Patents-in-Suit and its need to enter into a license agreement with Plaintiff Multimedia Patent Trust, Tandberg failed to take a license and has continued its infringement.

23. Defendant Tandberg's infringement of the Patents-in-Suit has been willful.

24. Plaintiff Multimedia Patent Trust has been, and continues to be, damaged and irreparably harmed by Defendant Tandberg's infringement, which will continue unless Tandberg is enjoined by this Court.

REQUEST FOR RELIEF

WHEREFORE, Plaintiff Multimedia Patent Trust respectfully requests the following relief:

A. A judgment holding Tandberg liable for infringement of the Patents-in-Suit;

B. A permanent injunction against Tandberg, its officers, agents, servants, employees, attorneys, parent and subsidiary corporations, assigns and successors in interest, and those persons in active concert or participation with them, enjoining it from continued acts of infringement of the Patents-in-Suit;

C. An accounting for damages resulting from Tandberg's infringement of the Patents-in-Suit, together with pre-judgment and post-judgment interest;

D. A judgment holding that Tandberg's infringement is willful, and a trebling of damages pursuant to 35 U.S.C. § 284;

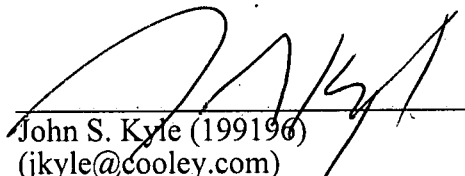
E. A judgment holding this Action to be an exceptional case, and an award to Plaintiff Multimedia Patent Trust for its attorneys' fees and costs pursuant to 35 U.S.C. § 285;

F. Such other and further relief as this Court deems just and proper.

Dated: June 25, 2009

Respectfully submitted,

COOLEY GODWARD KRONISH LLP
STEPHEN C. NEAL (170085)
FRANK V. PIETRANTONIO (25473) (PRO HAC VICE PENDING)
JONATHAN G. GRAVES (46136) (PRO HAC VICE PENDING)
NATHAN K. CUMMINGS (41372) (PRO HAC VICE PENDING)
JOHN S. KYLE (199196)



John S. Kyle (199196)
(jkyle@cooley.com)
Attorneys for Plaintiff
MULTIMEDIA PATENT TRUST

DEMAND FOR JURY TRIAL

Pursuant to Federal Rule of Civil Procedure 38, Plaintiff Multimedia Patent Trust hereby demands trial by jury.

Dated: June 25, 2009

Respectfully submitted,

COOLEY GODWARD KRONISH LLP
STEPHEN C. NEAL (170085)
FRANK V. PIETRANTONIO (25473) (PRO HAC VICE PENDING)
JONATHAN G. GRAVES (46136) (PRO HAC VICE PENDING)
NATHAN K. CUMMINGS (41372) (PRO HAC VICE PENDING)
JOHN S. KYLE (199196)

John S. Kyle (199196)
(jkyle@cooley.com)
Attorneys for Plaintiff
MULTIMEDIA PATENT TRUST

United States Patent [19]

Haskell et al.

[11] **Patent Number: 4,958,226**

[45] **Date of Patent: Sep. 18, 1990**

[54] **CONDITIONAL MOTION COMPENSATED INTERPOLATION OF DIGITAL MOTION VIDEO**

[75] Inventors: **Barin G. Haskell**, Tinton Falls, N.J.; **Atul Puri**, Bronx, N.Y.

[73] Assignee: **AT&T Bell Laboratories**, Murray Hill, N.J.

[21] Appl. No.: **413,520**

[22] Filed: **Sep. 27, 1989**

[51] **Int. Cl.**[5] **H04N 7/12**

[52] **U.S. Cl.** **358/136**; 358/135

[58] **Field of Search** 358/135, 136, 133

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,218,703	8/1980	Netravali et al. .	
4,218,704	8/1980	Netravali et al. .	
4,383,272	5/1983	Netravali et al. .	
4,442,454	4/1984	Powell	358/167
4,460,923	7/1984	Hirano et al.	358/136
4,665,436	5/1987	Osborne et al.	358/135 X
4,667,233	5/1987	Furukawa	358/136
4,727,422	2/1988	Hinman	358/135 X
4,782,387	11/1988	Sabri et al.	358/135

OTHER PUBLICATIONS

"Movement-Compensated Frame-Frequency Conversion of Television Signals," H. Yamaguchi, T. Sugi and K. Kinuhata, IEEE Transactions on Communications, vol. COM-35, No. 10, Oct. 1987, pp. 1069-1082.

Primary Examiner—James J. Groody
Assistant Examiner—Victor R. Kostak
Attorney, Agent, or Firm—Henry T. Brendzel

[57] **ABSTRACT**

Motion digital video is encoded and decoded by a motion compensated interpolation method and apparatus. In accordance with the method, selected frames of the video are interpolated in the decoder with the aid of interpolation correction codes that are generated in the encoder and sent to the decoder. In an encoder embodiment that interpolates half of the frames, every other frame is encoded and decoded within the encoder. The decoded versions of adjacent frames are appropriately combined and compared to the interleaved camera frame that is to be interpolated in the decoder. The differences, which correspond to "pels correction" information, are encoded and quantized. Those that exceed a predetermined threshold value are added to the encoder's output buffer. The inverse operation is carried out in the decoder. That is every pair of decoded frames is averaged and combined with the decoded "pels correction" information to form the interpolated frames.

12 Claims, 2 Drawing Sheets

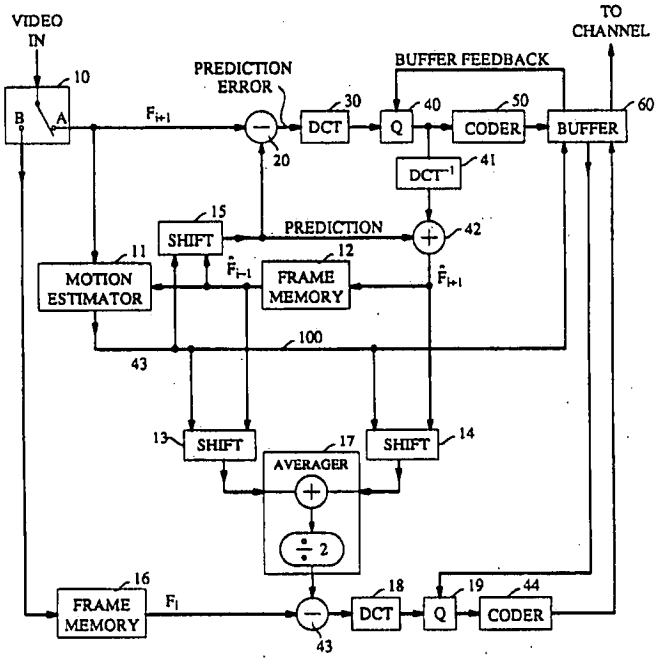


FIG. 1

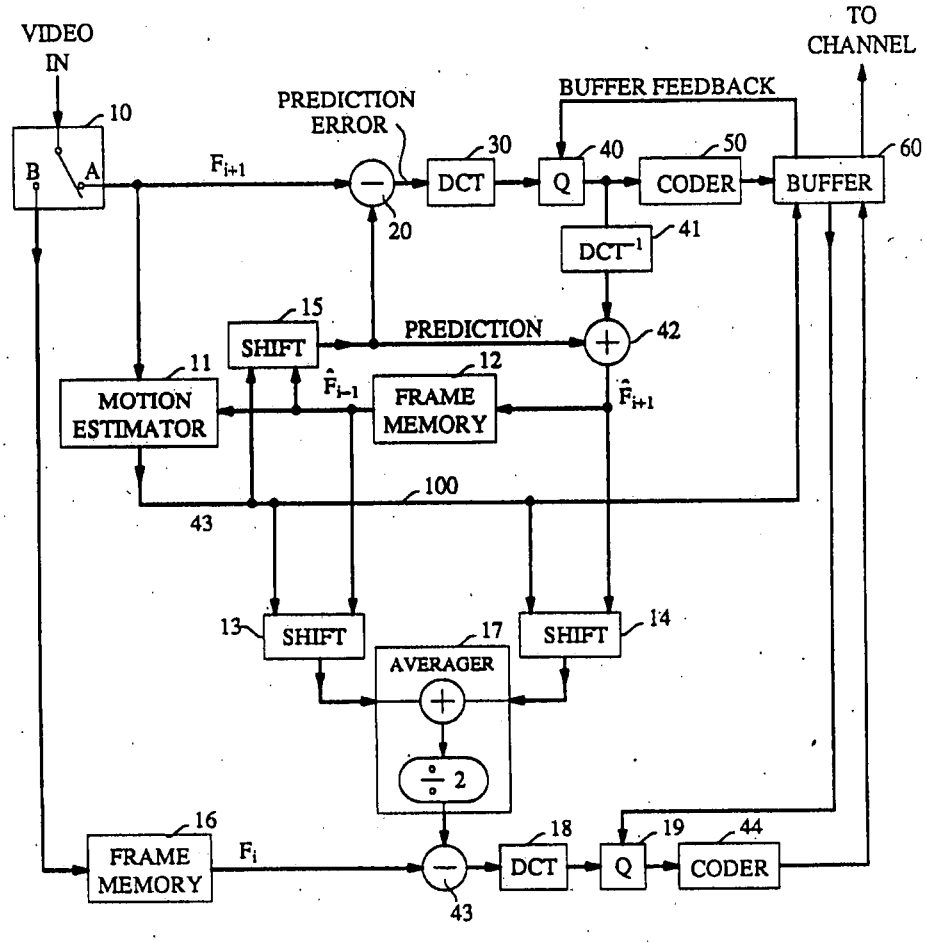
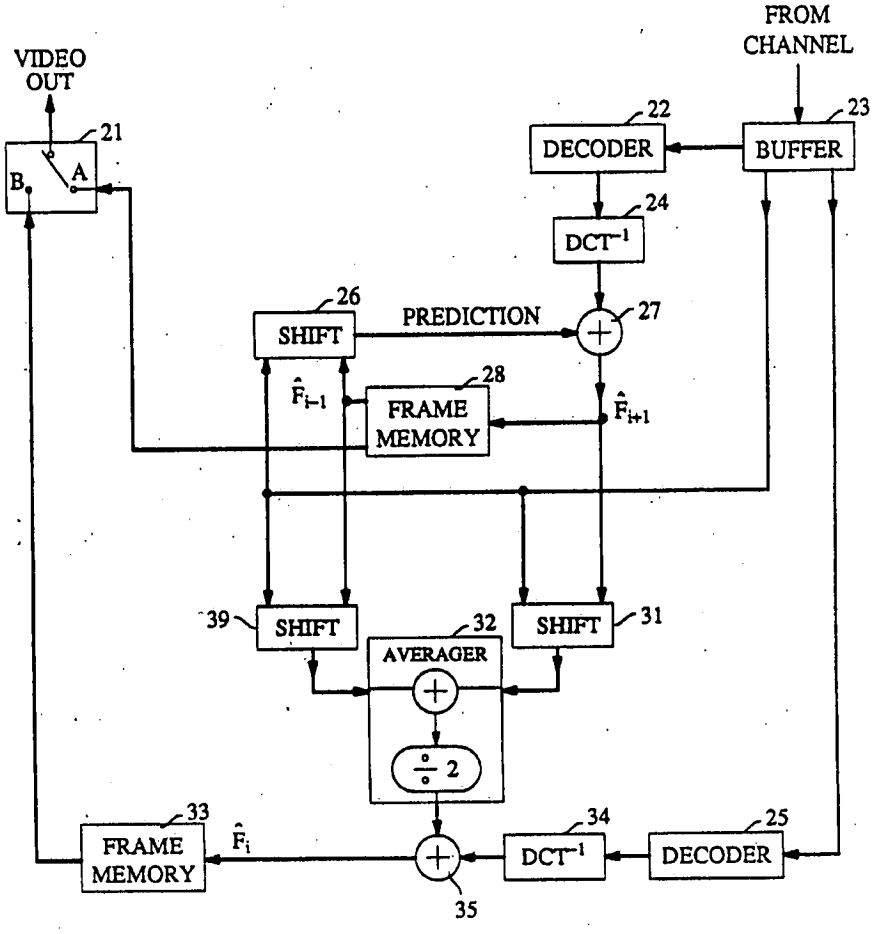

VIDEO IN
10
B
A
F_{i+1}
PREDICTION ERROR
20
30
DCT
40
Q
50
CODER
BUFFER FEEDBACK
BUFFER
60
TO CHANNEL
DCT^{-1}
41
15
SHIFT
PREDICTION
42
11
MOTION ESTIMATOR
$\hat{F}_{i-1}$
12
FRAME MEMORY
$\hat{F}_{i+1}$
100
43
13
SHIFT
17
AVERAGER
÷ 2
SHIFT
14
16
FRAME MEMORY
F_i
43
18
DCT
19
Q
44
CODER

FIG. 2


VIDEO OUT
21
B
A
FROM CHANNEL
22
DECODER
23
BUFFER
24
DCT⁻¹
26
SHIFT
PREDICTION
27
28
FRAME MEMORY
39
SHIFT
32
AVERAGER
÷ 2
SHIFT
31
33
FRAME MEMORY
34
DCT⁻¹
25
DECODER
35

CONDITIONAL MOTION COMPENSATED INTERPOLATION OF DIGITAL MOTION VIDEO

BACKGROUND OF THE INVENTION

This invention relates to signal coding and, more particularly, to a method and apparatus for encoding and decoding video signals of moving images.

Video signals typically originate from video cameras. The bandwidth of video signals is quite substantial and consequently, practioners in the art have tried to reduce the bandwidth of these signals without unduly degrading the the images. Typically to reduce bandwidth the video signals are encoded and redundancies in the encoded signals are extracted and deleted. Different techniques are used in the art and some are better suited for still images, while others are better suited for moving images. One of the techniques for reducing the bandwidth of moving images is generally referred to as motion compensated predictive coding.

In conventional motion compensated predictive coding, each video frame is first partitioned into square blocks of picture elements (pels); such as blocks fo 8 pels by 8 pels. Each block is coded, in turn, and the developed encoded sequence is transmitted over a communications channel to a decoder. The communications channel may be, or may include, a storage element. Next, a determination is made as to whether or not the pels of the block have changed significantly compared with the previous frame. If not, an indicator signal is sent which signifies to the decoder that it needs to merely repeat the pels of that block from the previous frame obtain the pels for the current block. This is known as "Conditional Replenishment". If the pels have changed since the previous frame, an attempt is made to determine the best estimate of motion that is occurring in the block. This is frequently done by a "Block Matching Motion Estimation" technique wherein the pels of the current block are successively compared with various small shifts of the corresponding block in the previous frame. The shift that gives the best match is deemed to be the "best estimate" of the displacement in the block's image between frames, and the amount of this shift, called the "Motion Vector", is selected and sent to the decoder.

The pels of the current block are then compared with those of the "best" shifted block from the previous frame to see if there is a significant difference. If not, an indicator signal is sent to the decoder, which merely causes the pels of the shifted block from the previous frame to be repeated for the pels for the current shifted block. Such blocks are said to have been successfully "Motion Compensated". However, if there is a significant difference between the two blocks, the difference is encoded and sent to the decoder so that the pels of the current block may be more accurately recovered. Coding of this difference is typically performed by means of the "Discrete Cosine Transform" (DCT).

The volume of code that is generated by the above procedure is variable. It can be appreciated, for example, that image changes that do not correspond to a uniform translation, or motion, of the image may require substantial encoding to describe the deviation of a block from its best translated replica. On the other hand, when the image does not change between successive frames, then there is a minimal amount of information that needs to be encoded. To accommodate these potentially wide variations in the amount of code that needs to be transmitted, typical encoders include a FIFO memory at the output, to serve as a buffer.

The FIFO is not a panacea. For a given transmission rate, when an excessive volume data is generated, there is always a danger that the FIFO would overflow. When it does, coding must stop until the transmission channel can empty the FIFO sufficiently so that new data to be accepted into it. Since it is inconvenient to stop encoding in the middle of a frame, most systems discard an entire frame whenever the FIFO buffer is full, or nearly so. To compensate for the loss of a frame, such systems cause the decoder to repeat its most recently available frame. Frame repeating results in moving objects in the scene being reproduced in a jerky fashion, rather than in the smooth way that would occur if frame repeating were not invoked.

There have been some suggestions for improving the quality of the repeated frames in order to make them more faithfully resemble the original. One technique is called "Motion Compensated Interpolation". With this technique, instead of simply repeating the pels from the previous frame, the Motion Vectors are used to laterally displace the block by the appropriate amount prior to display. In other words, this method creates the missing block of pels by averaging over the immediately previous and following blocks of pels that are available to the decoder. While this might seem to be a good idea, experimental results show that when the images of successive blocks do not represent translational motion, the reproduced image may be worse than with frame repeating. Although it has been observed that this degradation is caused by a relatively few pels that do not conform to the assumption of translational motion, putting these pels in the wrong place creates highly visible artifacts.

SUMMARY OF THE INVENTION

In accordance with the principles of this invention, pels that cause highly visible artifacts are detected, and corresponding correction information is transmitted to the decoder. The amount of correction information that must be sent is relatively small, and the improvement in picture quality is quite large.

Since the interpolation technique that employs the principles of this invention yields good results, it has been found acceptable to interpolate every other frame, or two out of three frames, on a regular basis. The benefit of such regular interpolation is a reduced transmission bit rate which results from the fact that the pel correction information comprises fewer bits than the actual frame coding information.

In an encoder embodiment that interpolates half of the frames, every other frame is encoded and thereafter decoded within the encoder. The decoded versions of adjacent frames are appropriately combined and compared to the interleaved camera frame that is to be interpolated in the decoder. The differences, which correspond to "pels correction" information, are encoded and quantized. Those that exceed a predetermined threshold value are added to the encoder's output buffer. The inverse operation is carried out in the decoder. That is every pair of decoded frames is averaged and combined with the decoded "pels correction" information to form the interpolated frames.

BRIEF DESCRIPTION OF THE DRAWING

FIG. 1 presents a block diagram of an encoder in accordance with the principles of this invention; and

FIG. 2 depicts a block diagram of a decoder in accordance with the principles of this invention.

DETAILED DESCRIPTION

Given a specified transmission rate in the communications channel, frame interpolation needs to be resorted to only when the FIFO is at, or near overflow. When that is the selected approach, the encoder of our invention encodes the blocks of every frame and concurrently develops the codes for interpolating the blocks from the information available to the encoder from previous and subsequent frames. At the input to the FIFO buffer, a switch is installed that is sensitive to the available memory in the buffer. When the available memory falls below a preselected threshold, the switch is set to accept the frame interpolation code. Otherwise, the switch is set to accept the frame encoding code. Other control techniques are also available, such as selecting some frames for encoding and some frames for interpolation, based on the occupancy level of the buffer. Both specific frames can thus be selected for interpolations as well as a proportion of frames to be interpolated.

The above insures that the encoder would not exceed the tranamission capacity of the communications channel. In some applications, however, it is more important to achieve a low transmission rate. Knowing that the frame interpolation code is less voluminous than the frame encoding code, it makes sense to accept the frame interpolation code wherever possible. The zeal to chose interpolation code in preference to the frame encoding code is tempered, however, by the level of degradation that is acceptable to the user in the reconstituted picture. It is further tempered by the observation that the volume of the frame interpolation code increase with increased use of the frame interpolation code so one could quickly reach a point of "diminishing returns" in the use of interpolation code.

Experimentally, it has been found that interpolating every other frame is quite beneficial. Accordingly, for the illustrative purposes of this disclosure, the following describes the structure and operation of an encoder and a decoder that interpolates every other frame in accordance with the principles of this invention.

FIG. 1 describes an encoder of our invention. In FIG. 1, a video signal is applied to switch **10**. The switch toggles at the video frame rate and thus feed alternate frames to outputs A and B. The control is such that switch **10** is in position A when frame F_{i+1} is coming out of the video camera. The index i designates the frame number from some arbitrary starting point. During the previous video frame period, frame F_i came out of the camera, passed through output B of switch **10** and to the input of frame memory **16**. Now, frame F_i is coming out of frame memory **16**. It is frame F_i that will be interpolated in the decoder.

The following segment describes the operation of the motion compensation coding portion of the coder, which is well known to those skilled in the art.

Frame F_{i+1} passes to subtractor **20** and to motion estimator **11**. Frame memory **12** contains the frame that was previously coded via motion compensation; and in this case it is frame F_{i-1}. The output of memory **12** forms the other input to motion estimator **11**. For each block of pels, motion estimator **11** compares the pels of frames F_{i+1} and F_{i-1} to determine the best estimate of motion. The best estimate is delivered as a motion vector signal on bus **100**, and thus it passes to shift circuit **15**. Circuit **15** also accepts the pels information about the previous frame, F_{i-1}, from frame memory **12**, applies the appropriate translational shift according to the above-mentioned motion vector and outputs a block of "Prediction" pels to be used as a prediction of the incoming frame F_{i+1} pels.

This prediction block of pels passes to the other input of substractor **20** whereupon it is subtracted from the incoming pels of frame F_{i+1} to give a "Prediction Error" signal. The prediction error typically is transformed by DCT **30** and the output coefficients are quantized by quantizer **40**. The quantized values are coded into bits coder **50** and passed to buffer **60** to await transmission to the decoder.

From the above, it is clear that the input to the quantizer depends on the nature of the moving image, and consequently and as indicated above, it has the possibility of emptying or overflowing. To avoid this, a feedback path is provided to quantizer **40**, so that the quantizer coarseness can be increased if buffer overflow threatens, or decreased if buffer emptying is imminent.

Continuing with the description of motion compensated coding, the quantized output signals of quantizer **40** are inverse transformed by inverse DCT **41**, and applied to adder **42**. Adder **42** also receives the prediction pels of shift circuit **15** resulting in a coded version of frame i+1, F_{i+1}, which is passed into frame memory **12** for use with a subsequent frame as described above.

This completes the discussion of conventional motion compensation coding.

With the coded versions of frames i−1 and i+1, i.e., F_{i-1} and F_{i+1} being available, frame F_i can be generated.

The F_i generation starts with the motion vectors that are produced by motion estimator **11**. These are used by shift circuit **13** to shift the incoming pels from frame F_{i-1}, perhaps by half the motion vector, to produce one estimate of the pels in frame F_i. Circuit **14** also uses the motion vectors of line **100** to shift the coded pels of F_{i+1}, perhaps by half and in the opposite direction from the motion vector. This produces another estimate of the pels of F_i.

The two estimates produced by shift circuits **13** and **14** are combined in averager **17** to produce the final prediction of frame F_i. This interpolated prediction is usually very good, but not always.

To improve the interpolated prediction in accordance with our invention, subtractor **43** calculates an error signal that corresponds to the difference between the actual frame data that exits frame memory **16** (F_i) and the predicted frame as it appears at the output of averager **17** (F_i). The error signal is transformed by DCT **18**, quantized by quantizer **19** and passed to coder **44**, which detects large occurrences of interpolation error and codes them for transmission. The coded interpolation error is passed to buffer **60** in the same way as from coder **50**. Similarly, a feedback path is provided to quantizer **19** to combat buffer overflow and underflow.

The decoder, depicted in FIG. **2**, is very similar to the encoder. The components mirror corresponding components in the coder with a few deviations. In particular, the input is received in buffer **23** and is distributed therefrom based on the nautre of the signals. Frame encoding code (e.g. corresponding to F_{i-1} and F_{i+1}) is

sent to deocder **22** and therefrom to DCT^{-1} **24**, adder **27**, memory **28** and shift circuit **26**. These elements correspond to elements **41**, **42**, **12**, and **15**, respectively, and operate in the same manner. That is completely expected, since the function of these elements in the encoder is to emulate the decoder. Thus, the contents of memory **28** correspond to the estimated frames. Similarly, elements **39**, **31** and **32** in the decoder correspond to elements **13**, **14** and **17**, respectively in the encoder and operate in the same manner.

The pels correction code also exits buffer **23**, is decoded in decoder **25** and inverse transformed in element **34**. This correction information is added to the estimate of F_l developed by circuit **35** and is applied to memory **33**. Memory **33** delays the F_l information to permit a proper interleaving of F_l between F_{l-1} and F_{l+1}. As can be observed from above, one of the deviations is that the interpolation error subtractor **43** of the encoder becomes adder **35** at the deocder. Also, another output of frame memory **28** is shown since frame F_{l-1} pels for the video output display may need to be read out at a different rate for the video output at switch **21** than the frame F_{l-1} pels are needed for shift circuits **26** and **39**.

It may be noted that there is a tradeoff between the buffer size of buffer **23** and the necessity for frame memory **33**. If the buffer is sufficiently large, frame memory **33** could be deleted. The frame F_l output from adder **35** would then pass directly to the video output via switch **21**, which would be in position B. Following this, switch **21** would toggle to its A input, and decoding would stop for a frame period while frame F_{l+1} was displayed via the output of frame memory **28** and the A input of switch **21**. During this time, decoder buffer **23** would fill with data from the channel.

Many alternative arrangements are possible for the basic conditional motion compensation interpolation approach. For example, more than one frame might be conditionally interpolated, in which case shifter circuits **13**, **14**, **30** and **31** need to be more versatile and frame memories **16** and **33** need to be larger. Also in computing the best estimnate of motion, motion estimator **11** might take frame F_l pels as additional input. This would enable simultaneous minimization of both motion compensation prediction error as well as interpolation error. Still other improvements may be introduced by skilled artisans practicing this invention without departing from the spirit and scope thereof.

We claim:

1. A circuit for encoding applied video signals that comprise successive frames, where each frame is divided into blocks, comprising:

first means for encoding the blocks of some of said frames by developing for each block of such frames (a) and approximated version of said block derived from an approximated version of said block developed for a previous frame, and (b) a code which represents the deviation of said block from said approximated version of said block;

second means for approximating the blocks of those of said frames that are to be interpolated by combining approximated versions of said blocks in selected ones of the frames that are encoded in said first means; and

third means responsive to said second means and to said frames to be interpolated for developing a code that corresponds to those pels in blocks approximated by said second means that differ from corresponding pels in said frames to be interpolated by greater than a preselected threshold.

2. A circuit for encoding applied video signals that comprise successive frames, where each frame is divided into blocks, including means for encoding the blocks of some of said frames by developing for each block of such frames (a) an approximated version of said block derived from an approximated version of said block developed for a previous frame, and (b) a code which represents the deviation of said block from said approximated version of said block, the improvement comprising:

second means for approximating the blocks of those of said frames that are to be interpolated by combining approximated versions of said blocks in selected ones of the frames that are encoded in said means for encoding; and

third means responsive to said second means and to said frames to be interpolated for developing code that corresponds to those pels in blocks approximated by said second means that differ from corresponding pels in said frames to be interpolated by greater than a preselected threshold.

3. The circuit of claim **2** wherein said code developed for a pel by said third means represents the difference between the value of said pel and the value of said pel approximated by said second means.

4. The circuit of claim **2** wherein the frames selected for combining in said second means include a frame encoded in said first means that precedes the frame approximated in said second means and a frame encoded in said first means that succeeds the frame approximated in said means.

5. The circuit of claim **4** wherein said combining includes developing anticipated versions of said blocks.

6. The circuit of claim **2** wherein a set proportion of frames of said applied video signals are interpolated.

7. The circuit of claim **6** wherein said proportion is approximately one half.

8. The circuit of claim **2** fruther comprising buffer means for interposed between the codes developed by said means for encoding and said third means and an output port of said circuit.

9. The circuit of claim **8** for controlling the proportion of frames selected for interpolation by said second means and code generation by said third means based on the occupancy level of said buffer.

10. The circuit of claim **8** for selecting frames for interpolation by said second means and code generation by said third means when said buffer is occupied beyond a chosen proportion of its capacity.

11. The circuit of claim **7** wherein granularity of the codes generated by said first means and said third means is controlled by the occupancy level of said buffer.

12. A circuit responsive to coded video signals where the video signals comprise successive frames and each frame includes a plurality of blocks and where the coded video signals comprise codes that describe deviations from approximated blocks and codes that describe deviations from interpolated blocks, comprising:

means for developing block approximations from said codes that describe deviations from approximated blocks; and

means responsive to said block approximations and to said codes that describe deviations from interpolated blocks to develop said interpolated blocks.

* * * * *

Exhibit B

US005227878A

United States Patent [19]

Puri et al.

[11] Patent Number: 5,227,878

[45] Date of Patent: Jul. 13, 1993

[54] **ADAPTIVE CODING AND DECODING OF FRAMES AND FIELDS OF VIDEO**

[75] Inventors: **Atul Puri**, New York, N.Y.; **Rangarajan Aravind**, Matawan, N.J.

[73] Assignee: **AT&T Bell Laboratories**, Murray Hill, N.J.

[21] Appl. No.: **793,063**

[22] Filed: **Nov. 15, 1991**

[51] **Int. Cl.**[5] **H04N 3/133**; H04N 3/137

[52] **U.S. Cl.** **358/136**; 358/105

[58] **Field of Search** 358/133, 135, 136, 138, 358/105

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,437,087	3/1984	Petr	375/27
4,475,213	10/1984	Medaugh	375/27
4,958,226	9/1990	Haskell et al.	358/136
4,969,040	11/1990	Charavi	358/136
4,999,705	3/1991	Puri	358/136
5,001,561	3/1991	Haskell et al.	358/133
5,091,782	2/1992	Krause	358/136

OTHER PUBLICATIONS

B. Astle, "A Proposal For MPEG Video Report, Rev. 1," Int'l Organization For Standardization, ISO-IEC/JTC1/SC2/WG11, Aug. 14, 1991, pp. 1–78.

"Coding Of Moving Pictures And Associated Audio—For Digital Storage Media At Up To About 1.5 Mbit/-s—Part 2 Video," 1991, pp. 2–51.

A. N. Netraveli et al., "Digital Pictures Representation and Compression," (Plenum Press, New York 1988), pp. 301–504.

Primary Examiner—Howard W. Britton

Attorney, Agent, or Firm—E. S. Indyk

[57] **ABSTRACT**

Improved compression of digital signals relating to high resolution video images is accomplished by an adaptive and selective coding of digital signals relating to frames and fields of the video images. Digital video input signals are analyzed and a coding type signal is produced in response to this analysis. This coding type signal may be used to adaptively control the operation of one or more types of circuitry which are used to compress digital video signals so that less bits, and slower bit rates, may be used to transmit high resolution video images without undue loss of quality. For example, the coding type signal may be used to improve motion compensated estimation techniques, quantization of transform coefficients, scanning of video data, and variable word length encoding of the data. The improved compression of digital video signals is useful for video conferencing applications and high definition television, among other things.

33 Claims, 20 Drawing Sheets

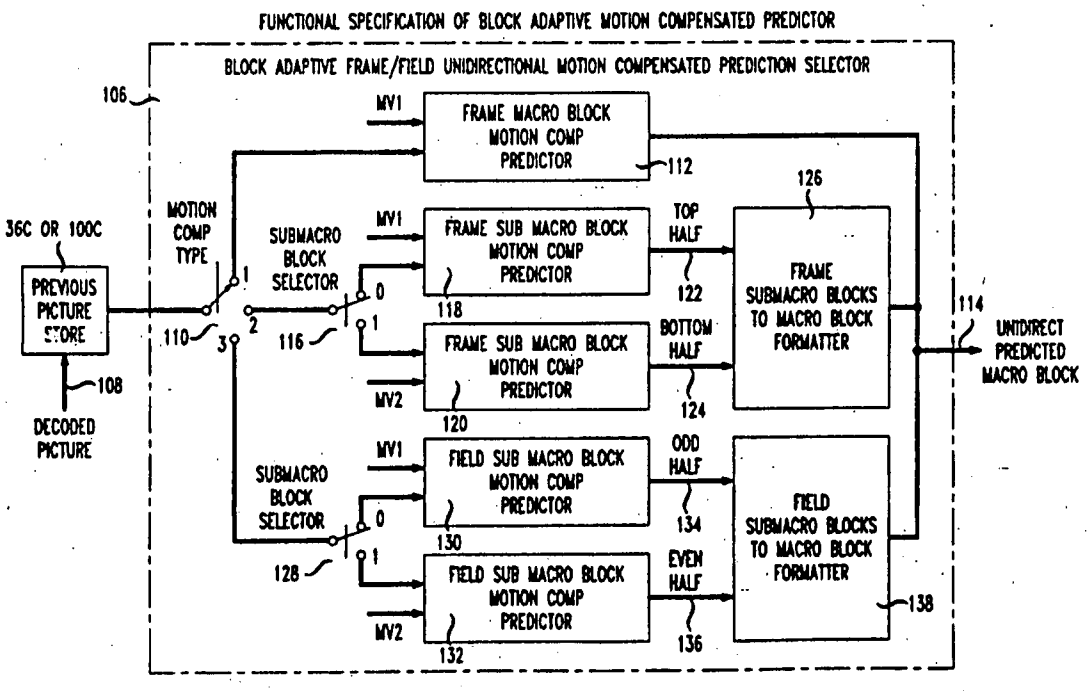


FIG. 1A

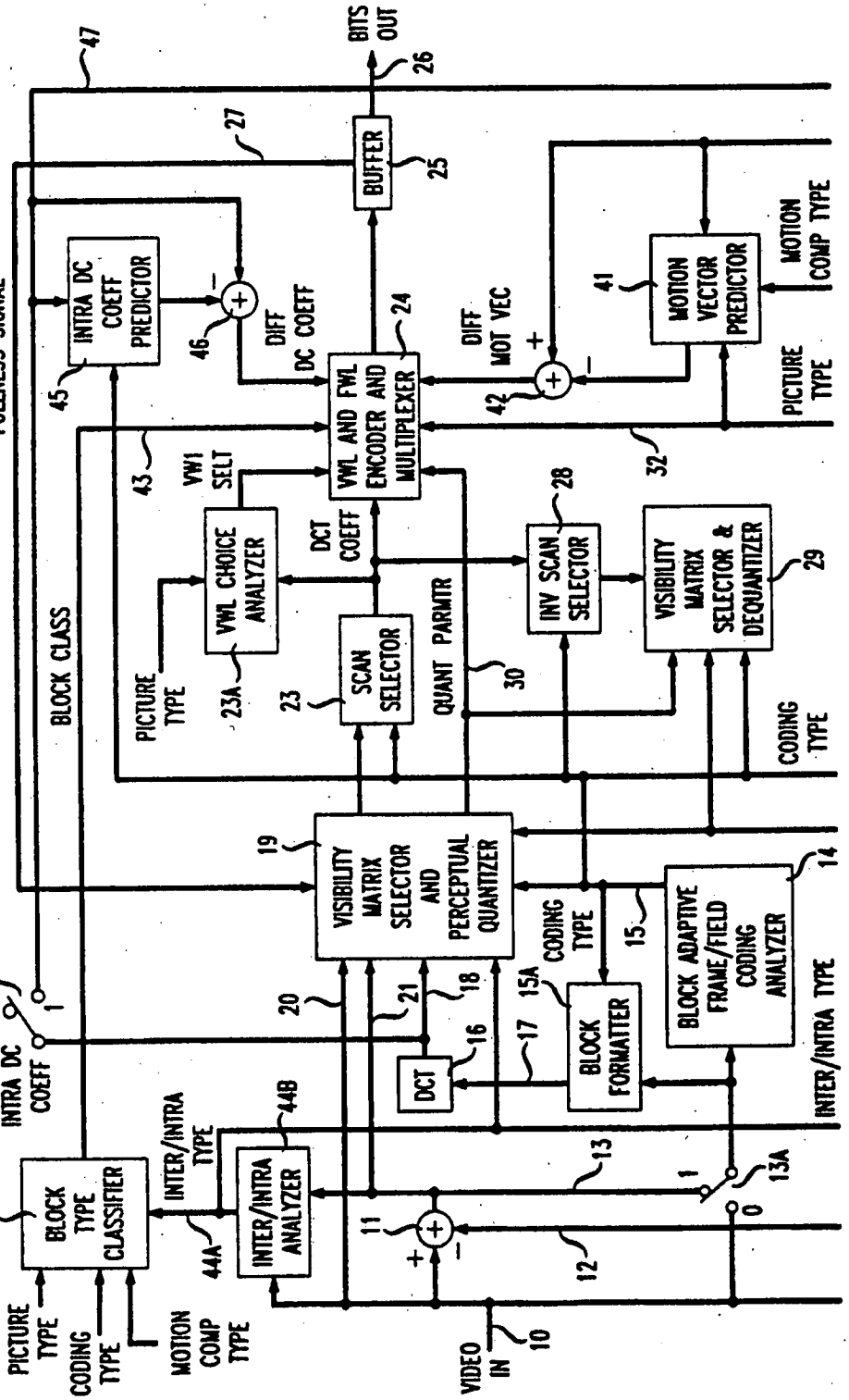

BITS OUT
26
47
27
BUFFER
25
FULLNESS SIGNAL
INTRA DC COEFF PREDICTOR
45
46
DIFF DC COEFF
24
DIFF MOT VEC
41
MOTION VECTOR PREDICTOR
MOTION COMP TYPE
PICTURE TYPE
42
32
VWL AND FWL ENCODER AND MULTIPLEXER
43
VWI SELT
VWL CHOICE ANALYZER
DCT COEFF
28
INV SCAN SELECTOR
VISIBILITY MATRIX SELECTOR & DEQUANTIZER
29
PICTURE TYPE
23A
23
SCAN SELECTOR
QUANT PARMTR
30
BLOCK CLASS
CODING TYPE
19
VISIBILITY MATRIX SELECTOR AND PERCEPTUAL QUANTIZER
14
CODING TYPE
15
BLOCK ADAPTIVE FRAME/FIELD CODING ANALYZER
48
1
20
21
18
15A
17
16
DCT
BLOCK FORMATTER
INTRA DC COEFF
INTER/INTRA TYPE
44B
13
1
13A
0
44
BLOCK TYPE CLASSIFIER
INTER/INTRA TYPE
INTER/INTRA ANALYZER
44A
11
12
PICTURE TYPE
CODING TYPE
MOTION COMP TYPE
VIDEO IN
10

FIG. 1C

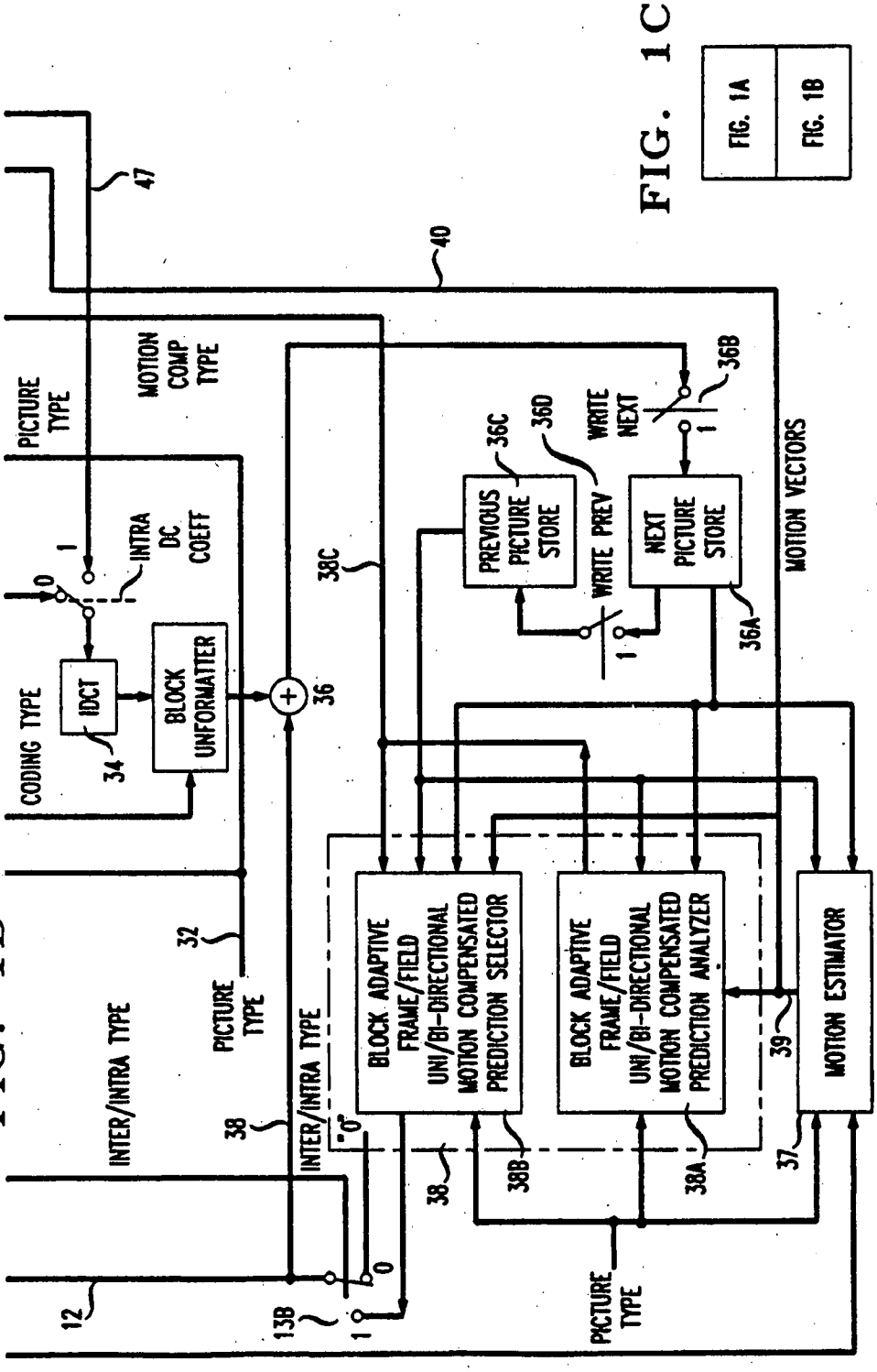


FIG. 1A
FIG. 1B
47
40
PICTURE TYPE
MOTION COMP TYPE
INTRA DC COEFF
CODING TYPE
IDCT
34
BLOCK UNFORMATTER
36
38C
PREVIOUS PICTURE STORE
36C
WRITE PREV
36D
WRITE NEXT
NEXT PICTURE STORE
36B
36A
MOTION VECTORS
32
PICTURE TYPE
INTER/INTRA TYPE
38
BLOCK ADAPTIVE FRAME/FIELD UNI/BI-DIRECTIONAL MOTION COMPENSATED PREDICTION SELECTOR
BLOCK ADAPTIVE FRAME/FIELD UNI/BI-DIRECTIONAL MOTION COMPENSATED PREDICTION ANALYZER
39
MOTION ESTIMATOR
38B
38A
37
12
13B
PICTURE TYPE

FIG. 1D

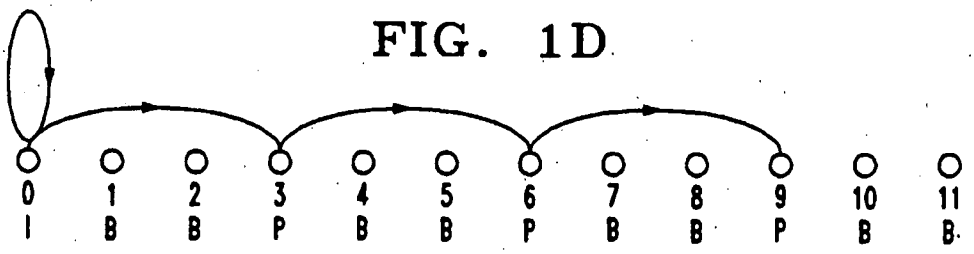


FIG. 1E

FRAME BLOCK SCAN

FRAME SCAN

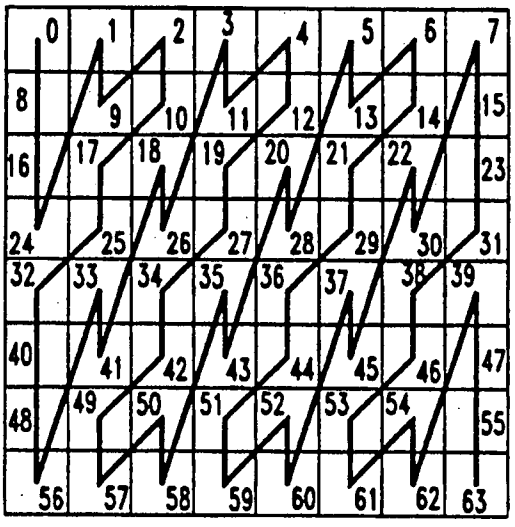


FIG. 1F

FIELD BLOCK SCAN

FIELD SCANS

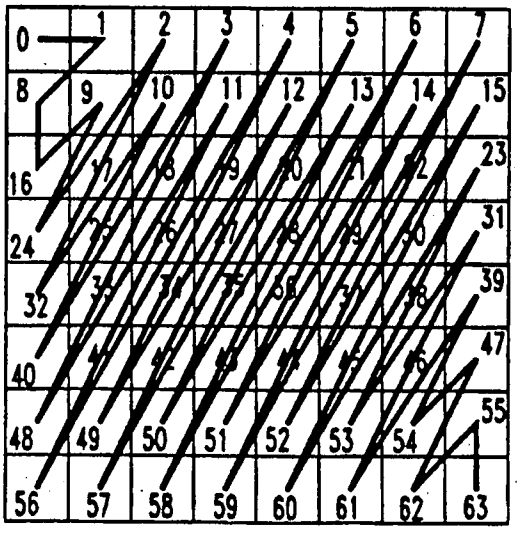


FIG. 1G

LENGTHS OF INTRA VLC #1 (EOB = 2 BITS)

RUN \ LEVEL (ABSOLUTE VALUE)	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	3	4	5	6	6	7	8	8	9	9	9	11	11	11
1	5	7	9	11										
2	6	9												
3	7	11												
4	7													
5	8													
6	8													
7	9													
8	9													
9	9													
10	11													
11	11													
12	13													

FIG. 1H

LENGTHS OF INTRA VLC #2 (EOB = 2 BITS)

RUN \ LEVEL (ABSOLUTE VALUE)	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	3	5	6	7	9	9	11	11						
1	4	6	9	11										
2	5	8	11											
3	6	9												
4	7	11												
5	7	11												
6	7													
7	8													
8	8													
9	8													
10	9													
11	9													
12	9													
13	9													
14	11													
15	11													

FIG. 1I

LENGTHS OF INTRA VLC #3 (EOB = 3 BITS)

RUN \ LEVEL (ABSOLUTE VALUE)	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	3	4	4	6	6	7	8	8	9	9	9	11	11	11
1	4	7	9	11										
2	6	9												
3	7	11												
4	7													
5	8													
6	8													
7	9													
8	9													
9	9													
10	11													
11	11													
12	13													

FIG. 1J

LENGTHS OF INTRA VLC #4 (EOB = 4 BITS)

RUN \ LEVEL (ABSOLUTE VALUE)	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	3	3	5	6	6	7	8	8	9	9	9	11	11	11
1	4	7	9	11										
2	6	9												
3	7	11												
4	7													
5	8													
6	8													
7	9													
8	9													
9	9													
10	11													
11	11													
12	13													

FIG. 1K

LENGTHS OF PRED. VLC #1 (EOB = 2 BITS)

RUN \ LEVEL (ABSOLUTE VALUE)	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	3	5	6	7	9	9	11	11						
1	4	6	9	11										
2	5	8	11											
3	6	9												
4	7	11												
5	7	11												
6	7													
7	8													
8	8													
9	8													
10	9													
11	9													
12	9													
13	9													
14	11													
15	11													

FIG. 1L

LENGTHS OF PRED. VLC #2 (EOB = 2 BITS)

RUN \ LEVEL (ABSOLUTE VALUE)	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	3	4	5	6	6	7	8	8	9	9	9	11	11	11
1	5	7	9	11										
2	6	9												
3	7	11												
4	7													
5	8													
6	8													
7	9													
8	9													
9	9													
10	11													
11	11													
12	13													

FIG. 1M

LENGTHS OF PRED. VLC #3 (EOB = 3 BITS)

RUN \ LEVEL (ABSOLUTE VALUE)	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	3	4	6	7	9	9	11	11						
1	4	6	9	11										
2	4	8	11											
3	6	9												
4	7	11												
5	7	11												
6	7													
7	8													
8	8													
9	8													
10	9													
11	9													
12	9													
13	9													
14	11													
15	11													

FIG. 1N

LENGTHS OF PRED. VLC #4 (EOB = 4 BITS)

RUN \ LEVEL (ABSOLUTE VALUE)	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	3	4	6	7	9	9	11	11						
1	3	6	9	11										
2	5	8	11											
3	6	9												
4	7	11												
5	7	11												
6	7													
7	8													
8	8													
9	8													
10	9													
11	9													
12	9													
13	9													
14	11													
15	11													

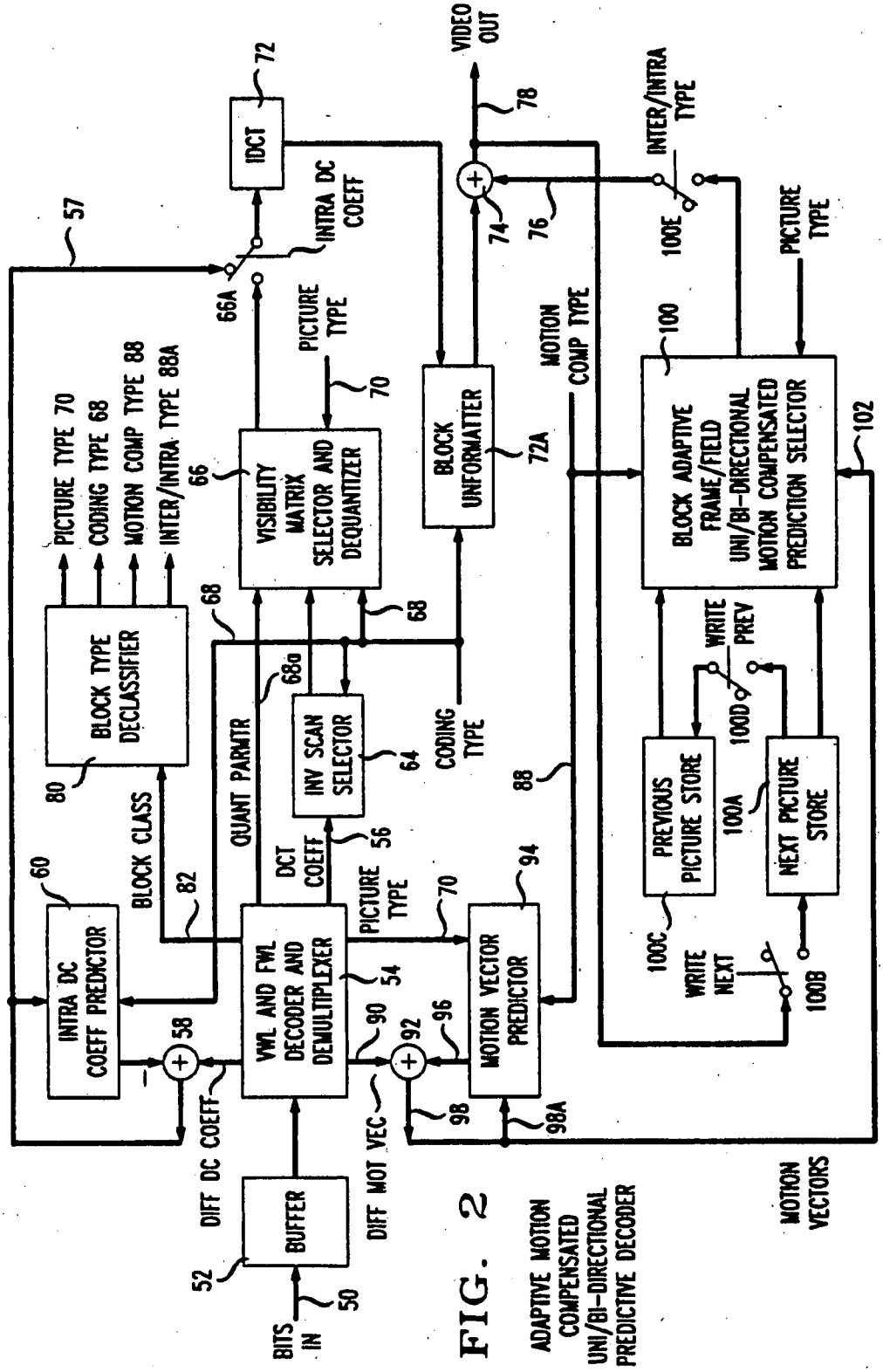


FIG. 2

ADAPTIVE MOTION COMPENSATED UNI/BI-DIRECTIONAL PREDICTIVE DECODER

FIG. 3

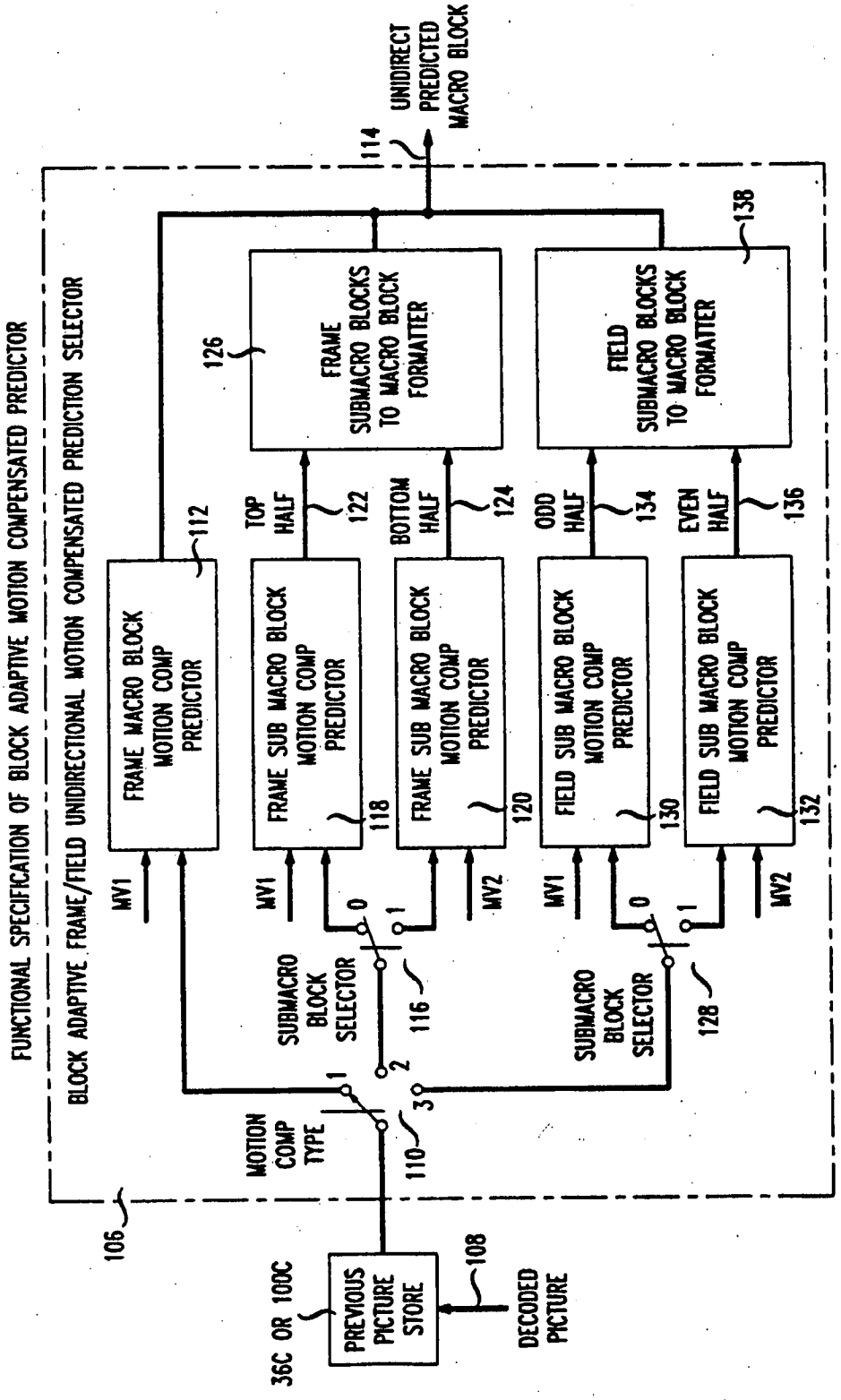
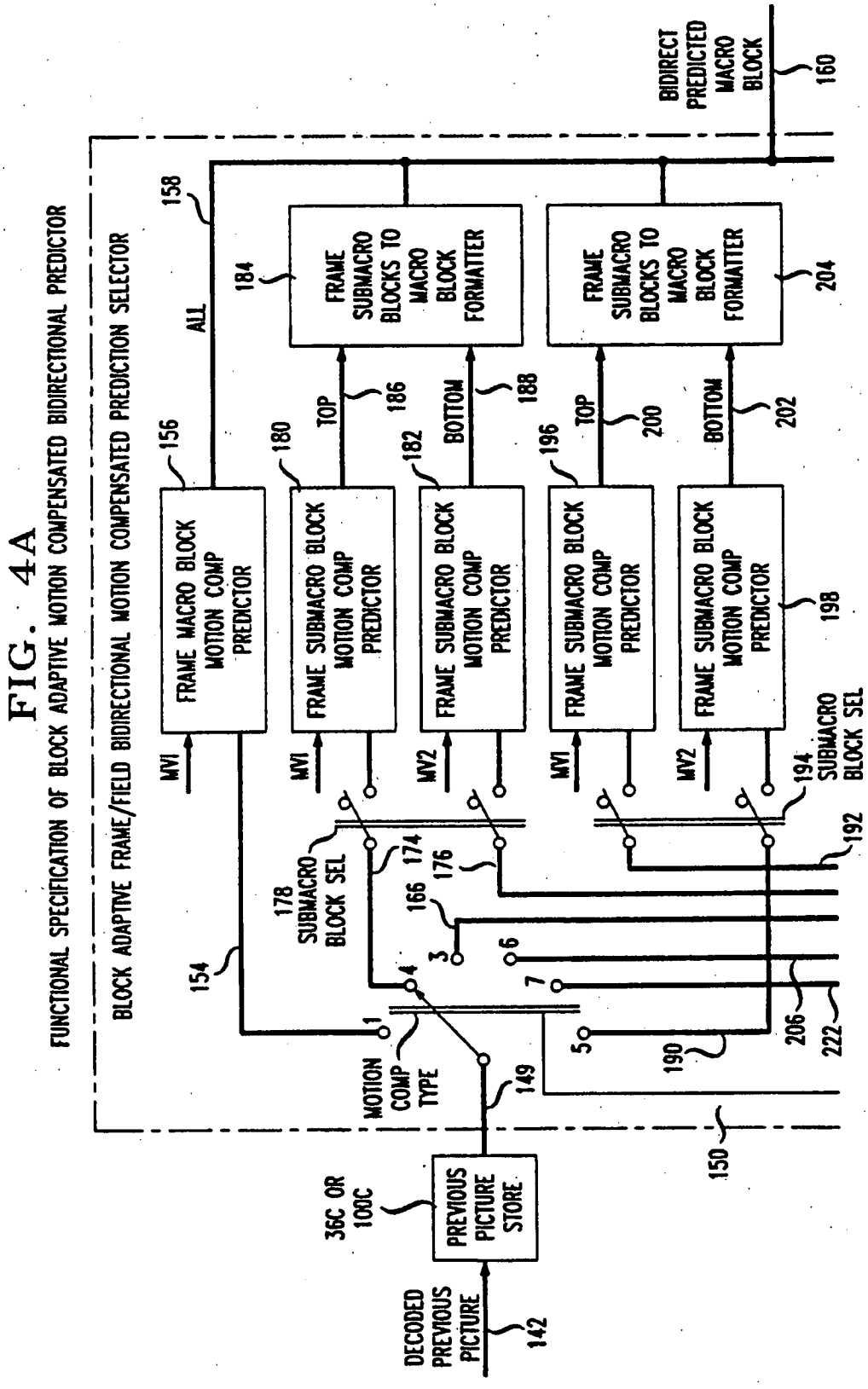

FUNCTIONAL SPECIFICATION OF BLOCK ADAPTIVE MOTION COMPENSATED PREDICTOR
BLOCK ADAPTIVE FRAME/FIELD UNIDIRECTIONAL MOTION COMPENSATED PREDICTION SELECTOR
UNIDIRECT PREDICTED MACRO BLOCK
114
FRAME SUBMACRO BLOCKS TO MACRO BLOCK FORMATTER
126
FIELD SUBMACRO BLOCKS TO MACRO BLOCK FORMATTER
138
TOP HALF
122
BOTTOM HALF
124
ODD HALF
134
EVEN HALF
136
FRAME MACRO BLOCK MOTION COMP PREDICTOR
112
FRAME SUB MACRO BLOCK MOTION COMP PREDICTOR
118
FRAME SUB MACRO BLOCK MOTION COMP PREDICTOR
120
FIELD SUB MACRO BLOCK MOTION COMP PREDICTOR
130
FIELD SUB MACRO BLOCK MOTION COMP PREDICTOR
132
MV1
MV1
MV2
MV1
MV2
SUBMACRO BLOCK SELECTOR
116
SUBMACRO BLOCK SELECTOR
128
MOTION COMP TYPE
110
106
PREVIOUS PICTURE STORE
108
36C OR 100C
DECODED PICTURE

FIG. 4A


FUNCTIONAL SPECIFICATION OF BLOCK ADAPTIVE MOTION COMPENSATED BIDIRECTIONAL PREDICTOR
BLOCK ADAPTIVE FRAME/FIELD BIDIRECTIONAL MOTION COMPENSATED PREDICTION SELECTOR
BIDIRECT PREDICTED MACRO BLOCK
160
158
ALL
184
FRAME SUBMACRO BLOCKS TO MACRO BLOCK FORMATTER
204
FRAME SUBMACRO BLOCKS TO MACRO BLOCK FORMATTER
156
FRAME MACRO BLOCK MOTION COMP PREDICTOR
180
TOP
186
FRAME SUBMACRO BLOCK MOTION COMP PREDICTOR
182
BOTTOM
188
FRAME SUBMACRO BLOCK MOTION COMP PREDICTOR
196
TOP
200
FRAME SUBMACRO BLOCK MOTION COMP PREDICTOR
BOTTOM
202
FRAME SUBMACRO BLOCK MOTION COMP PREDICTOR
198
MV1
MV1
MV2
MV1
MV2
194
SUBMACRO BLOCK SEL
192
178
SUBMACRO BLOCK SEL
174
176
166
154
3
6
7
4
1
5
206
222
190
MOTION COMP TYPE
149
150
36C OR 100C
PREVIOUS PICTURE STORE
DECODED PREVIOUS PICTURE
142

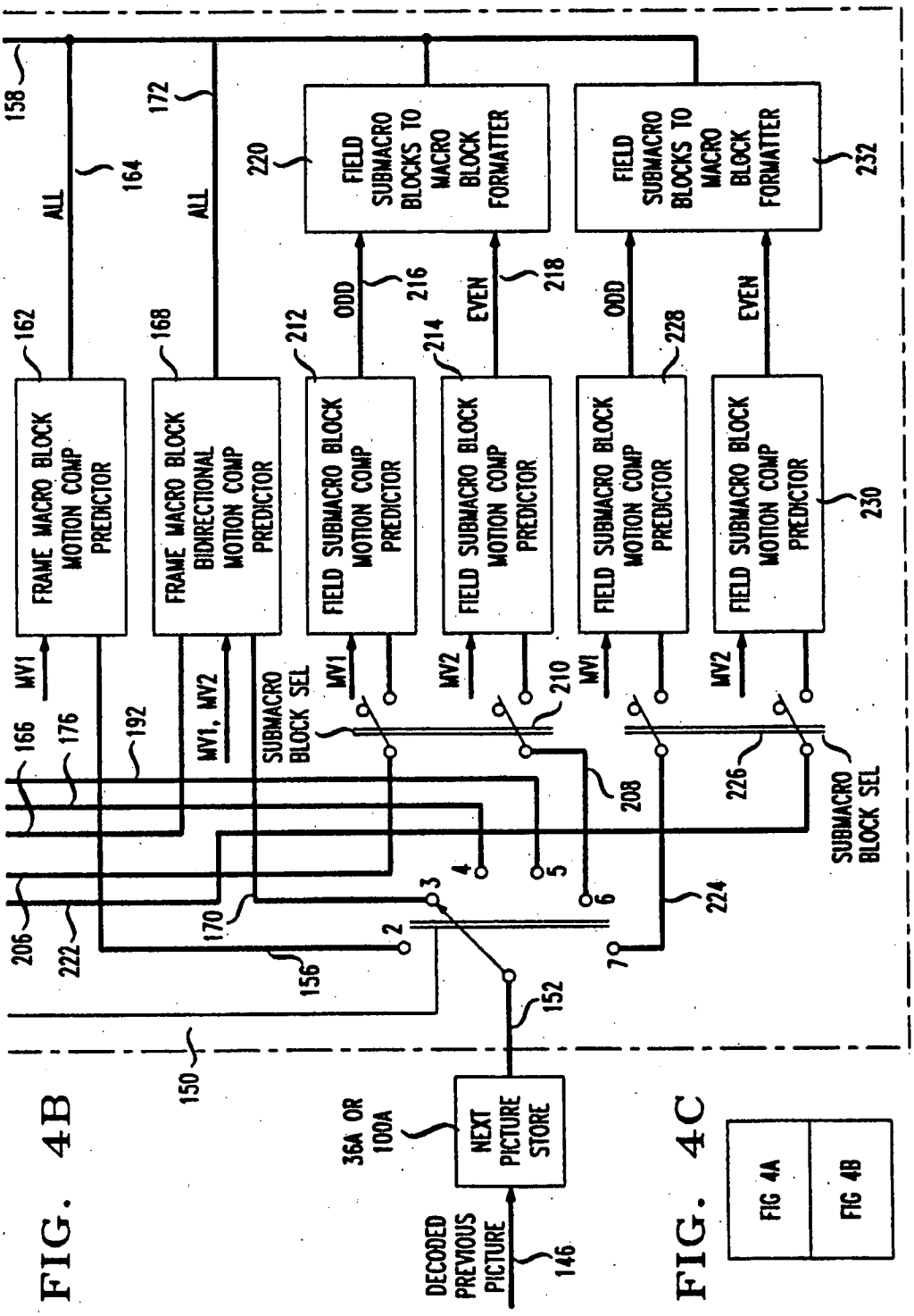

158
172
ALL
164
ALL
220
FIELD SUBMACRO BLOCKS TO MACRO BLOCK FORMATTER
FIELD SUBMACRO BLOCKS TO MACRO BLOCK FORMATTER
232
ODD
216
EVEN
218
ODD
228
EVEN
162
168
212
214
FRAME MACRO BLOCK MOTION COMP PREDICTOR
FRAME MACRO BLOCK BIDIRECTIONAL MOTION COMP PREDICTOR
FIELD SUBMACRO BLOCK MOTION COMP PREDICTOR
FIELD SUBMACRO BLOCK MOTION COMP PREDICTOR
FIELD SUBMACRO BLOCK MOTION COMP PREDICTOR
FIELD SUBMACRO BLOCK MOTION COMP PREDICTOR
230
MV1
MV1, MV2
SUBMACRO BLOCK SEL
MV1
MV2
210
MV1
MV2
166
176
192
208
226
SUBMACRO BLOCK SEL
206
222
170
156
2
3
4
5
6
7
224
152
150
36A OR 100A
NEXT PICTURE STORE
DECODED PREVIOUS PICTURE
146

FIG. 4B

FIG. 4C

FIG 4A
FIG 4B

FIG. 5

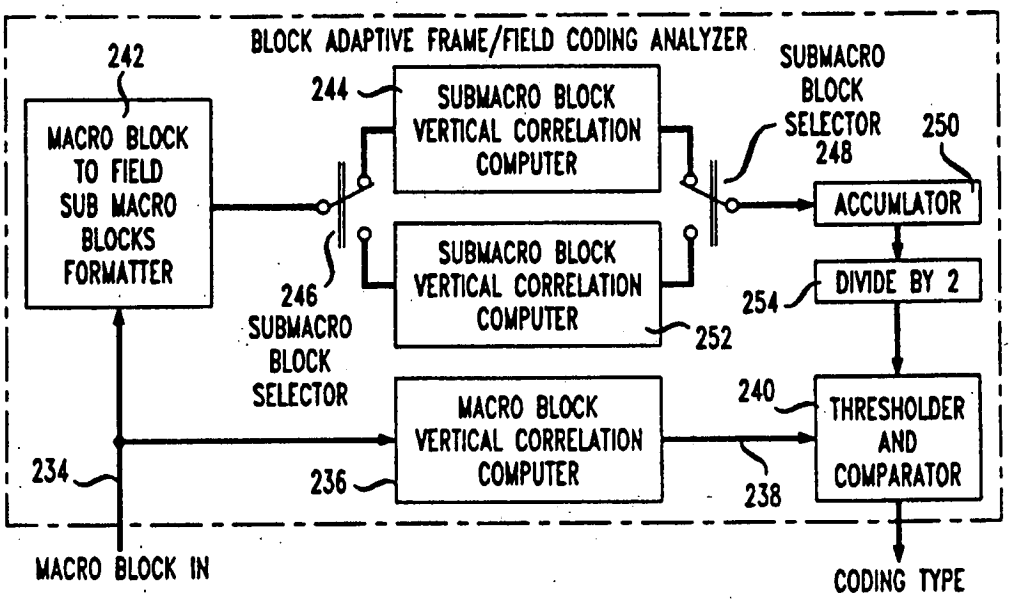


FIG. 6

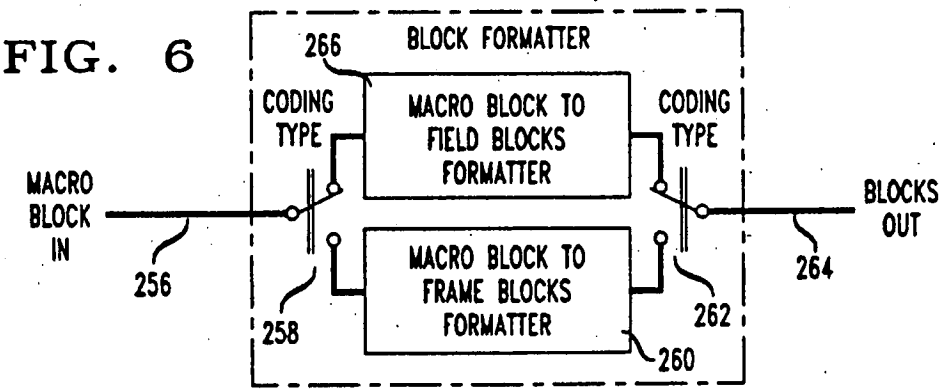


FIG. 7

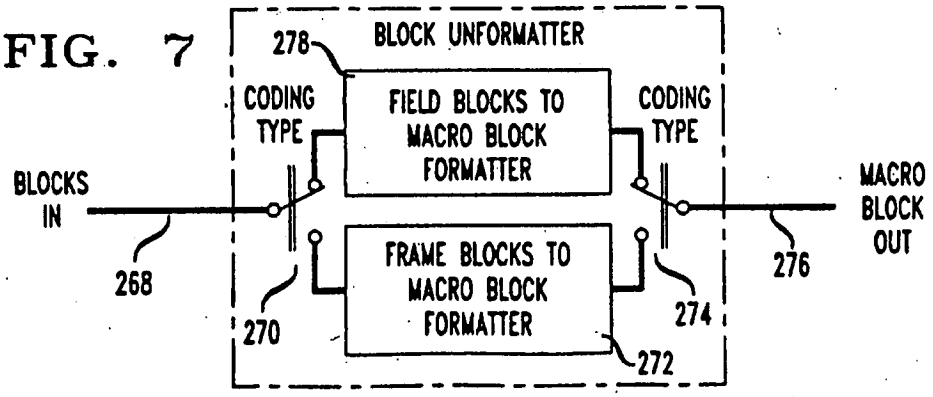


FIG. 8

FLOWCHART FOR INTRA DC PREDICTION

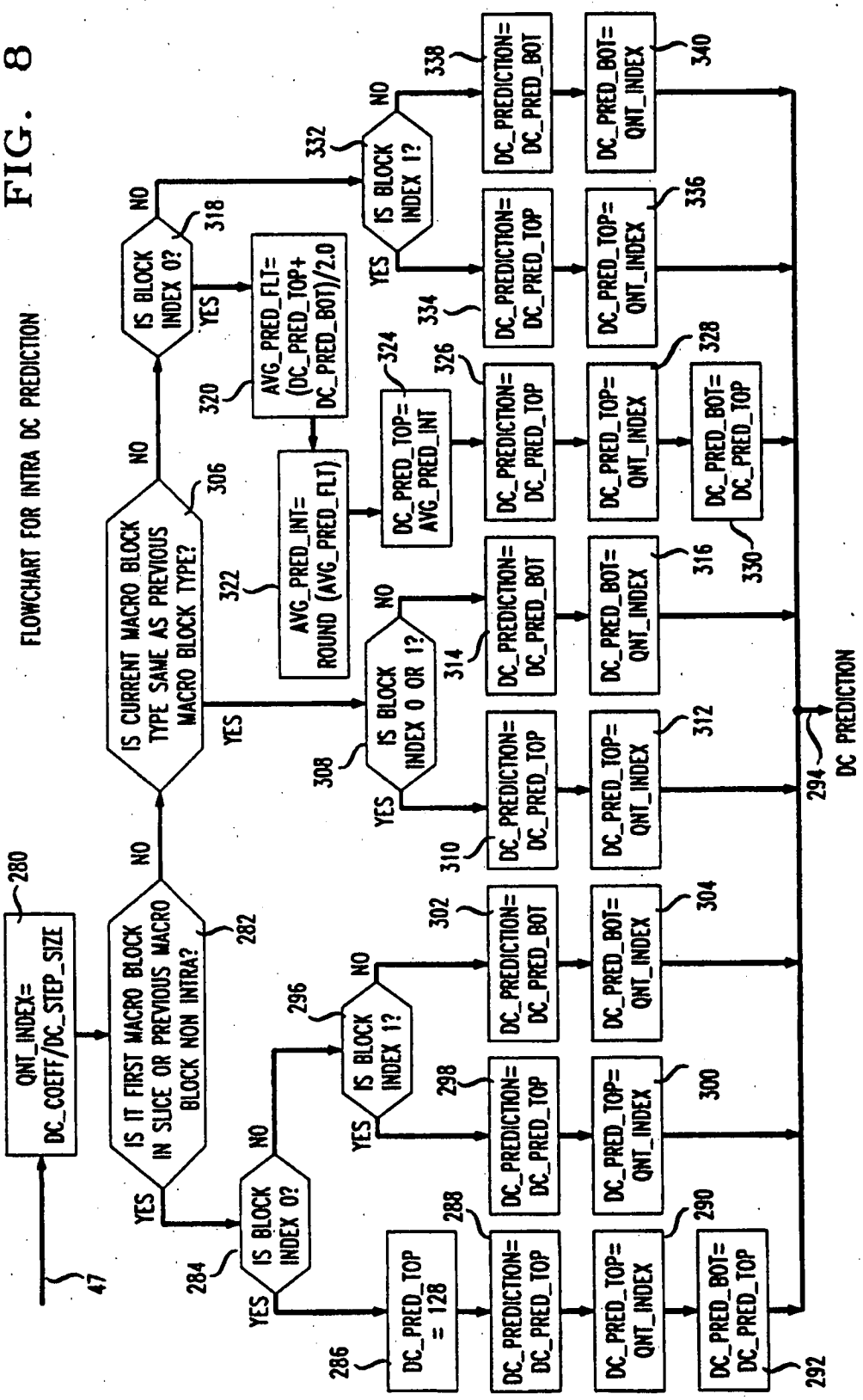


FIG. 9

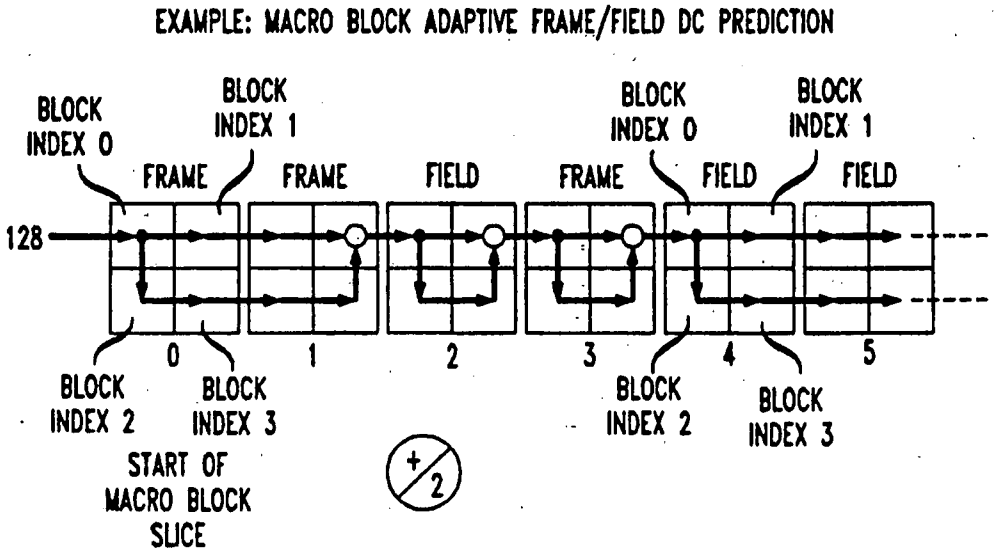
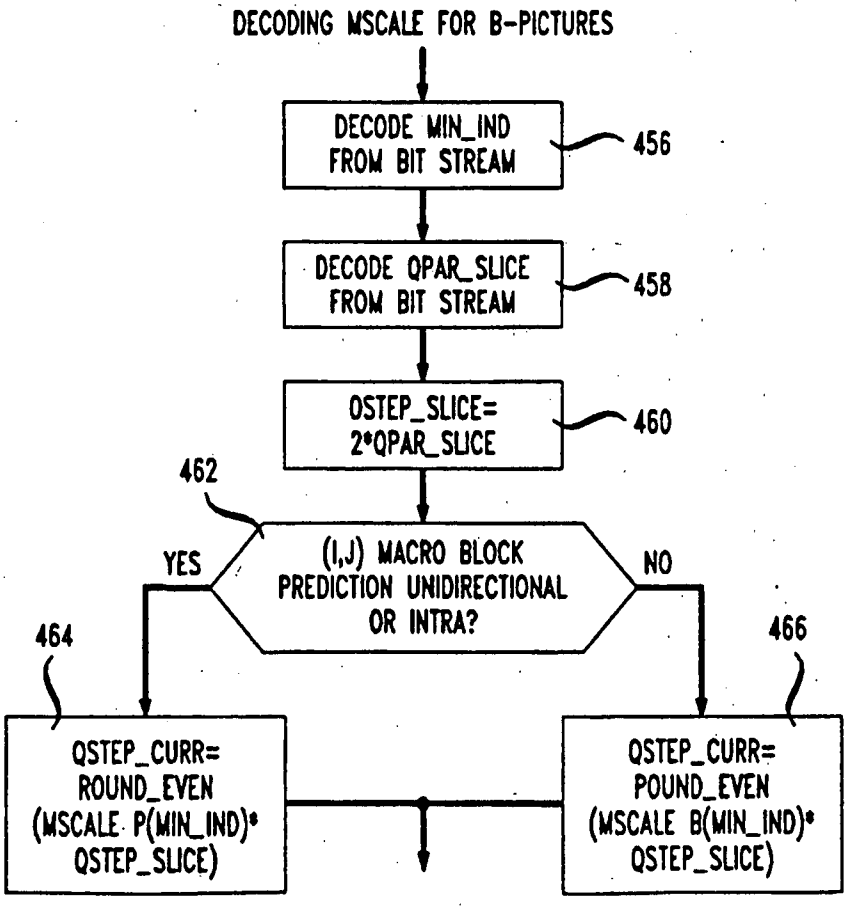

EXAMPLE: MACRO BLOCK ADAPTIVE FRAME/FIELD DC PREDICTION
BLOCK INDEX 0
BLOCK INDEX 1
FRAME
FRAME
FIELD
FRAME
BLOCK INDEX 0
FIELD
BLOCK INDEX 1
FIELD
128
0
1
2
3
4
5
BLOCK INDEX 2
BLOCK INDEX 3
BLOCK INDEX 2
BLOCK INDEX 3
START OF MACRO BLOCK SLICE
+/2

FIG. 12


DECODING MSCALE FOR B-PICTURES
DECODE MIN_IND FROM BIT STREAM
456
DECODE QPAR_SLICE FROM BIT STREAM
458
QSTEP_SLICE= 2*QPAR_SLICE
460
462
(I,J) MACRO BLOCK PREDICTION UNIDIRECTIONAL OR INTRA?
YES
NO
464
466
QSTEP_CURR= ROUND_EVEN (MSCALE P(MIN_IND)* QSTEP_SLICE)
QSTEP_CURR= POUND_EVEN (MSCALE B(MIN_IND)* QSTEP_SLICE)

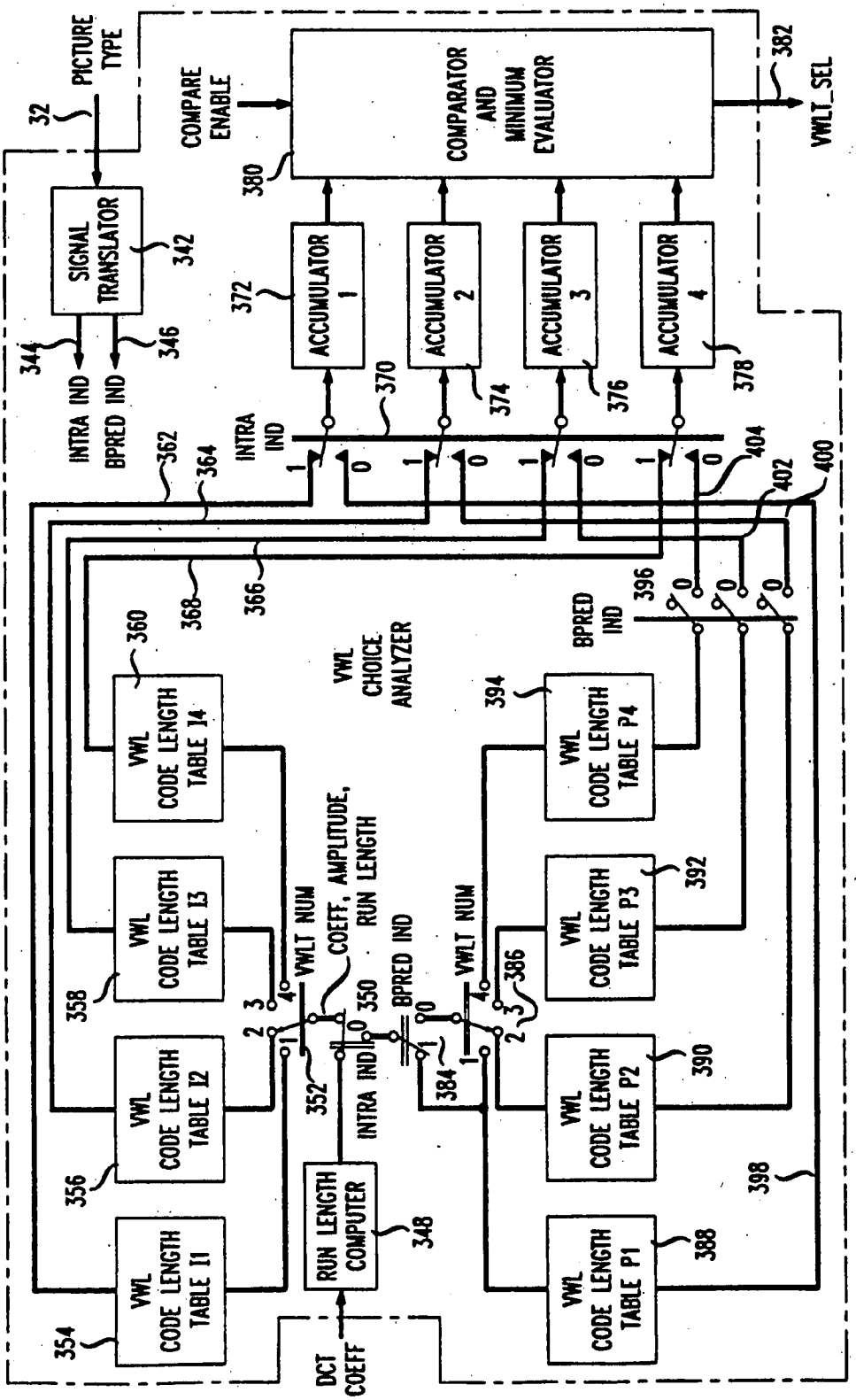

PICTURE TYPE
32
COMPARE ENABLE
COMPARATOR AND MINIMUM EVALUATOR
380
382
VWLI_SEL
SIGNAL TRANSLATOR
342
344
346
INTRA IND
BPRED IND
ACCUMULATOR 1
ACCUMULATOR 2
ACCUMULATOR 3
ACCUMULATOR 4
372
370
374
376
378
INTRA IND
362
364
404
402
400
396
BPRED IND
368
366
VWL CHOICE ANALYZER
VWL CODE LENGTH TABLE I4
VWL CODE LENGTH TABLE I3
VWL CODE LENGTH TABLE I2
VWL CODE LENGTH TABLE I1
VWL CODE LENGTH TABLE P4
VWL CODE LENGTH TABLE P3
VWL CODE LENGTH TABLE P2
VWL CODE LENGTH TABLE P1
360
358
356
354
394
392
390
388
VWLI NUM
COEFF, AMPLITUDE, RUN LENGTH
BPRED IND
INTRA IND
350
352
384
386
RUN LENGTH COMPUTER
348
398
DCT COEFF

FIG. 10

FIG. 11

ENCODING MSCALE FOR B-PICTURES

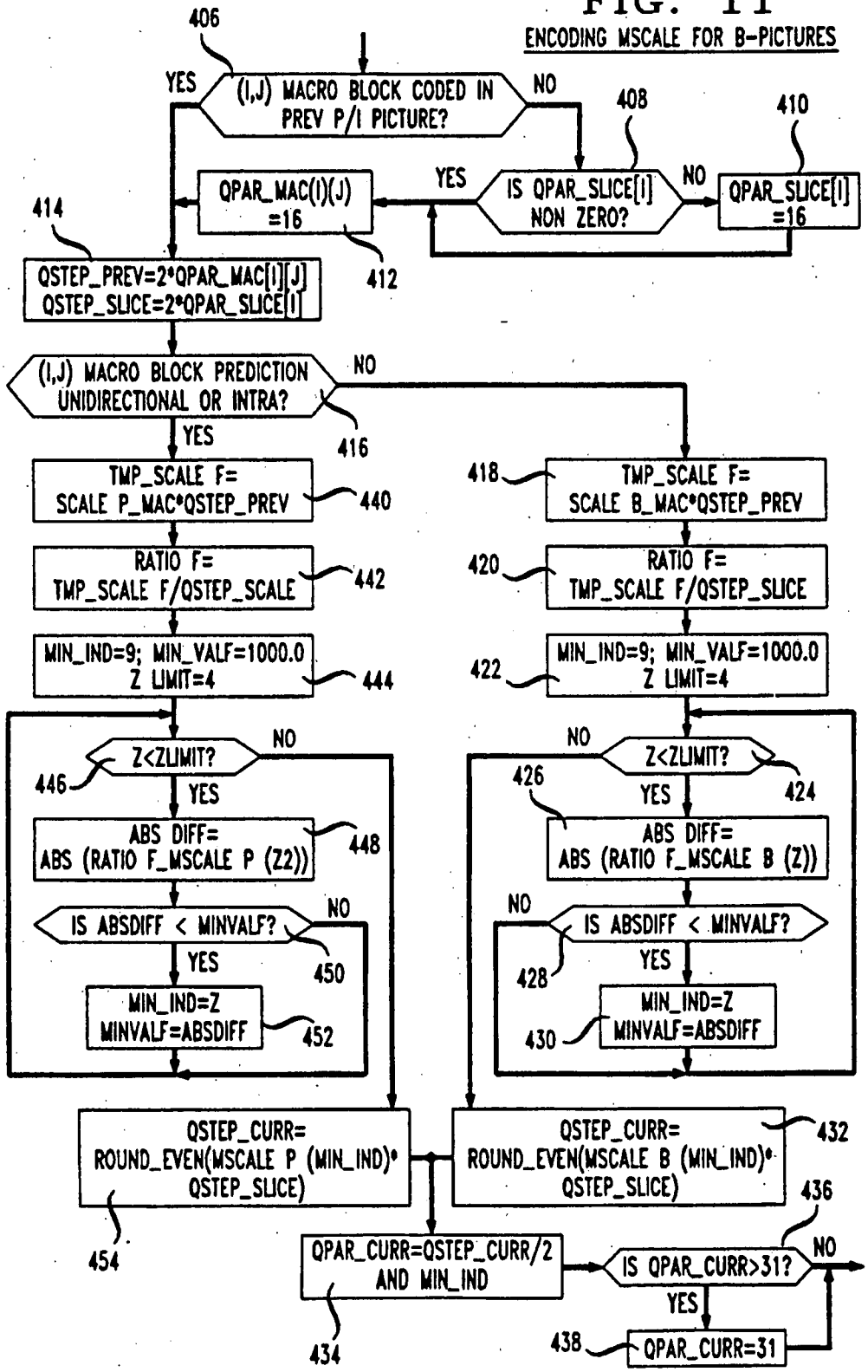


FIG. 13

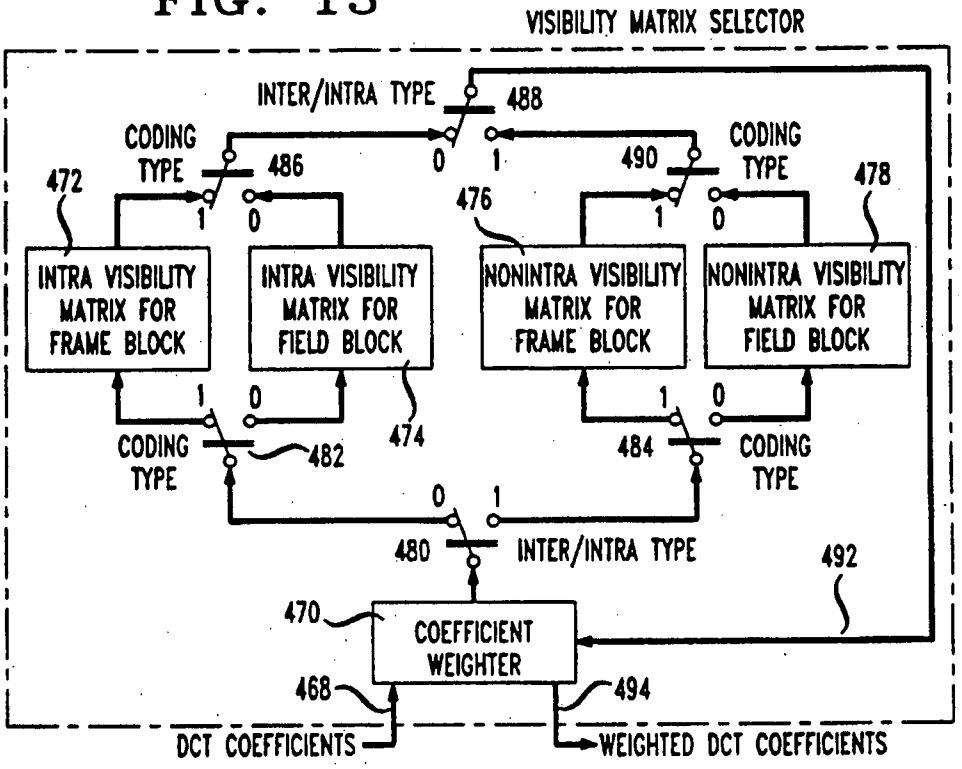

VISIBILITY MATRIX SELECTOR
INTER/INTRA TYPE
488
CODING TYPE
486
490
CODING TYPE
472
476
478
INTRA VISIBILITY MATRIX FOR FRAME BLOCK
INTRA VISIBILITY MATRIX FOR FIELD BLOCK
NONINTRA VISIBILITY MATRIX FOR FRAME BLOCK
NONINTRA VISIBILITY MATRIX FOR FIELD BLOCK
CODING TYPE
482
474
484
CODING TYPE
480
INTER/INTRA TYPE
492
470
COEFFICIENT WEIGHTER
468
494
DCT COEFFICIENTS
WEIGHTED DCT COEFFICIENTS

FIG. 14

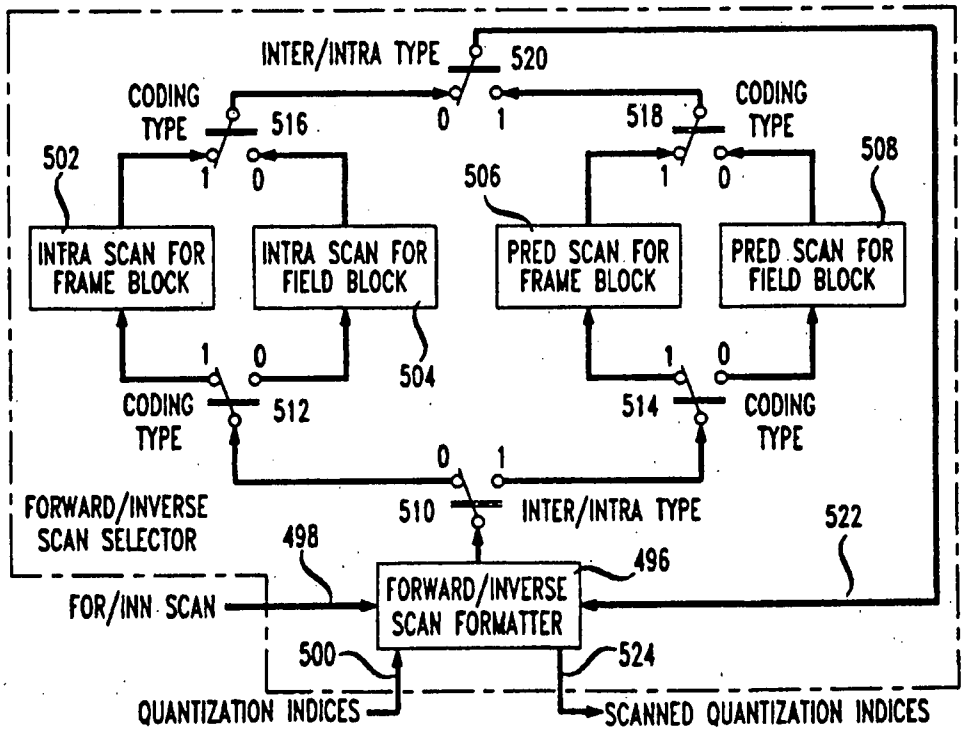

INTER/INTRA TYPE
520
CODING TYPE
516
518
CODING TYPE
502
506
508
INTRA SCAN FOR FRAME BLOCK
INTRA SCAN FOR FIELD BLOCK
PRED SCAN FOR FRAME BLOCK
PRED SCAN FOR FIELD BLOCK
CODING TYPE
512
504
514
CODING TYPE
FORWARD/INVERSE SCAN SELECTOR
510
INTER/INTRA TYPE
522
498
496
FOR/INN SCAN
FORWARD/INVERSE SCAN FORMATTER
500
524
QUANTIZATION INDICES
SCANNED QUANTIZATION INDICES

FIG. 15

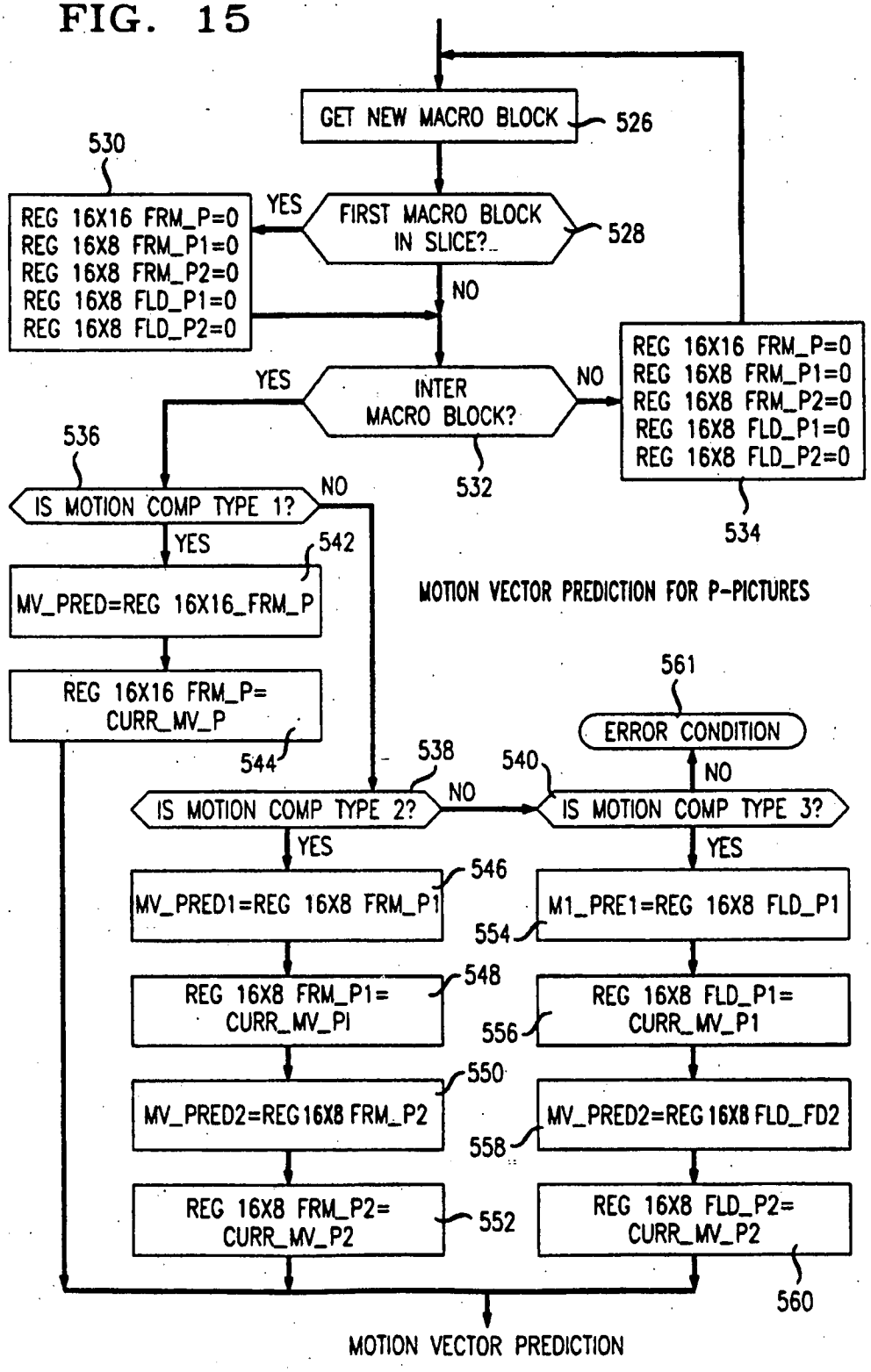
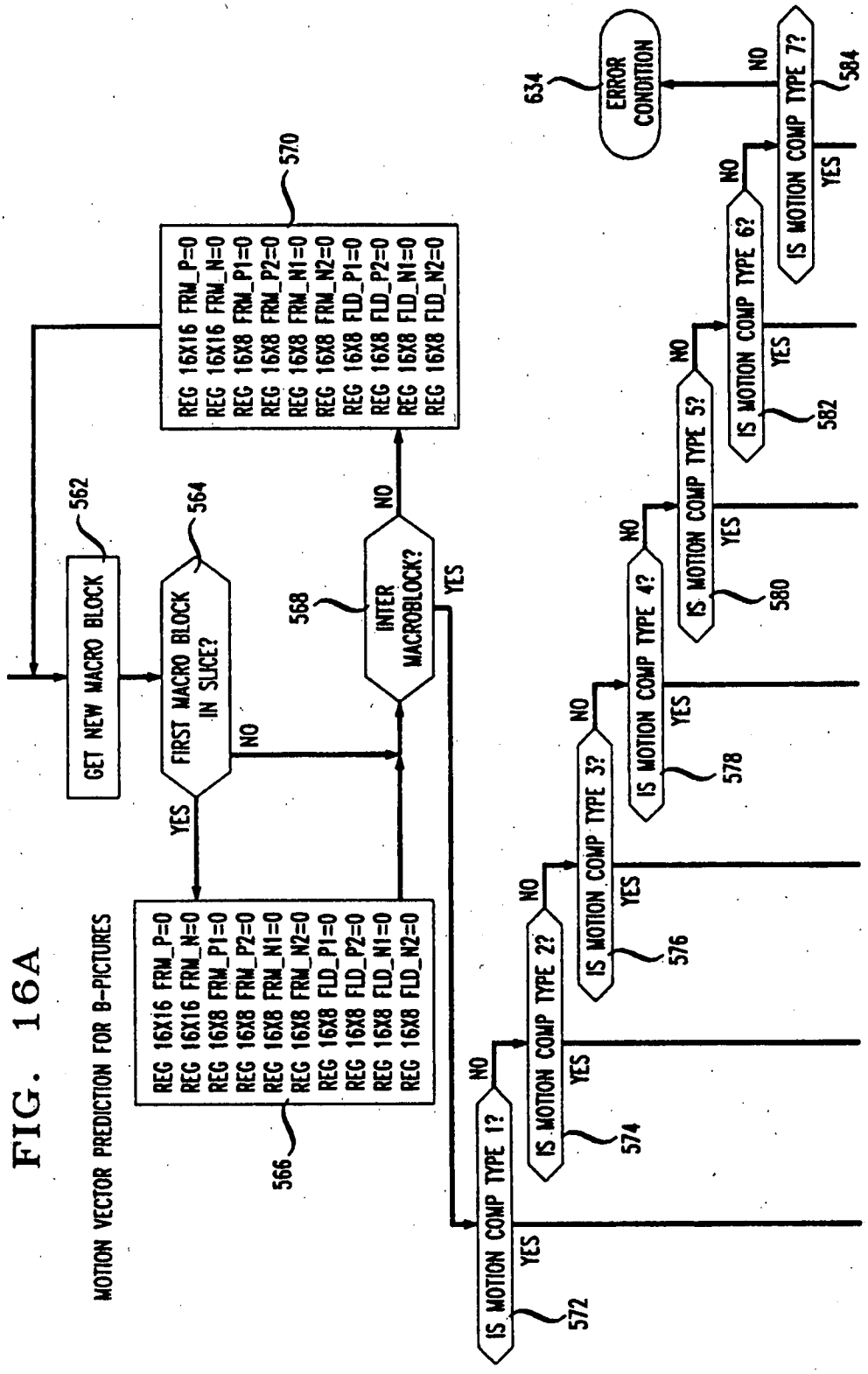

GET NEW MACRO BLOCK
526
530
REG 16X16 FRM_P=0
REG 16X8 FRM_P1=0
REG 16X8 FRM_P2=0
REG 16X8 FLD_P1=0
REG 16X8 FLD_P2=0
YES
FIRST MACRO BLOCK IN SLICE?
528
NO
YES
INTER MACRO BLOCK?
NO
532
REG 16X16 FRM_P=0
REG 16X8 FRM_P1=0
REG 16X8 FRM_P2=0
REG 16X8 FLD_P1=0
REG 16X8 FLD_P2=0
534
536
IS MOTION COMP TYPE 1?
NO
YES
542
MV_PRED=REG 16X16_FRM_P
REG 16X16 FRM_P= CURR_MV_P
544
MOTION VECTOR PREDICTION FOR P-PICTURES
561
ERROR CONDITION
538
IS MOTION COMP TYPE 2?
NO
540
IS MOTION COMP TYPE 3?
NO
YES
YES
546
MV_PRED1=REG 16X8 FRM_P1
M1_PRE1=REG 16X8 FLD_P1
554
548
REG 16X8 FRM_P1= CURR_MV_P1
REG 16X8 FLD_P1= CURR_MV_P1
556
550
MV_PRED2=REG 16X8 FRM_P2
MV_PRED2=REG 16X8 FLD_FD2
558
REG 16X8 FRM_P2= CURR_MV_P2
552
REG 16X8 FLD_P2= CURR_MV_P2
560
MOTION VECTOR PREDICTION

FIG. 16A


MOTION VECTOR PREDICTION FOR B-PICTURES
GET NEW MACRO BLOCK
562
FIRST MACRO BLOCK IN SLICE?
564
YES
NO
REG 16X16 FRM_P=0
REG 16X16 FRM_N=0
REG 16X8 FRM_P1=0
REG 16X8 FRM_P2=0
REG 16X8 FRM_N1=0
REG 16X8 FRM_N2=0
REG 16X8 FLD_P1=0
REG 16X8 FLD_P2=0
REG 16X8 FLD_N1=0
REG 16X8 FLD_N2=0
566
INTER MACROBLOCK?
568
NO
YES
REG 16X16 FRM_P=0
REG 16X16 FRM_N=0
REG 16X8 FRM_P1=0
REG 16X8 FRM_P2=0
REG 16X8 FRM_N1=0
REG 16X8 FRM_N2=0
REG 16X8 FLD_P1=0
REG 16X8 FLD_P2=0
REG 16X8 FLD_N1=0
REG 16X8 FLD_N2=0
570
IS MOTION COMP TYPE 1?
572
YES
NO
IS MOTION COMP TYPE 2?
574
YES
NO
IS MOTION COMP TYPE 3?
576
YES
NO
IS MOTION COMP TYPE 4?
578
YES
NO
IS MOTION COMP TYPE 5?
580
YES
NO
IS MOTION COMP TYPE 6?
582
YES
NO
IS MOTION COMP TYPE 7?
584
YES
NO
ERROR CONDITION
634

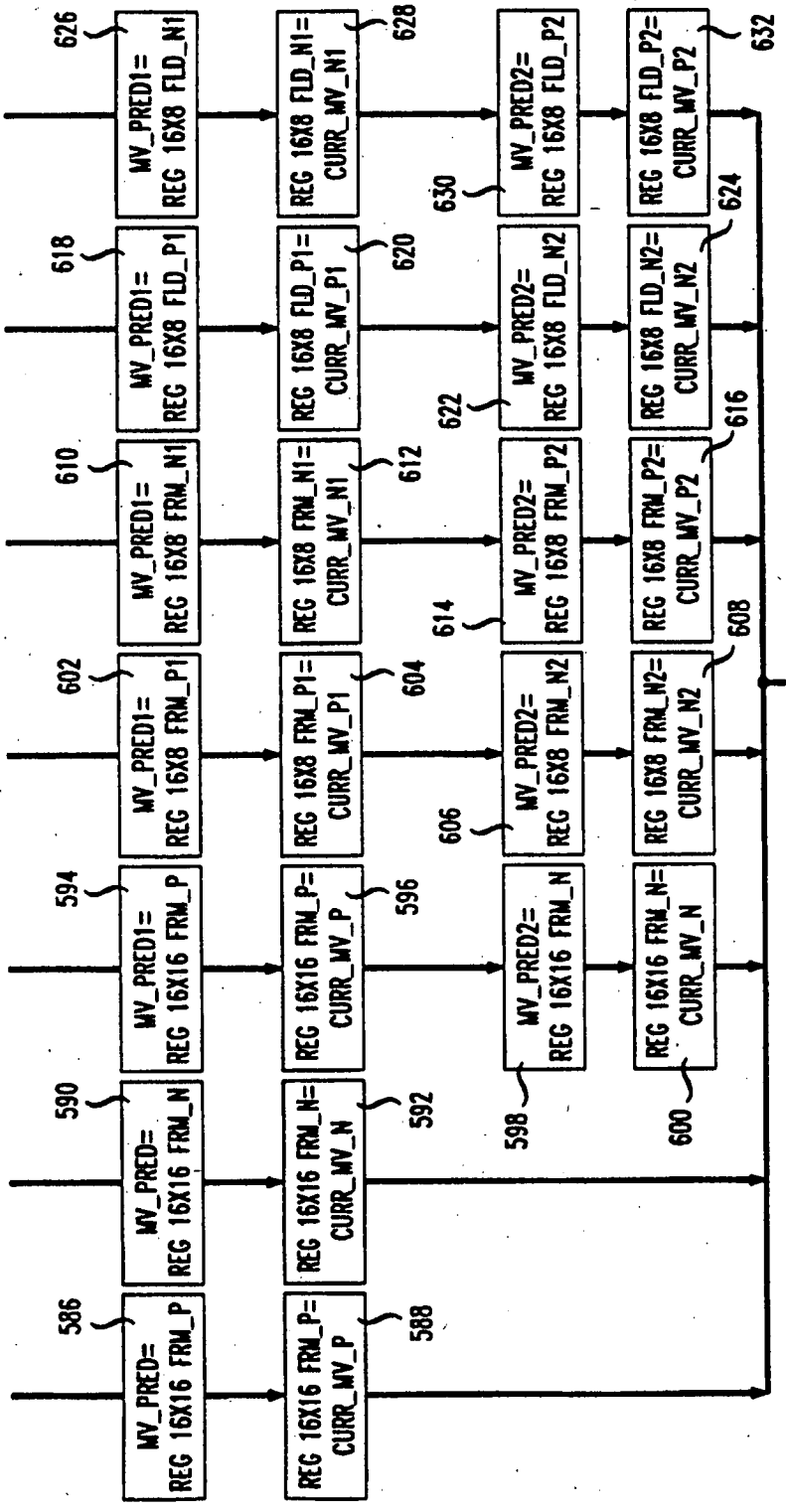


FIG. 16B

FIG. 16C

FIG. 16A
FIG. 16B

ADAPTIVE CODING AND DECODING OF FRAMES AND FIELDS OF VIDEO

TECHNICAL FIELD

This invention relates to coding and decoding of video signals. More particularly, this invention relates to adaptive encoders and decoders involved in the transmission and reception of digital video signals.

BACKGROUND OF THE INVENTION

Worldwide efforts are underway to improve the quality of video signal production, transmission, and reproduction because a great deal of commercial importance is being predicted for improved quality video systems. These efforts involve, at least in part, increasing the resolution with which are converted into representative electrical signals by increasing the spatial and temporal sampling rates that are used to convert video images into electrical signals. This increase in resolution consequently means that more data about images must be produced, processed, and transmitted in a given period of time.

Video images such as those images in the field of view of a television camera are scanned at a predetermined rate and converted into a series of electrical signals, each electrical signal representing a characteristic of a predetermined region of the image generally referred to as a picture element, pel, or pixel. A plurality of the picture elements taken together at a predetermined instant of time form what amounts to a still picture representing the nature of the image at the predetermined instant of time. Increasing the quality of video signals produced in this manner involves, at least in part, the use of larger number of smaller-size picture elements to represent a given image frame and the production of a large number of image frames per unit time. For example, the CCIR-601 recommendation specifies the number of picture elements in a frame to be 720 horizontal picture elements×486 vertical picture elements (U.S. and Japan) or 576 vertical picture elements (Europe). Thirty or 25 interlaced pictures are produced each second. In high definition television (HDTV) projects, it has been proposed to have about 700–1000 horizontal lines each having 1200–2000 picture elements. These HDTV efforts contemplate production of 25 or 30 interlaced pictures per second or 60 or 50 non-interlaced pictures per second.

As the number of picture elements for each video frame and the rate at which frames are produced increases, these is an increasing amount of video data which must be produced, transmitted, and received in a given period of time. It would be advantageous if video signals produced by these systems could be compressed so that a smaller amount of data could be generated which would still contain enough information so that higher quality video images could be reproduced.

A number of data compression schemes have been proposed which attempt to transmit higher quality video images using the same numbers of bits and the same bit rates used for lower quality images. One such scheme involves an encoder which receives digital video signals representing the characteristics of a sequence of picture elements. The encoder transforms blocks of such video signals into blocks of transform coefficients relating to the spatial frequency components in the areas of the image represented by the blocks of picture elements. The blocks of frequency coefficients are then quantized and scanned according to some predetermined sequence. The quantized frequency coefficients are then sent in the order defined by the scanning sequence to a variable word length coder then encodes the quantized frequency coefficients and then transmits the encoded quantized frequency coefficients. It has been found that less bits need to be sent when these encoded quantized frequency coefficients are sent instead of pixel data bits.

Another data compression scheme which has been proposed involves estimating the characteristics of a segment of video signal and subtracting the estimate from the actual segment of video signal to produce an estimate error signal which is then coded and transmitted instead of the actual segment of video signal. Again, it has been found that a lesser number of bits need to be transmitted when estimation error signals are transmitted in place of pixel data signals.

Yet another technique of compressing video data involves the production and transmission of data representing motion vectors calculated in light of a current segment of video signal and a prior segment of video signal instead of transmission of pixel data. These motion vectors may be used to provide motion compensation for producing more accurate estimates of a video signal and smaller estimate error signals, which thereby reduces the number of bits which must be used to transmit video signals.

Each of these techniques seeks to send data derived from an actual video signal which can be used by a decoder in a receiver of the video signals to reconstruct the actual video signal from a limited subset of the data defined the actual video signal. The actual number of bits which must be transmitted in these situations is less than the number of bits needed to define each picture element in the video signal and, thus, higher resolution video signals may be transmitted at the same bit rate. Although each of these techniques is to a certain degree successful in achieving suitable compression of video data without undue loss of information, there are significant areas whereby the encoding of video data may be improved so that the number of bits which must be transmitted is reduced and an accurate reconstruction may be made by a video decoder.

SUMMARY OF THE INVENTION

Improved compression of video data is achieved by an adaptive video frame/field encoder and decoder. In one example of the invention, different size blocks of video data are processed to achieve different modes of coding and different modes of motion estimation. The behavior of an encoder in accordance with one example of the invention and the behavior of a corresponding decoder adapt to certain characteristics of the video image. This adaptive behavior involves changing between a process of coding and decoding information from a frame of video or coding and decoding information from a field of video. In a more specific example, this invention involves an adaptation between coding and decoding information from a frame of video data or coding and decoding information from each of a plurality of interlaced fields of video data. Depending upon the coding mode used, certain steps may be taken, either individually or in any combination, to improve the compression and decompression of video data. In specific examples, appropriate quantization is chosen, different scanning techniques are adopted, different techniques of

predicting certain components of the video signals are used, or different motion compensation modes are employed. The benefits of this invention are useful for all video systems involving digital video signals, including high definition television and video telecommunication systems.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram representing a video signal encoder in accordance with this invention.

FIG. **1***a* is an illustration of a group of pictures structure useful in one example of the invention.

FIGS. **1***b* and **1***c* illustrate examples of block scans useful in one example of the invention.

FIGS. **1***d* to **1***k* illustrate examples of tables identifying the lengths of variable length codes used in one example of the invention.

FIG. **2** is a block diagram representing a video signal decoder in accordance with this invention.

FIG. **3** is a block diagram representing the functional specification of a block adaptive motion compensated unidirectional predictor in accordance with this invention.

FIG. **4** is a block diagram representing the functional specification of a block adaptive motion compensated bidirectional predictor in accordance with this invention.

FIG. **5** is a detailed block diagram of the block adaptive frame/field coding analyzer shown in FIG. **1**.

FIG. **6** is a block diagram of the block formatter of FIG. **1**.

FIG. **7** is a block diagram of the block unformatter of FIGS. **1** and **2**.

FIG. **8** is a flow chart illustrating intra DC coefficient prediction of FIGS. **1** and **2**.

FIG. **9** is an example of adaptive frame/field DC coefficient prediction.

FIG. **10** is a block diagram of the variable word length choice analyzer of FIG. **1**.

FIG. **11** is a flow chart illustrating the encoding of an mscale parameter for B-pictures in one example of the invention.

FIG. **12** is a flow chart illustrating the decoding of an mscale parameter for B-pictures in one example of the invention.

FIG. **13** is a block diagram of an example of a visibility matrix selector useful in one example of the invention.

FIG. **14** is a block diagram of a forward/inverse scan selector useful in one example of the invention.

FIG. **15** is a flow chart illustrating motion vector prediction for P-pictures in accordance with one example of the invention.

FIG. **16** is a flow chart illustrating motion vector prediction for B-pictures in accordance with one example of the invention.

DETAILED DESCRIPTION

FIG. **1** shows an adaptive motion compensated predictive/interpolative encoder in accordance with one example of this invention. The encoder of FIG. **1** receives digital video input signals on an input line **10** and compresses those video input signals for transmission to a receiver which decompresses those signals to produce video images. The digital video input signals are spatial and temporal samples of a video image and may be produced by scanning an image field and producing an electrical signal relating to the characteristics of the image field at predetermined points. The characteristics determined in the scanning operation are converted into electrical signals and digitized. The video input signals comprise a succession of digital words, each representing some information at a particular instant of time about a small region of the image field generally referred to as a picture element. A complete set of digital representations for the image at a particular instant of time is called a frame or a picture. Each frame may be considered to be composed of a number of smaller regions generally known as fields; for example, each frame may be composed of two interlaced fields representing odd- and even-numbered horizontal lines or rows of picture elements in the image. The frame may also be considered to represent a number of macroblocks, submacroblocks, and blocks of picture elements which are groups of contiguous picture elements, for example, 16×16 macroblocks of picture elements, 16×8 sublocks of picture elements, and 8×8 blocks of picture elements.

The digital input video signal may be a monochrome video signal or a color video signal. In the case of a monochrome video signal, each frame may comprise a set of digital representations of the brightness or intensity of a two-dimensional array of picture elements which make up a video image. In the case of a color video signal, each picture comprises not only a brightness component but also a color component. For example, in the CCIR 601 recommendation, a color video signal picture (i.e., a temporal sample of the image) may be composed of a luminance frame of 720 horizontal picture elements×480 vertical picture elements and two chrominance frames Cb and Cr at ½ resolution of 360 horizontal picture elements×240 vertical picture elements each. A sequence of such pictures may be transmitted at a rate of 29.97 pictures per second. The luminance frame is formed as the interlaced union of the two constituent CCIR-601 luminance fields, while the chrominance frames are derived by filtering and subsampling the respective 4:2:2 CCIR-601 chrominance frames.

For the purpose of illustrating a specific example of the invention, the description below will assume that the video signal on input line **10** is a video signal in accordance with the CCIR 601 recommendation. Those skilled in the art will appreciate that the principles of the invention are applicable to other types of video signals, such as HDTV video signals. To assist in the description of the example of the invention shown in FIG. **1**, some terminology should be defined. A block is an 8-horizontal-row by 8-vertical-column array of contiguous picture elements. Blocks may be groups of luminance data or groups of chrominance data. A macroblock is composed of four contiguous 8×8 luminance data blocks and the two 8×8 chrominance data blocks corresponding to the area of the image represented by the four luminance data blocks. A slice is one horizontal row of macroblocks starting at the left edge of the picture and ending at the right end of the picture. A luminance frame is formed as an interlaced union of two CCIR 601 luminance fields. One field comprises even-numbered horizontal rows of picture elements and the other field comprises odd-numbered horizontal rows of picture elements.

In the examples of the invention shown in FIGS. **1** and **2**, a plurality of picture types are encoded and decoded. Specifically, I-pictures, P-pictures, and B-pictures are encoded and decoded. I-pictures or intra-

coded pictures are pictures which are coded and decoded without reference to any other pictures. P-pictures, or predicted pictures are pictures which are coded in light of a previous picture. Motion compensation may be used to produce P-pictures. B-pictures, or bidirectionally predicted pictures are pictures which are coded in light of characteristics of a previous I- or P-picture and future I- or P-picture. As in the case of P-pictures, B-pictures may also be coded by using motion compensation. In appropriate circumstances, P-pictures and I-pictures may have some of their blocks coded in the same fashion that the blocks of the I-pictures are coded, i.e., without reference to other pictures ("intra coding"). A Group-of-Picture (GOP) structure is used in this example with N=12 and M=3 in Motion Picture Experts Group (MPEG) parlance. This GOP consists of one intra-coded I-picture, 3 predictively coded P-pictures, and 8 bidirectionally predictive code B-pictures. This GOP ensures that a complete I-picture occurs every 12/29.97 (approximately 0.4) seconds, which is hence the maximum delay for acquiring a picture from a bit stream. See FIG. 1*a*.

A picture is divided into macroblocks where a macroblock is a 16×16 luminance block plus the co-sited 8×8 Cb- and Cr-blocks (one of each). This definition, however, is easily extended to full CCIR-601 vertical chrominance resolution, where a 16×16 luminance block is associated with two 8×8 Cb and two 8×8 Cr blocks. The macroblock is the unit for which motion-compensation and quantization modes are defined. A slice is defined to be one row of macroblocks, starting at the left edge of the picture and ending at the right edge.

The digital video input signal on input line **10** is encoded into a compressed bit stream by the encoder shown in FIG. **1** which may be transmitted on output line **26** to another location having a receiver which may decode the bit stream and produce a video image. One important feature of the encoder of FIG. **1** is that a variety of coding techniques may be employed in appropriate circumstances to efficiently compress the video input signal on line **10** and to permit accurate reconstruction of the video image in the decoder without significant loss of information. Specifically, the encoder of FIG. **1** adaptively selects its coding operation so that either coding of frames in the video signal or coding of the interlaced fields in the frames of the video input signal takes place. Once the type of coding to be used for the video input signal has been selected, a variety of techniques used to compress video data may be improved in an adaptive fashion in light of the coding technique adopted to the encoder of FIG. **1**. For example, techniques of estimating future video signals may be made more accurate. Techniques of using motion compensation with these estimation techniques are also improved. In addition, items such as quantization procedures, scanning techniques, DC coefficient prediction, and variable word length coding may also be improved.

Two basic quantization and coding modes are allowed for a macroblock: frame coding and field coding. These quantization and coding modes are completely independent of the motion-compensation modes. In frame coding, four 8×8 luminance subblocks are formed from a macroblock. In field coding, four 8×8 luminance subblocks are derived from a macroblock by separating the lines of the two fields, such that each subblock contains only lines of one field. Frame coding is superior to field coding where there is not much motion between the two fields, and field coding is superior when there are detailed moving areas. The mode decision is made in the pixel domain once for the entire macroblock. An 8×8 DCT is then applied to each frame subblock or field subblock, depending on the mode selected.

The digital video input signal on input line **10** in FIG. **1** is directed to the noninverting input of a summing element **11**. An inverting input of the scanning element **11** receives a signal on line **12** related to an estimate of the video input signal on line **10**. The estimate for P-pictures is based upon a prediction made in light of past I- and P-pictures. The estimate for B-pictures is based upon a prediction made in light of past and future I- and P-pictures. No estimate is made for I-pictures and intra-coded portions of P- and B-pictures, so that the estimate signal on line **12** in these situations are zero, as indicated symbolically in FIG. **1** by the opening of an inter/intra type switching element **13***b* in series with line **12**. The summing element **11** produces an output signal on line **13** which is related to the error between the digital video input signal on line **10** and the estimate signal on line **12**. The estimate error signal on line **13** is directed to the input of a block adaptive frame/field coding analyzer **14**. The coding analyzer **14** examines predetermined characteristics of the video input signals on line **10** or of the estimate error signal on line **13**, depending on the state of a switching element **13***a* and makes a decision as to the type of coding to be used by the encoder of FIG. **1**. The analyzer **14** decides, when the switching element **13***a* connects line **13** to the input of the analyzer **14**, whether or not it would be advantageous to either code the frames of the estimate error signal on line **13** or to code the interlaced field of that estimate error signal. When the switching element **13***a* connects the input signal on line **10** to the input of the analyzer **14**, the analyzer decides whether or not it would be advantageous to either code the frames of the input signal on line **10** or to code the interlaced fields of that input signal on line **13**. The nature of analyzer **14**'s decision is indicated by the production of a coding type signal on line **15**. In a specific example of the invention using a video input signal produced by an interlace scanning technique, the selector **14** checks to see whether or not there are similarities in adjacent or alternating horizontal scan lines in the input signal or the estimate error signal. If the selector finds that the differences between adjacent scan lines are less than the differences between alternate scan lines, then the selector **14** produces a coding type signal on line **15** which indicates that frames of video information in the estimate error signal or the input signal are to be coded by the encoder of FIG. **1**. If the selector **14** finds that the differences between adjacent scan lines are greater than the differences between alternate odd and even scan lines, then the selector **14** produces a coding type signal on line **15** which indicates that each field of oddly numbered scan lines and each field of evenly numbered scan lines are to be coded separately.

The input signal on line **10** or the estimate error signal on line **13** is selectively directed to the input of a block formatting circuit **15***a*, depending on the state of the switching element **13***a*. The formatting circuit **15***a* is also responsive to the coding type signal on line **15** to direct the signals on either line **10** or line **13** in proper order on a line **17** to the input of a discrete cosine transform circuit **16**. When field coding has been selected by selector **14**, the order in which the data comprising the input signal on line **10** or the estimate error signal on

line 13 is changed so that first oddly numbered scan lines are consecutively directed to the input of a discrete cosine transform circuit 16 on the input line 17 followed by consecutive evenly numbered scan lines, or vice versa. The discrete cosine transform circuit 16 then converts each submacroblock of either evenly numbered or oddly numbered scan lines to a matrix of transform coefficients representing, in this example of the invention, spatial frequency components in the portion of the image represented by each submacroblock.

When frame coding has been chosen by the selector 14, each macroblock is sent to the discrete cosine transform circuit 16 on line 17 in the order they were received at the input of selector 14 on line 13. The discrete cosine transform circuit 16 then converts each block in the macroblock into a similarly sized matrix of transform coefficients representing, in this example of the invention, spatial frequency components in the portion of the image represented by each macroblock.

In addition to frame coding of entire macroblocks and field coding of submacroblocks representing oddly numbered and evenly numbered scan lines, the selector 14 may also be configured so that it may direct the encoder of FIG. 1 to code other kinds of submacroblocks such as groups of contiguous blocks which are smaller than the size of a macroblock.

The transform coefficients produced by the discrete cosine transform circuit 16 are directed on output line 18 to the input of a visibility matrix selector and perceptual quantizer 19. The quantizer 19 divides each of the transformed coefficients from discrete transform circuit 16 by predetermined scaling factors in one of a plurality of visibility matrices and a quantization parameter determined in light of the characteristics of the digital input signal communicated to the quantizer 19 on line 20 and in light of the characteristics of the estimate error signal communicated on the quantizer 19 on line 21. The amount by which the transform coefficients are quantized is also determined by the coding type signal produced by the selector 14 on line 15. The coding type signal is used by the quantizer 19 to adjust the quantization levels applied to the transform coefficients from discrete cosine transform circuit to improve the compression of video signals produced by the operation of the quantizer 19.

For AC-coefficient quantization, a 5-bit quantization parameter and a set of quantizer matrices are employed. Quantization is performed with a dead-zone for non-intra coding and without one for intra coding. This example of the invention allows four different quantizer matrices: one for each of the combinations of intra/-nonintra- and frame/field-coded macroblocks. There are no default matrices specified; the ones used are loaded and transmitted at the sequence layer. The Cb- and Cr-subblocks use the same matrices as the luminance subblocks.

In I-pictures, all macroblocks are coded, and the 5-bit quantization parameter is transmitted for every macroblock. In P- and B-pictures, some macroblocks may contain no coded coefficient data. A one-bit flag is sent for each macroblock to signal whether the macroblock is coded or not. In P-pictures, the quantization parameter is then transmitted for every coded macroblock.

In B-pictures, a 5-bit quantization parameter is transmitted at the start of every slice. A two-bit index (denoted mscale addr) is transmitted for every coded macroblock in the slice that identifies one of four multipliers (all of which are transmitted at the sequence layer). The slice quantization parameter is multiplied by the selected multiplier (denoted mscale) and the product rounded to the nearest integer and limited to 5 bits. The resulting number becomes the quantization parameter for that macroblock.

A coded block pattern framework (for signalling which of the subblocks inside a macroblock contain coded data) is used only with B-pictures.

Once the AC coefficients are quantized, they are coded for transmission. A scanning matrix ("scan") defines the order in which they are processed for encoding. Two fixed scans are defined: one for use in the frame-coding mode and the other for use in the field-coding mode. These scans do not change with the picture type in this example of the invention. See FIG. 1*b* and 1*c*.

Run-length and level combination are VL-coded for non-zero quantized AC coefficients. For each macroblock in I- and P-pictures, the encoder is allowed to choose one codebook out of a small number of codebooks. In this example of the invention, four codebooks for I-pictures and four for P-pictures are used. These eight codebooks are derived basically by permuting a set of codewords. Among other things, they differ in the length of the end of block (EOB) codeword (2, 3 or 4 bits). The lengths of the codewords in the top left corner of each codebook are shown in FIGS. 1*d*-1*k*.

For a particular macroblock in an I- or P-picture, the codebook yielding the smallest bit-count is selected, and signalled to the decoder with a 2-bit identifier. In B-pictures, this overhead for codebook selection was found to be excessive; therefore, one fixed codebook is used for all macroblocks in B-pictures. This codebook is one of the four used with P-pictures, and is the one shown in FIG. 1*b*.

In FIG. 1, the quantized transform coefficients are scanned by a scan selector circuit 23 in a predetermined order and sent to an encoder and multiplexer 24 which may select between a fixed word length codings of the transform coefficients or one or more variable word length coding of the transform coefficients. The scan analyzer 23 is responsive to the coding type signal produced by the selector 14 on line 15. The scan selector 23 determines one of a plurality of possible scanning orders which may be used to direct the transform coefficients to the encoder and multiplier 24. The order which the scanning selector 23 choses is based on what order may most conveniently be used by the encoder and multiplexer 24 to most efficiently code the transform coefficients with the fewest number of bits. For example, when frame coding has been used, the scan selector 23 may be configured to perform a zig zag scanning of the transform coefficients in the quantizer 19. When field coding has been used, the scan selector 23 may perform vertical scanning of the quantized transform coefficient in the quantizer 19. One of the strategies which may be employed in scanning the quantized transform coefficients in a strategy which will group like valued coefficients together for sequential transmission to the encoder and multiplexer 24. Examples of scanning sequences for frame block scans and field block scans are illustrated in FIGS. 1*a* and 1*b*, respectively. When like-valued coefficients are grouped together, a more efficient coding, such as a variable word length coding may be employed by the encoder 24. Longer word lengths may be used to represent higher valued coefficients while shorter word lengths may be used to represent coefficients having a value of zero or close to zero. An end of block (EOB)

code may be used to transmit pluralities of like valued coefficients.

As described in more detail below, the circuit of FIG. 1 includes a variable word length choice analyzer 23*a* which is responsive to the DCT coefficients output by the scan selector 23 and the picture type signal from line 32. The analyzer 23*a* produces a variable word length table select signal which is input to the encoder and multiplexer 24 and which is used by the encoder and multiplexer 24 to select one of a plurality of code tables which may be used to perform different kinds of fixed word length and variable word length coding of the DCT coefficients.

In addition to transmitting encoded transform coefficients, the encoder and multiplexer 24 receives and transmits along with the encoded coefficients a number of control signals which are used by a decoder in a video signal receiver to create a video image from the coded signals produced by the encoder in FIG. 1. These control signals include a signal related to the quantization parameter produced on line 30 between the quantizer 19 and the encoder and multiplexer 24. These control signals also include a picture type signal on line 32, which may be produced by an independently running sequencer not shown in FIG. 1. FIG. 1*a* shows an example of a sequence of I-, P-, and B-pictures which may be used. The picture type signal on line 32 represents what kind of picture is being produced by the encoder of FIG. 1, namely, either an I-, P-, or B-picture in this example of the invention.

The words produced by the encoder 24 are sent to a buffer 25 which then directs those words in an output bit stream on an encoder output line 26 at appropriate times. A fullness signal is produced on line 27 which is directed to an input of the quantizer 19 which thereby controls its operation to prevent the buffer 25 from overflowing or underflowing.

The production of an estimate of the digital video input signal is now described. The transform coefficients on the output of the scan selector 23 are connected to the input of an inverse scan selector 28 which rearranges the transform coefficients into the order they had in the quantizer 19 prior to being scanned by the scan selector 23. As shown in FIG. 1, the inverse scan selector 28 is also responsive to the coding type signal on line 15 which notifies the inverse scan selector 28 of the predetermined order into which the quantized transform coefficients were ordered by the scan selector 23 and thereby is the mechanism by which the inverse scan selector 28 is able to use the correct inverse scanning sequence. The transform coefficients as reordered by the inverse scan selector 28 are directed to a visibility matrix selector and dequantizer 29 which performs an inverse quantization procedure on the transform coefficients which basically reverses the operation of the quantizer 19. As shown in FIG. 1 the dequantizer 29 is responsive to the coding type signal on line 15 and the quantization parameter produced by the quantizer 19 to determine the correct dequantization procedure.

The output of the dequantizer 29 is directed to the input of an inverse discrete cosine transform circuit 34 which produces an output signal corresponding to the estimate error signal produced on line 13. The output signal from the inverse discrete cosine transform circuit 34 is directed on the noninverting input of a summing element 36 which receives at its inverting input a signal on line 38 which is related to an estimate of the video input signal on line 10. The output of the summing element 36 is directed to a next previous picture store 36*a* via a write next switching element 36*b* between the output of the summing element 36 and the input of the next picture store 36*a*. The output of the summing element 36 represents a frame of video data which has been coded by the encoder of FIG. 1. Writing a picture into the next picture store 36*a* causes a picture previously stored in the next picture store 36*a* to be written into a previous picture store 36*c* via closure of a write previous switching element 36*d*.

A motion estimation circuit 37 receives the digital video input signal from line 10, signals relating to the contents of the stores 36*a* and 36*c*, and the picture type signal from line 32. The motion estimation circuit 37 produces signals related to motion vectors which are used by an estimation circuit 38 which produces an estimate or prediction of the video input signal on line 10. An estimation circuit 38 is responsive to the contents of the stores 36*a* and 36*c* and the motion vectors produced by the motion estimation circuit 37 to produce a motion compensated estimate of the video input signal on line 10.

The motion compensated estimate produced by the estimation circuit 38 in this example of the invention, involving the previously described macroblock structure of video signals, takes into account the fact that a macroblock contains two interlaced fields of luminance pixels. Accordingly, two main categories of motion compensation are used by the estimation circuit 38, namely, a frame motion compensation mode and a field motion compensation mode. In the frame motion compensation mode, picture elements of an entire frame are predicted on a macroblock by macroblock basis from picture elements in reference frames. In the field compensation mode, the picture elements of one field are predicted only from pixels of reference fields corresponding to that one field. For example, pixels in a field of oddly numbered scan lines are predicted only from pixels in a reference field of oddly numbered scan lines.

In addition to having different motion compensation modes based on prediction of frames or fields of picture elements, there can be other modes of compensation based on the type of pictures being handled. In this example of the invention, the modes of compensation can also be based on whether P-pictures or B-pictures are being predicted. (No prediction is made for I-pictures). Examples of motion compensation types are summarized below.

A. Motion Compensation Modes for P-pictures

1. 16×16 frame motion compensation mode (type 1)

In this mode, the 16×16 luminance block is compensated by another 16×16 block from a reference frame which is fetched using one forward motion vector. No distinction is made between the picture element scan lines of the interlaced fields. The prediction block contains picture elements of both fields of the reference frame.

2. 16×8 frame motion compensation mode (type 2)

In this mode, the 16×16 luminance block is divided by a horizontal line of demarcation into a top 16×8 subblock and a bottom 16×8 subblock. Each subblock is independently compensated using a forward motion vector. Again, no distinction is made between the scan lines of the two interlaced fields making up the luminance subblocks. Two motion vectors are created and transmitted for each macroblock in this mode.

3. 16×8 field motion compensation mode (type 3)

In this mode, the 16×16 luminance block is separated by field polarity, namely, by oddly numbered and evenly numbered scan lines, into two 16×8 subblocks. Each of the 16×8 subblocks contains only picture elements lines of one of the interlaced fields in the original 16×16 luminance block. Each field subblock is compensated independently using a separate forward motion vector with a 16×8 subblock that is derived from picture element scan lines of a field of the same polarity in the reference frame. Two motion vectors are created and transmitted for each macroblock in this mode.

B. Motion Compensation Modes For B-pictures

1. 16×16 bidirectional (P and N) frame motion compensation mode (type 3)

A forward (from a previous [P] frame) motion vector fetches a 16×16 block from a past reference frame and a backward (from a new [N] frame) motion vector fetches a 16×16 block from a future reference frame. The 16×16 blocks are averaged to yield a final prediction block.

2. 16×16 forward (P) unidirectional frame motion compensation mode (type 1)

This is a forward unidirectional predictional mode in which only one forward motion vector is used for each macroblock.

3. 16×16 backward (N) unidirectional frame motion compensation mode (type 2)

This is a backward unidirectional predictional mode in which only one backward motion vector is used for each macroblock.

4. 16×8 frame motion compensation mode (type 4); top-forward (P1) with bottom-backward (N2)

In this mode the 16×16 luminance block is divided by a horizontal line of demarcation into a 16×8 top subblock and a 16×8 bottom subblock. The top subblock is compensated using a forward motion vector which fetches a 16×8 block from a past reference frame. The bottom subblock is compensated using a backward motion vector which fetches a 16×8 block from a future reference frame. Two motion vectors are produced and transmitted for each macroblock in this mode.

5. 16×8 frame motion compensation mode (type 5); top-backward (N1) with bottom-forward (N2)

This mode is similar to the mode B.4. described above. The top subblock is compensated using a backward motion vector and the bottom subblock is compensated using a forward motion vector.

6. 16×8 field motion compensation mode (type 6); odd-forward (P1) with even-backward (N2)

In this mode the 16×16 luminance block is separated by field polarity into two 16×8 field subblocks. One of the field subblocks contains the oddly numbered picture element scan lines and the other field subblock contains the evenly numbered picture element scan lines. The 16×8 field subblock containing the oddly numbered field lines is compensated using a forward motion vector and another 16×8 subblock derived from only the oddly numbered scan lines of a past reference frame. Similarly, the 16×8 field subblock containing the evenly numbered field scan lines is compensated using a backward motion vector and a 16×8 subblock derived from only the evenly numbered field lines of a future reference frame. Two motion vectors are produced and transmitted per macroblock in this mode.

7. 16×8 field motion compensated mode (type 7); odd-backward (N1) with even-forward (P2)

This mode is similar to mode B.6. described above with the 16×8 subblock containing the oddly numbered field lines being compensated by using a backward motion vector and a subblock from a future reference frame and the 16×8 block containing the evenly numbered field lines compensated using a forward motion vector and a subblock from a past reference frame. Two motion vectors are produced and transmitted per macroblock in this mode.

The motion estimation circuit **37** thus may produce motion vectors needed to effectuate a variety of motion compensation modes. These motion vectors are transmitted to the estimation circuit **38** via a line **39** and are used by the estimation circuit to perform in an adaptive motion compensated prediction of the video input signal on line **10**.

The estimation circuit is composed of two main parts, a block adaptive frame/field uni/bidirectional motion compensated prediction analyzer **38***a* and a block adaptive frame/field uni/bidirectional motion compensated prediction selector **38***b*. The prediction analyzer **38***a* is responsive to the motion vectors produced by the motion estimation circuit **37**, the picture type signal from line **32**, and the contents of the next picture store **36***a* and the previous picture store **36***c*. The prediction analyzer **38***a* produces a motion compensation type signal on line **38***c* which identifies which one of the motion compensation odes, such as the compensation modes described here, is used to produce an estimation of the video input signal on line **10**. The prediction selector **38***b* takes appropriate ones of the motion vectors computed by the motion estimation circuit **37** and the motion compensation type signal on line **38***c* and computes an estimation of the video input signal on line **10**. The estimate is produced in light of appropriate ones of the frames stored in stores **36***a* and **36***c*.

Each motion vector component in encoded differently, with respect to a previously transmitted component. The motion vectors produced by the motion estimation circuit **37** are also transmitted on a line **40** to a motion vector prediction circuit **41** and to summing element **42**. The motion vector prediction circuit **41** also receives the picture type signal on line **32** and the motion compensation type signal on line **38***c*, which identifies the motion compensation mode being used by the estimation circuit **38**. The circuit **41** produces an output signal to the inverting input of the summing element **42** which is related to a predicted value of the motion vector produced by the motion estimation circuit **37**. The summing element **42** subtracts the motion vector estimate from the motion vector signal on line **40** and produces a differential motion vector signal which is directed to the input of the encoder and multiplexer **24**. The encoder and multiplexer **24** sends the differential motion vector signal to the buffer **25** for insertion into the output bit stream on line **26**. The differential motion vector signal will be used by a decoder to reconstruct a video image in accordance with the video input signal on line **10**.

The decoder and multiplexer **24** also receives a block classification signal on line **43** which is produce by a block type classification circuit **44** in response to the previously described picture type signal, coding type signal, and motion compensation type signal.

The block type classification circuit **44** also receives an inter/intra type signal on line **44***a* which identifies whether intercoding or intracoding is being used for the block of video being classified. The inter/intra type

signal is produced by an inter/intra analyzer circuit 44*b* which receives the video input signal on line 10 and the estimate error signal on line 13. The analyzer circuit 44*a* determines and compares the energies present in the input signal and the estimation signal and makes a decision as to whether inter or intra coding is to be used. Inter coding, namely, coding of the estimate error signal on line 13, is used for P- and B-pictures when the energy of the estimate error signal is less than the energy of input signal. Sometimes in P- and B-pictures, such as in the situation where there is a scene change, it would be advantageous to code in an intra fashion, namely, to code the input video signal on line 10 instead of the estimate error on line 17. This is the case when the energy in the video input signal on line 10 is less than the energy in the estimate error signal on line 13. When this condition is sensed by the analyzer circuit 44*b*, the inter/intra type signal indicates that intra coding is to be used. As shown in FIG. 1, the inter/intra type coding signal from analyzer 44*b* controls the states of switching element 13*a* and the inter/intra type switching element 13*b*. The inter/intra coding type signal from the analyzer 44*b* also is directed to the input of the quantizer 19 to control the quantizing operation in accordance with how the video data is being coded. The inter/intra coding type signal is also a part of the block classification signal sent to the encoder and multiplexer 24.

The block classification signal on line 43 is transmitted to the buffer 25 by the encoder and multiplexer for insertion into the output bit stream on line 26. The block classification signal is used by a decoder to create an image in accordance with the video input signal on line 10.

All macroblocks in I-pictures are intra coded; intra coding of macroblocks is also allowed in P- and B-pictures. In intra coded macroblocks (field or frame mode), the DC coefficient of each subblock is uniformly quantized to 255 levels. DC-prediction is then employed to reduce the number of bits needed to encode the DC coefficients. Two predictors are maintained for luminance DC-prediction to increase its efficiency in switching between the frame- and field-coding modes. For chroma DC-prediction, one predictor suffices for each color component. All DC predictors are reset to zero at the start of a slice and at a nonintra coded macroblocks.

A signal on line 47 related to the magnitudes of intra coded DC coefficients produced by the discrete cosine transform circuit 16 is directed to the input of an intra DC coefficient prediction circuit 45 and the non-inverting input of a summing element 46. The switch 48 in line 47 symbolically represents the direction of only the intra DC coefficients produced by the discrete cosine transform circuit 16 to the input of the intra DC coefficient prediction circuit 45 and the summing element 46. The prediction circuit 45 also receives the coding type signal produced by the coding analyzer 14. The prediction circuit 45 produces an output relating to a predicted value of the intra code DC coefficient and directs that output signal to the inverting input of the summing element 46. The noninverting input of the summing element 46 receives the actual intra DC coefficient. The summing element 46 subtracts the prediction from the DC coefficient to produce a differential DC coefficient prediction which is directed to the input of the encoder and multiplexer 24. The differential DC coefficient is directed by the encoder and multiplexer 24 to the buffer 25 for insertion into the bit stream on output line 26. A decoder uses the differential DC coefficient to construct an image in accordance with the video input signal 10.

FIG. 2 shows an adaptive motion compensated uni/-bi-directional predictive/interpolative decoder which may be used to decode the output bit stream produced by an encoder such as the one shown in FIG. 1. An input bit stream is received by the decoder of FIG. 2 on an input line 50. The input bit stream on line 50 is placed in a buffer 52 and then sent to a variable word length decoder and demultiplexer 54 which performs a decoding and demultiplexing operation which is the inverse of the encoding and multiplexing produced by the encoder and multiplexer 24 shown in FIG. 1. The decoder and demultiplexer 54 produces on an output line 56 the quantized discrete cosine transform coefficients of FIG. 1. An inverse scan selector 64 rearranges the order of the coefficients appearing on input line 56 into the same order that those coefficients had in the quantizer 19 of FIG. 1. The inverse scan selector 64 directs the coefficients it receives in inverse order to a visibility matrix selector and dequantizer 66. The dequantizer circuit 66 is responsive to a coding type signal on line 68, a picture type signal on line 70, and the quantization parameter on line 68*a* retrieved from the bit stream input to the decoder of FIG. 2 by the decoder and demultiplexer circuit 54 to perform a dequantization which is the inverse of the quantization performed by the quantizer 19 in FIG. 1.

The intra DC transform coefficients for I-pictures and intracoded portions of P- and B-pictures are produced at the output of a summing element 58 which receives the differential DC coefficient as decoded and demultiplexed by the decoder and demultiplexer 54. The summing element 58 also receives an intra DC coefficient prediction signal from a prediction circuit 60 which is also responsive to intra DC coefficient signals on line 57 and a coding type signal on line 68. The state of a switching element 66*a* is controlled to switch an input line of an inverse discrete cosine transform circuit 72 between the DCT coefficient signal on the output of the dequantizer 66 and the intra DC coefficient signal on line 57. The dequantized transform coefficients and the intra DC coefficients are directed through a switching element 66*a* to an inverse discrete cosine transfer circuit 72 which performs a transform operation which is the inverse of the transform operation produced by the discrete cosine transform circuit 16 in FIG. 1. The output of the inverse discrete cosine transform circuit 72 in FIG. 2 is a decoded version of either the video signal on input line 10 in the case of I-pictures and intra coded portions of P- and B-pictures or it is a decoded version of the estimate error signal on line 13 in FIG. 1. The output signal from the transform circuit 72 is directed to the input of a block formatting circuit 72*a* which performs an unformatting operation which is the inverse of the formatting operation performed by block 15*a* in FIG. 1. The output of the unformatting circuit 72*a* is directed to one input of a summing element 74. Another input of the summing element 74 receives an estimate signal on line 76. The output of the summing element 74 is related to the difference between the estimate error signal from the unformatter 72*a* and the estimate signal on line 76 and comprises a video output on line 78 which is analogous to the video input signal on line 10 in FIG. 1.

The decoder and demultiplexer **54** in FIG. **2** directs the block classification signal received from the encoder of FIG. **1** to a block type declassification circuit **80** via an input line **82**. The block type declassification circuit **80** produces a coding type signal on line **68**, a picture type signal on line **70**, a motion compensation type signal on line **88**, and an inter/intra type signal on line **88***a* which correspond to the coding type, picture type, motion compensation type, and the inter/intra type signals produced by the encoder of FIG. **1**.

The decoder and demultiplexer **54** also retrieves the differential motion vector signal from the bit stream sent by the encoder of FIG. **1** and directs the differential motion vector signal on a line **90** to the input of a summing element **92**. Another input to summing element **92** is received from a motion vector prediction circuit **94** along line **96**. The motion vector prediction circuit **94** is responsive to a motion compensation type signal on line **88**, a picture type signal on line **70**, and selected motion vectors on line **98***a* to produce a prediction of the motion vectors which is directed to one of the inputs to the summing element **92**.

The output of the summing element **92** is a motion vector signal on line **98** which corresponds to the motion vector signals produced by the motion estimator **37** and used in the encoding performed by the circuit of FIG. **1**. The motion vector signals are directed to an estimation circuit **100** on an input line **102**. A next picture store **100***a* in responsive to the video output signals on line **78** through selective opening and closing of a switching element **100***b* shown in FIG. **2** to store selected frames of the video output signal. Writing a new frame of information into store **100***a* causes a previous frame in store **100***a* to be written into previous picture store **100***c* via closure of a switching element **100***d*. The estimation circuit **100** is responsive to the contents of the stores **100***a* and **100***c*, the motion vectors on line **102**, the motion compensation type signal on line **88**, and a picture type signal on line **70** to produce an estimate on line **76** of the video output signal on line **78** in a manner analogous to the estimate produced by the estimation circuit **38** in FIG. **1**. As in FIG. **1**, switching element **100***e* in series with line **76** acts to disconnect the estimate signal from the input of summing element **74** when I-pictures are being decoded or when intra coded portions of P- and B-pictures are involved.

FIG. **3** shows a block diagram of a block adaptive motion compensated predictor **106** in accordance with this invention. A decoded picture is received on line **108**. The circuit of FIG. **3** produces predictions of P-pictures in accordance with the motion compensation modes identified above for P-pictures. The decoded picture may be either received from the output of the summing element **36** in FIG. **1** or the output of the summing element **74** in FIG. **2**. The decoded picture on line **108** is received in the previous picture store **36***c* or **100***c*. The decoded picture from the previous picture store is sent to the input of a switching element **110** which selectively directs the decoded picture to one of three output lines in accordance with the motion compensation type signal. If the motion compensation type is type A.**1**., the decoded picture is sent to a frame macroblock motion compensation predictor **112** which produces a motion compensated prediction signal on the output line **114** of the predictor **106**. If the motion compensation type is type A.**2**., the decoded picture is sent to the input of a submacroblock switching element **116**. The switching element **116** selectively directs appropriate portions of the decoded picture to one of a pair of frame submacroblock motion compensated prediction circuits **118** and **120**. The top half frame submacroblock is directed to the prediction circuits **118** and the bottom half frame submacroblock is directed to the prediction circuit **120**. The predictions produced by circuits **118** and **120** are sent to two inputs **122** and **124** of a frame submacroblock to macroblock formatter **126** which then directs the resulting formatted prediction to the output line **114** of the prediction circuit **106**.

If the motion compensation type is type A.**3**., the decoded picture is sent by the switching element **110** to the input of a submacroblock switching element **128** which selectively directs the top field submacroblock portion of the decoded picture to one of two field submacroblock motion compensated prediction circuits **130** and **132** and the bottom field submacroblock portion of the decoded picture to the other of the prediction circuits **130** and **132**. The prediction signals produced by the prediction circuits **130** and **132** are directed on lines **134** and **136** to the inputs of a field submacroblocks to macroblock formatting circuit **138** which then outputs the formatted predictions on the output line **114** of the prediction circuit **106**. The predictor circuits are responsive to either one or two motion vector signals (MV**1** or MV**1** and MV**2**) as described above and shown in FIG. **3**.

FIG. **4** is a block diagram of a block adaptive motion compensated bidirectional predictor **140** in accordance with this invention. The circuit of FIG. **4** produces predictions of B-pictures in accordance with the motion compensation modes identified above for B-pictures. A decoded previous picture is received on line **142** from either one of the outputs of summing element **36** in FIG. **1** or summing element **74** in FIG. **2**. The decoded previous picture is sent to the previous picture store **36***c* or **100***c* of FIGS. **1** and **2**, respectively. A decoded next picture is received on an input line **146** from the output of either summing element **36** in FIG. **1** or summing element **74** in FIG. **2**. The decoded next picture is directed to the next picture store **36***a* or **100***a* of FIGS. **1** and **2**, respectively. The decoded previous picture in the previous picture store is selectively directed by a switching element **150** on input line **149** to one of six output lines depending upon the motion compensation type signal in FIGS. **1** and **2**. The decoded next picture in the next picture store is selectively directed by the switching element **150** on input line **152** to one of a second set of six output lines, depending again on the value of the motion compensation type signal in FIGS. **1** and **2**.

If the motion compensation type is type B.**1**., the decoded previous picture in the previous picture store is directed on an output line **154** to the input of a frame macroblock motion compensation prediction circuit **156** which then outputs a frame macroblock motion compensated prediction on a line **158** and on an output line **160** of the prediction circuit **140**. Type B.**1**. motion compensation is undefined for the decoded next pictures in the next picture store **146** so that the decoded next pictures are not involved in the prediction operation and do not influence the prediction signal produced on output line **160** in type B.**1**. situations.

If the motion compensation type is type B.**2**., the decoded next picture in the next picture store is directed to an output line **156** of the switching element **150** toward the input of a frame macroblock motion compensated prediction circuit **162** which produces a frame

macroblock motion compensated prediction on an output line 164 and an output line 160 of the prediction circuit 140. Type B.2. motion compensation is undefined for the decoded previous pictures in the previous picture store so that the decoded previous pictures are not involved in the prediction operation and do not influence the prediction signal produced on output line 160 in type B.2. situations.

When the motion compensation type is type B.3., the decoded previous picture in the previous picture store is directed on output line 166 of the switching element 150 to one input of a frame macroblock bidirectional motion compensated prediction circuit 168. The decoded next picture in the next picture store is directed to an output line 170 of the switching element 150 to a second input of the prediction circuit 168. The prediction circuit 168 produces a prediction signal on line 172 which is directed to the output line 160 of the prediction circuit 140.

When the compensation type is type B.4., the decoded previous picture in the previous picture store is directed to an output line 174 of the switching element 150 and the decoded next picture in the next picture store 148 is directed to an output line 176 of the switching element 150. The pictures on line 174 and 176 are directed to the inputs of a switching element 178 which selectively directs the pictures to the inputs of a pair of frame submacroblock motion compensated prediction circuits 180 and 182. The outputs of the prediction circuits 180 and 182 are directed to the inputs of a frame submacroblocks to macroblock formatting circuit 184 on lines 186 and 188. The output of the formatting circuit 184 is directed to the output line 160 of the prediction circuit 140.

When the motion compensation type is type B.5., the decoded previous picture in the previous picture store is directed to an output line 190 of the switching element 150 and the decoded next picture in the next picture store is directed to an output line 192 of the switching element 150. The pictures on lines 190 and 192 are directed to the inputs of a switching element 194 which selectively directs the pictures to a pair of frame submacroblock motion compensated prediction circuits 196 and 198. Prediction signals on lines 200 and 202 from the prediction circuits 196 and 198 are input to a frame submacroblocks to macroblocks formatting circuit 204 which sends formatted prediction signals to the output line 160 of the prediction circuit 140.

When the motion compensation type is type B.6., the decoded previous picture in the previous picture store is directed to an output line 206 of the switching element 150 and the decoded next picture in the next picture store is sent to an output line 208 of the switching element 150. The pictures on lines 206 and 208 are sent to the inputs of a switching element 210 which selectively directs those pictures to the inputs of a pair of field submacroblock motion compensated prediction circuits 212 and 214. The prediction circuits 212 and 214 send prediction signals on lines 216 and 218 to the inputs of a field submacroblocks to macroblock formatting circuit 220. The formatting circuit 220 directs formatted prediction signals to the output line 160 of the prediction circuit 140.

When the motion compensation type is type B.7., the decoded previous picture in the previous picture store is sent to an output line 222 of the switching element 150 and the decoded next picture in the next picture store is sent to an output line 224 of the switching element 150. The pictures on lines 222 and 224 are sent to inputs of a switching element 226 which selectively directs the picture signals to a pair of field submacroblock motion compensated prediction circuits 228 and 230. The prediction circuits send prediction signals to a field submacroblocks to macroblocks formatting circuit 232 which directs the prediction signals to the output line 160 of the prediction circuit 140. As shown in FIG. 3, the predictor circuits are responsive to one or two motion vector signals (MV1 or MV1 and MV2).

FIG. 5 shows a detailed block diagram of the block adaptive frame/field coding analyzer 14 shown in FIG. 1. Macroblocks are received by the coding analyzer on an input line 234. The macroblocks received on the input line 234 are directed to a macroblock vertical correlation computer 236 which determines the amount of correlation between successive horizontal lines of each macroblock and produces a frame correlation signal on line 238 which is sent to a thresholder and comparator circuit 240. The macroblocks received on the input line 234 are also sent to a macroblock to field submacroblocks formatting circuit 242. The formatter 242 separates each macroblock into two separate fields, one of which comprises the evenly numbered horizontal lines of picture elements in each macroblock and the other or which comprises the oddly numbered horizontal lines of picture elements in each macroblock. The evenly numbered horizontal lines are sent to a submacroblock vertical correlation computer 244 when a selector switch 246 connects the output of the formatting circuit 242 to the input of the correlation computer 244. The correlation computer 244 determines the level of correlation between the evenly numbered horizontal lines sent to the computer 244 and produces a signal to a submacroblock selector switch 248 relating to the level of correlation. When the switch 248 is in an appropriate position, the signal from the correlation computer 244 relating to correlation level is sent to the input of an accumulator 250. The oddly numbered horizontal lines are sent to a submacroblock vertical correlation computer 252 when the selector switch 246 connects the output of the formatting circuit 242 to the input of the correlation computer 252. The correlation computer 252 determines the level of correlation between the oddly numbered horizontal lines sent to the computer 252 and produces a signal sent to the submacroblock selector switch 248 relating to the level of correlation. When the switch 248 is connected appropriately, the signal from the correlation computer 252 relating to the correlation level of the oddly numbered lines is sent to the accumulator 250. The accumulator 250 sums the correlation levels of the oddly numbered and evenly number fields in the macroblocks and produces a total correlation signal indicating the level of correlation in the fields. The total correlation signal is divided by two in a divider circuit 254 and sent to the threshold in comparator circuit 240. The comparator circuit 240 compares the correlation levels of the frame correlation signal from computer 236 and the field correlation signal from divider 254 and produces the coding type signal shown in FIGS. 1 and 2. The coding type signal indicates whether frames of video information are to be encoded and decoded or fields of video information are to be encoded and decoded. Specifically, frames are to be encoded and decoded when the frame correlation level is greater than the field correlation level. Fields are to be encoded and decoded when the field correlation level is greater than the frame correlation level.

FIG. 6 shows the block formatter 15a of FIG. 1. Macroblocks are received on an input line 256 and directed to the input of a switching element 258. The state of the switching element is determined by the characteristics of the coding type signal produced by the circuit of FIG. 5. When the coding type signal indicates that frame coding is to take place, the switching element 258 connects the line 256 to the input to a macroblock to frame blocks formatter 260, the output of which is directed to a switching element 262 which connects the output of the formatter 260 to a block output line 264 of the formatter of FIG. 6. When the coding type signal indicates that field coding is to take place, the switching element 258 connects the line 256 to the input of a macroblock to field blocks formatter 266, the output of which is directed to the switching element 262 which connects the output of the formatter 266 to the block output line 264 of the formatter of FIG. 6.

FIG. 7 is a diagram of a block unformatter shown both in FIGS. 1 and 2. Blocks are received by the unformatter of FIG. 7 on an input line 268 which directs those blocks to the input of a switching element 270. When the coding type signal indicates that frame coding is to take place, the switching element 270 connects the blocks received on the output line 268 to the input of a frame blocks to macroblock formatter 272 which directs frame macroblocks to the input of a switching element 274 which is configured by the coding type signal in this situation to direct frame blocks from the formatter 272 to a macroblock output line 276. When the coding type signal indicates that field coding is to take place, the switching element 270 connects the blocks received on the input line 268 to the input of a field blocks to macroblock formatter 278 which directs the field blocks to the input of a switching element 274 which is configured by the coding type signal to direct macroblocks from the formatter 278 to the macroblock output line 276.

FIG. 8 is a flow chart representing an intra DC coefficient prediction circuit as shown as in FIGS. 1 and 2. Actually computed intra DC coefficients are directed to the input of the circuit of FIG. 8 on line 47. A quantization index is computer in block 280 by dividing the DC coefficient present on line 47 by a DC step-size parameter which may be eight. A decision then is made at block 282 to determine whether or not this DC coefficient is the DC coefficient for the first macroblock in a slice or whether the previous macroblock was a nonintra macroblock. For purposes of this discussion, the block index identifies the location of a block in a macroblock as shown for a frame macroblock and a field macroblock in FIG. 9. If the decision of block 282 is yes, a decision then is made at block 284 as to whether the block index is zero. If the block index is zero, a DC top block predictor parameter is set equal to some arbitrary value in block 286, for example, the predictor is set to 128. The DC prediction which is made by the prediction circuit of FIG. 8 is set equal to this top block DC predictor in block 288. In block 290, the top block predictor is overwritten with the value of the quantization index computed in block 280. In block 292 a bottom block DC predictor parameter is set equal to the value of the top block DC predictor set in block 290. The DC prediction is output from the circuit of FIG. 8 on output line 294 which corresponds to the output of the intra DC coefficient predictor blocks in FIGS. 1 and 2.

If the block index is 1, 2, or 3, as determined in block 284 in FIG. 8, a check is made in block 296 to see if the block index is one. If the block index is one, the DC prediction is set equal to the top block DC predictor in block 298. The top block DC predictor then is overwritten with the quantization index in block 300 and the DC prediction is output by the circuit of FIG. 8 on line 294. If the block index, as determined in block 296, is 2 or 3, the DC prediction is set to the bottom block DC predictor in block 302 and the bottom block DC predictor then is overwritten with the value of the quantization index in block 304. The DC prediction signal is output from the circuit of FIG. 8 on line 294.

If the macroblock is not the first macroblock in a slice or the previous macroblock is not a nonintra type macroblock as determined in block 282 in FIG. 8, a decision is made in block 306 as to whether or not the current macroblock type is the same as the previous macroblock type. If the current macroblock is a frame type macroblock, block 306 in FIG. 8 determines if the previous macroblock was also a frame type macroblock. If the current macroblock is a field type macroblock, block 306 in FIG. 8 determines if the previous macroblock was also a field-type macroblock. If either of the determinations is in the affirmative, a check is made at block 308 to see if the block index is zero or one. If the block index is zero or one, the DC prediction is set equal to the top block DC predictor in block 310 and then in block 312 the top block DC predictor is set equal to the quantization index computed in block 280. The DC prediction then is output by the circuit of FIG. 8 on line 294. If the block index is 2 or 3, as determined in block 308, then the DC prediction is set equal to the bottom block predictor in block 314. The bottom block DC predictor then is overwritten with the quantization index in block 316 and the DC prediction is sent from the circuit of FIG. 8 on line 294.

If the current macroblock type is not the same as the previous macroblock type, as determined in block 306, then a check is made in block 318 to see if the block index is zero. If the block index is zero, an average of the current values of the top block DC predictor and the bottom block DC predictor is made in block 320. The average computed in block 320 is rounded in block 322. The top block DC predictor is set equal to the rounded average in block 324. The DC prediction then is set equal to the top block DC predictor in block 326. The top block DC predictor then is overwritten with the value of the quantization index in block 328. The bottom block DC predictor is set equal to the top block DC predictor in block 330. The DC prediction is sent to the output line 294 of the circuit of FIG. 8.

If the block index is 1, 2, or 3, as determined in block 318, a determination is made in block 332 to see if the block index is 1. If the block index is 1, the DC prediction is set to be equal to the top block DC predictor in block 334. The top block DC predictor then is overwritten in block 336 with the quantization index and the DC prediction is directed to the output line 294 of the circuit of FIG. 8. If the block index is not 1, as determined in block 332, then the DC prediction is set equal to the value of the bottom block DC predictor in block 338 and the bottom block DC predictor then is overwritten in block 340 with the value of the quantization index computed in block 280. As in the other cases, the DC prediction is directed to the output line 294.

FIG. 9 illustrates a sequence of six macroblocks in a slice of video data. The arrows in FIG. 9 illustrate schematically the relationship between the predicted DC coefficients for each of the blocks of video data and the

actual values of DC coefficients computer for adjacent blocks. The origin of each of the arrows is placed in the block for which an actual DC coefficient has been determined. The head of the arrow indicates the location of the block for which the actual value of the DC coefficient is used as a prediction of the DC coefficient for the block containing the head of the arrow. The circles in some of the blocks indicate the situation where an average of actual values of DC coefficients is used to predict the DC coefficients for the block in which the circles reside. FIG. 9 shows a frame type macroblock numbered 0 at the start of the macroblock slice, followed by a frame type macroblock numbered 1, a field type macroblock numbered 2, a frame type macroblock numbered 3, a field type macroblock numbered 4, and a field type macroblock numbered 5. Each of the macroblocks contains four blocks, the upper left hand block of each macroblock being given a block index of 0. The upper right hand block of each macroblock is given a block index of 1, the lower left hand block of each macroblock is given a block index of 2, and the lower right hand block of each macroblock is given a block index of 3. By way of example, a frame macroblock 0 and a field macroblock 4 are labeled with the block indices in FIG. 9. The block in frame macroblock 0 with an index of 0 is predicted to have a value of DC coefficient equal to an arbitrary value such as 128. The predicted value of the DC coefficient for the block in the frame macroblock 0 having an index of 1 is predicted to be the DC coefficient computed for the block having an index of 0 as illustrated by the arrow from the block having an index of 0 to the block having an index of 1. The predicted value of the DC coefficient for the block with an index of 2 in frame macroblock 0 is predicted to be the same as the actual value of the DC coefficient computed for the block having an index of 0 in the frame macroblock 0 as illustrated by the vertically downwardly directed arrow from the block of index 0 to the block of index 2. The predicted value of the DC coefficient for the block with an index of 3 in macroblock 0 is the same as the actual value of the DC coefficient computed for the block with an index of 2 in macroblock 0. The value of the DC coefficient predicted for the blocks of index 0 in the field macroblock 2, the frame macroblock 3, and the field macroblock 4 are each predicted to be the average of the computed coefficients for blocks of index 1 and 3 in the prior macroblocks in each instance. The rest of the predictions are apparent in light of the content of FIGS. 8 and 9 and are not further discussed.

FIG. 10 is a block diagram of the variable word length choice analyzer shown in FIG. 1. The analyzer receives the picture type signal on line 32 as an input to a signal translator 342 which produces two signals, an intracoding indication 344 which identifies whether or not there is intra-type coding and a B-type predictive indicator 346 which indicates whether or not predictive coding of B-pictures is being undertaken.

The variable word length choice analyzer also receives the DCT coefficients from the scan selector 23 in a run length computer 348 which determines, for all nonzero coefficients received, the number of zeros immediately prior to the receipt of that nonzero DCT coefficient. In other words, the computer 348 determines the run length for each DCT coefficient and sends signals relating to the amplitude and run length of each received DCT coefficient to the input of an intra indication switching element 350. The switching element 350 directs the signals from the run length computer 348, when it is in the number 1 state shown in FIG. 10, to the input of sequencing switching element 352, which may be a counting circuit. The switching element 350 is in the number 1 state when the choice analyzer of FIG. 10 is analyzing DCT coefficient for I-pictures. The switching element 352 sequentially directs the amplitude and run length of each coefficient to the inputs of each of a series of four variable word length code length tables 354, 356, 358, and 356. The tables 354, 356, 358, and 360 contain data which represents the expected code length which will be produced if the DCT coefficient having an amplitude and run length as represented by the signals from the run length computer 348 are coded in accordance with four different coding schemes, one for each of the tables 354, 356, 358, and 360. When the switching element 352 connects the output of the run length computer 348 to the input of one of the tables 354, 356, 358, and 360, that table produces a signal on a respective output line 362, 364, 366, and 368 relating to the expected code length which will result from coding the DCT coefficient in accordance with the coding scheme for which the data in the table has been produced. The expected code lengths on lines 362, 364, 366, and 368 are directed to the number 1 inputs of a switching element 370. When the DCT coefficients are for I-pictures, the state of the switching element 370 is such that the signals on lines 362, 364, 366, and 368 are directed to the inputs of accumulators 372, 374, 376, and 378, respectively. The contents of the accumulators 372, 374, 376, and 378 are directed to four inputs of a comparator and minimum evaluation circuit 380 which is responsive to a compare enable signal to determine which of the expected run lengths stored in the accumulators 372, 374, 376, and 378 is the smallest. The evaluation circuit 380 produces a variable word length table select signal on line 382 which is then sent to the encoder and multiplexer 24 in FIG. 1 which then uses the signal produced on line 382 to select an efficient fixed or variable word length coding scheme identified by the nature of the variable word length table select signal on line 382. The select signal on line 382 causes the encoder 24 to select one of a plurality of coding tables stored in the encoder 24 which each control the encoding of quantized DCT coefficients in accordance with a separate and distinct fixed word length or variable word length coding scheme.

When P- and B-pictures are being coded, the switching element 350 is placed in state 0 such that the DCT coefficient amplitudes and run lengths are directed to the input of a bidirectional prediction indication switching element 384. When P-pictures are being coded, the switching element 384 is in the state 0 which causes the coefficient amplitudes and run lengths to be connected to the input of a sequencing switching element 386, which may be a counting circuit. The switching element 386 sequentially connects the P-picture coefficient amplitude and run lengths to each of four variable word length code length tables 388, 390, 392, and 394. The tables 388, 390, 392, and 394 each contain information about expected code lengths which will be produced by coding the DCT coefficient having the computed amplitude and run lengths in accordance with separate and distinct codes represented by each table 388, 390, 392 and 394. The expected code lengths from the tables 388, 390, 392, and 394 are sent to the 0 inputs of the switching element 370. For P-pictures, the state of the switching elements 370 and 396 is such that the outputs of the tables 388, 390, 392, and 394 are directed on output lines

398, 400, 402, and 404 and to the inputs of the accumulators 372, 374, 376, and 378. As in the case of I-pictures described above, the accumulators 372, 374, 376, and 378 direct the expected code lengths to the evaluation circuit 380 which then produces a variable word length select signal on line 382 which is used by the encoder and multiplexer 24 to code the quantized DCT coefficients using a code which is most efficient in terms of the smallest number of bits which must be used to transmit the DCT coefficients.

When B-pictures are being analyzed by the circuit of FIG. 10, the DCT coefficient amplitudes and run lengths from the computer 348 are directed to the input of a switching element 384 which is in a state which will connect those amplitudes and run lengths only to the variable word length code length table 388 which produces an expected run length signal and sends it to the input of the accumulator 372 and is used to produce a variable word length table select signal on line 382. Note that in the case of B-pictures, the state of switching element 396 is such that the output of the tables 390, 392, and 394 are not connected the inputs of the accumulators 374, 376, and 378.

In the example of the invention described here, a parameter Mscale is produced. The M scale parameter is a multiplexer to be used in computing the quantization parameter for bidirectionally coded frames. One M scale parameter is produced for each macroblock by selection from predetermined tables which have been determined experimentally for a particular picture resolution and bit rate.

FIG. 11 is a flow chart illustrating the encoding of the M scale parameter for bidirectionally coded pictures. In block 406, a decision is made as to whether or not the current macroblock was coded in the previous P-picture or I-picture. If not, whether decision is made in Block 408 as to whether or not the average quantization parameter was nonzero for a corresponding slice in the previous P-picture or I-picture. If it was zero, the average quantization parameter is set equal to 16 in block 410. If the average quantization parameter was nonzero, as determined in block 408, or after the average quantization parameter has been set to 16 in block 410, the quantization parameter for the corresponding macroblock in the previous P-picture or I-picture is set equal to 16 in block 412, since there was no quantization parameter for that macroblock. If the macroblock was coded in the previous P-picture or I-picture, as determined in block 406, or after the operation of block 412, the quantization step size is set equal to 2 times the quantization parameter in block 414. The quantization step size for the slice is set equal to 2 times the average quantization parameter for the corresponding slice in the previous P-picture or I-picture in block 414. A decision then is made in block 416 as to whether or not the prediction for the macroblock is to be unidirectional or intra. If the prediction is to be bidirectional, in other words, if the no route is followed at the output of block 416, then a temporary scale factor is set equal to a scale factor tuned to the resolution and bit rate times the quantizer step size for the macroblock in the previous P-picture or I-picture in block 418. In block 420 a ratio parameter is computed by dividing the temporary scale factor by the quantization step size for the slice. In block 422 a variable min_ind is set equal to 9, a variable min_valf is set equal to 1,000, and a variable zlimit is set equal to 4. A decision then is made at block 424 to see whether an index z is less than zlimit. If z is less than zlimit, a variable absdiff is set equal to the absolute value of the difference between the ratio computed in block 420 and a mscale parameter in block 426. Then in block 428 a decision is made as to whether or not the value of absdiff is less than min_valf. If the yes route is followed from block 428, a variable min_ind is set equal to z and the variable min_valf is set equal to the variable absdiff in block 430. The routine of FIG. 11 then returns to the input of block 424. If the no route is followed on the output of block 428, the routine of FIG. 11 returns directly to the input of block 424. The loop just described is repeated until z is no longer less than zlimit and the no route is followed on the output of block 424, after which in block 432 the current quantization step size is set equal to a rounded product of an mscale variable shown in block 432 and the quantization stepsize for the slice. The current quantization parameter is set equal to one-half of the current quantization step size in block 434 and the value of min_ind is sent for insertion into the encoder output bit stream for use by the decoder to recreate images from the compressed bit stream. The current quantization parameter is clipped to a valve of 31 by the operation of blocks 436 and 438.

If it is determined in block 416 that there is unidirectional prediction, then a temporary scale factor is computed in block 440, a ratio of the temporary scale factor to the quantization step size for the slice is computed in block 442 and in block 444, the min_ind, min_valf, and the zlimit variables are set to the same values they were set to in block 422. A decision then is made in block 446 to see if an index z is less than zlimit. If that is the case, a variable absdiff is computed in block 448 as the absolute value of the difference between the ratio computed in block 442 and an mscale parameter identified in block 448. A check then is made in block 450 to see if absdiff is less than min_valf. If that is the case, min_ind is set equal to z and min_valf is set equal to absdiff in block 452. The routine of FIG. 11 then returns to the input of block 446. The loop continues until z is no longer less than zlimit and then the routine follows the no route as the output of block 446. When the no route is followed in block 454, the current quantization step parameter is set equal to a product similar to the product already described for block 432 and the routine of FIG. 11 performs the previously described steps identified in blocks 434, 436 and 438.

FIG. 12 illustrates the operation of a decoder in accordance with this invention with respect to the mscale parameter produced for B-pictures. The min_ind parameter is extracted from the bit stream sent to the decoder in block 456. The quantization parameter for the slice is extracted from that bit stream in block 458. Block 460 computes the quantization step size from the quantization parameter for the slice in block 460. A decision then is made in block 462 as to whether the prediction for the macroblock is unidirectional or intra. If the prediction is unidirectional or intra, the current quantization step is computed as specified in block 464. If the prediction is bidirectional, the current quantization step size is computed as specified in block 456.

FIG. 13 shows a detailed block diagram of a visibility matrix selector. DCT coefficients are input to the selector on line 468 which is directed to the input of a coefficient weighting circuit 470. As those skilled in the art will appreciate, weighting factors are retrieved from a selected one of four visibility matrices 472, 474, 476 and 478, depending on the states of switching elements 480, 482, 484, 486, 488, and 490, which states are determined

by the condition of signals identified in FIG. 13 and discussed elsewhere. Depending on which visibility matrix is selected, predetermined weighting factors are directed on line 492 to an input of the coefficient waiting circuit 470 which then produces weighted DCT coefficients on an output line 494 in light of the DCT coefficients introduced on line 468 and the weighting factors introduced on line 492.

FIG. 14 is a block diagram of a forward/inverse scan selector in accordance with this invention. The scan selector of FIG. 14 comprises a forward/inverse scan formatting circuit 496 which receives a forward inverse/scan designation signal on line 498 and quantization indices on line 500. One of four predetermined scanning orders may be produced in accordance with information stored in four scanning blocks 502, 504, 506, and 508. Depending on the states of switching elements 510, 512, 514, 516, 518, and 520, which are determined by the nature of control signals identified in FIG. 14 and discussed elsewhere, the operation of the scan formatter 496 is controlled by signals on an input line 522 produced by a selected one of the scanning blocks 502, 504, 506, and 508. The scan formatter 496 produces scan quantization indices on line 524 in light of the inputs received on lines 498, 500 and 522.

FIG. 15 is a flow chart illustrating motion vector prediction for P-pictures. A new macroblock is obtained in block 526. Block 528 determines if this macroblock is the first macroblock of a slice. If it is the first macroblock, the variables stored in five registers identified in block 530 are initialized to zero and the routine of FIG. 15 proceeds to the input of a decision block 532. If it is determined in block 528 that this macroblock is not the first macroblock in a slice, the routine of FIG. 15 proceeds directly from the block 528 to the input of block 532. Block 532 determines whether or not this is an inter macroblock. If it is not a inter macroblock, the variables stored in five registers identified in block 534 are initialized to zero and the routine of FIG. 15 returns to the input of block 526.

If block 532 determines that the macroblock is an inter macroblock, then decisions are made as to which motion compensation type is involved. These decisions are made in blocks 536, 538, and 540. If the motion compensation type is type 1, then in block 542 a motion vector prediction variable is set equal to the value of the REG 16×16 FRM_P variable. In block 544 the REG 16×16 FRM_P variable is set equal to the value of a current motion vector variable and the motion prediction is output by the routine of FIG. 15.

If the motion compensation type is type 2, in block 546 a first motion vector prediction variable is set equal to the REG 16×16 FRM_P1 variable. The REG 16×16 FRM_P1 variable is changed to the value of a current motion vector variable (P1) in block 548. In block 550, a second motion vector prediction variable is set equal to the REG 16×16 FRM_P2 variable. In block 532, the REG 16×8 FRM_P2 variable is set equal to the value of a current motion vector variable (P2) and the motion prediction is output by the routine of FIG. 15.

If the motion compensation type is type 3, then as first motion vector prediction variable is set equal to the REG 16×8 FLD_P1 variable in block 554. In block 556, the REG 16×8 FLD_P1 variable then is set equal to a current motion vector (P1) variable. In block 558 a second motion vector prediction variable is set equal to the REG 16×8 FLD_P2 variable. The REG 16×8 FLD_P2 variable then is set equal to the value of a current motion vector (P2) variable in block 560 and the motion vector prediction is output by the routine of FIG. 15. An error condition is identified in block 561 in a situation where it is determined that the motion compensation type is not one of the three permitted types in FIG. 15.

FIG. 16 is a flow chart illustrating motion vector prediction for B-pictures. A new macroblock is obtained in block 562. A determination is made in block 564 as to whether this is the first macroblock in a slice. If it is the first macroblock in a slice, a list of variables stored in register identified in block 566 is initialized to zero. After the operation of block 566, the routine of FIG. 16 proceeds to the input of block 568. If the macroblock is not the first macroblock in a slice, as determined in block 564, then the output of block 564 is directly connected to the input of block 568.

Block 568 determines if this is an inter macroblock. If it is no an inter macroblock, block 570 initializes a list of variables stored in registers identified in block 570 to zero and the routine of FIG. 16 returns to the input of block 562. If block 568 determines that this is an inter macroblock, then blocks 572, 574, 576, 578, 580, 582, and 584 determine whether the motion compensation type is one of seven possible types.

If the motion compensation type is type 1 for B-pictures, block 586 sets a motion vector prediction variable equal to the variable REG 16×16 FRM_P. Block 588 then sets the REG 16×16 FRM_P variable to be equal to a current motion vector variable (P) and the motion vector prediction is output by the routine of FIG. 16.

If the motion compensation type is type 2 for B-pictures, block 590 sets a motion vector prediction variable equal to the REG 16×16 FRM_N variable. Block 592 sets the REG 16×16 FRM_N variable to the value of a current motion vector variable (N) and the motion vector prediction is output by the routine of FIG. 16.

If the motion compensation type is type 3 for B-pictures, block 594 sets a first motion vector prediction variable to the value of the REG 16×16 FRM_P variable. Block 596 then sets a current motion vector (P) variable to the value of the REG 16×16 FRM_P variable.

Block 598 sets a second motion vector prediction variable equal to the REG 16×16 FRM_N variable. Block 600 then sets the value of the REG 16×16 FRM_N to the value of a current motion vector variable (N) and the motion vector predictions are output by the routine of FIG. 16.

If the motion compensation type is type 4 for B-pictures, block 602 sets a first motion vector prediction variable equal to the REG 16×8 FRM_P1 variable. Block 604 changes the value of the REG 16×8 FRM_P1 variable to the value of a current motion vector (P1) variable. Block 606 sets a second motion vector prediction variable to the value of the REG 16×8 FRM_N2 variable. Block 608 changes the value of the REG 16×8 FRM_N2 variable to the value of a current motion vector (N2) variable and the motion vector predictions are output by the routine of FIG. 16.

If the motion compensation type is type 5 for B-pictures, block 610 sets a first motion vector prediction variable to be equal to the REG 16×8 FRM_N1 variable. Block 612 changes the value of the REG 16×8 FRM_N1 variable to the value of a current motion vector (N1) variable. Block 614 sets the value of a second motion vector prediction variable to be equal to the

value of the REG 16×8 FRM_P2 variable. Block **616** then changes the value of the REG 16×8 FRM_P2 variable to the value of a current motion vector (P2) variable and the motion vector prediction are output by the routine of FIG. **16**.

If the motion compensation type is type 6 for B-pictures, block **618** sets the value of a first motion vector prediction variable to the value of the REG 16×8 FLD_P1 variable. Block **620** changes the value of the RE 16×8 FLD_P1 variable to the value of a current motion vector (P1) variable. Block **622** sets the value of a second motion vector prediction variable to be equal to the value of the REG 16×8 FLD_N2 variable. Block **624** changes the value of the REG 16×8 FLD_N2 variable to the value of a current motion vector (N2) variable and the motion vector predictions are output by the routine of FIG. **16**.

If the motion compensation type is type 7 for B-pictures, block **626** sets the value of a first motion vector prediction variable to the value of the REG 16×8 FLD_N1 variable. Block **628** changes the value of the REG 16×8 FLD-N1 variable to be equal to the value of a current motion vector (N1) variable. Block **630** sets the value of a second motion vector prediction variable to be equal to the value of the REG 16×8 FLD_P2 variable. Block **632** changes the value of the REG 16×8 FLD_P2 variable to be the same as the value of a current motion vector (P2) variable and the motion vector predictions are output by the routine of FIG. **16**.

If the routine of FIG. **16** determines that the motion compensation type is not one of the seven permitted types, the routine of FIG. **16** identifies an error condition in block **634**.

A summary of the syntactical details of this example of the invention are shown below:

Sequence Header

sequence_header_code
horizontal_size
vertical_size
pel_aspect_ratio
picture_rate
bit_rate
intra_frame_quantizer_matrix [64]
intra_field_quantizer_matrix [64]
nonintra_frame_quantizer_matrix [64]
nonintra_field_quantizer_matrix [64]
mscale [64]

Group-of-Pictures Layer

group_start-code
time_code
closed_gap
broken_link

Picture Layer

picture_start_code
temporal_reference
picture_coding_type
full_pel_forward_vector (for P- and B-pictures)
forward_f (for P- and B-picture)
full_pel_backward_vector (for B-pictures)
backward_f (for B-pictures)

Slice Layer

slice_start_code
quantization_parameter

Macroblock Layer in Intra Coded MBs

macroblock_type
quantization_parameter (5 bits)
vlc_select (2 bits)

Macroblock Layer in Predictive-Coded MBs

macroblock_type
motion_horizontal_forward
motion_vertical_forward
macroblock_code_nocode (1 bit)
quantization_parameter (5 bits, sent when macroblock-_code_nocode is "1")
vlc_select (2 bits, sent when macroblock-_code_nocode is "1")

Macroblock Layer in Bidirectionally Predictive-Coded MBs

macroblock_type
motion_horizontal_forward
motion_vertical_forward
motion_horizontal_backward
motion_vertical_backward
macroblock_code_nocode (1 bit)
mscale_addr (2 bits, sent when macroblock-_code_nocode is "1")
coded_block_pattern (sent when macroblock-_code_nocode is "1")

Block Layer in Intra-Coded MBs

dct_dc_size
dct_dc_differential
dct_coeff_next
end_of_block (codeword depends on the codebook used)

Block Layer in Non Intra Coded MBs

dct_coeff_first
dct_coeff_next
end_of_block (in P-pictures, codeword depends on the codebook used)

The entire set of macroblock modes for all three picture types is listed in Table 1 below.

TABLE I

VLC TABLES FOR MACROBLOCK TYPES

macroblock_type in I-pictures:				
1	1	Intra		, frame-code
2	01	Intra		, field-code
macroblock_type in P-pictures:				
1	10	16×8	frame-MC,	frame-code
2	11	16×8	field-MC,	frame-code
3	01	16×16	frame-MC,	frame-code
4	0010	16×8	frame-MC,	field-code
5	0011	16×8	field-MC,	field-code
6	0001	16×16	frame-MC,	field-code
7	00001	Intra		, field-code
8	000001	Intra		, frame-code
macroblock_type in P-pictures:				
1	10	16×16	frame-MCbdr,	field-code
2	11	16×16	frame-MCbdr,	frame-code
3	010	16×16	frame-MCb,	frame-code
4	011	16×16	frame-MCb,	field-code
5	0010	16×16	frame-MCf,	field-code
6	0011	16×16	frame-MCf,	frame-code
7	00010	16×8	frame-MCbf,	frame-code
8	00011	16×8	field-MCfb,	frame-code
9	000010	16×8	frame-MCfb,	frame-code
10	000011	16×8	field-MCbf,	frame-code
11	0000010	16×8	field-MCfb,	field-code
12	0000011	16×8	frame-MCbf,	field-code

TABLE I-continued

VLC TABLES FOR MACROBLOCK TYPES			
13	00000010	16×8	field-MCbf, field-code
14	00000011	16×8	frame-MCfb, field-code
15	000000010	Intra	, frame-code
16	000000011	Intra	, field-code

In a fully automated one-pass encoder, the delay incurred by the encoding process is 2 to 3 picture periods. The delay at the decoder is 1 to 2 picture periods (not counting any postprocessing delays). However, the encoder and decoder buffers introduce a significantly greater delay of 0.5 seconds. The total code delay is thus 0.65 to 0.7 seconds.

I-pictures provide access points roughly every 0.4 seconds. Beginning with the I-picture, the other pictures in the GOP can be decoded as necessary to obtain the desired picture from the bit stream.

Fast forward and reverse enabled by having regularly spaced I-pictures.

The basic feature of our proposal have been presented, with all its syntactical details.

APPENDIX

The following appendix is an example of computer code written in the C programming language which may be used to create a working example of this invention on a programmed digital computer.

```
/*-------------------------------------------------------------------------------------------------*/
/*
        Atul Puri
        Copyright AT&T Bell Labs
        Sept 1991

        mscale.c

        Example of a code segment used for MSCALE quantization of B-frames
*/
/*-------------------------------------------------------------------------------------------------*/

                if (mean_hdr->step_mac[y_index][x_index] != 0)
                        {
                        if ((*ptr_type_pnb == 4) || (*ptr_type_pnb == 5))
                                {
                                tempscalef = qntpar->scaleb_mac*mean_hdr->step_mac[y_index][x_index];
                                ratiof = tempscalef/((float) dia_step); /* dia_step is default */
                                min_ind = 9;
                                min_valf = 1000.0;
                                for (z=0; z < 4; z++)
                                        {
                                        diff = ratiof - qntpar->mscaleb[z];
                                        absdiff = fabs(diff);
                                        if (absdiff < min_valf)
                                                {
                                                min_ind = z;
                                                min_valf = absdiff;
                                                }
                                        }
                                if (min_ind == 9)
                                        {
                                        fprintf(stderr, "In quant3..INTERP frame, error in MSCALE computation...quitting\n");
                                        exit(0);
                                        }
                                round_even(qntpar->mscaleb[min_ind]*dia_step, &qntpar->step_size);
                                }
                        else
                                {
                                tempscalef = qntpar->scalep_mac*mean_hdr->step_mac[y_index][x_index];
                                ratiof = tempscalef/((float) dia_step); /* dia_step is default */
                                min_ind = 9;
                                min_valf = 1000.0;
                                for (z=0; z < 4; z++)
                                        {
                                        diff = ratiof - qntpar->mscalep[z];
                                        absdiff = fabs(diff);
                                        if (absdiff < min_valf)
                                                {
                                                min_ind = z;
                                                min_valf = absdiff;
                                                }
                                        }
                                if (min_ind == 9)
                                        {
                                        fprintf(stderr, "In quant3..INTERP frame, error in MSCALE computation...quitting\n");
                                        exit(0);
```

```
                }
        round_even(qntpar->mscalep[min_ind]*dis_step, &qntpar->step_size);
        }
*mscale_ind = min_ind;
}
else
        {
        /* no coding, stay with default */
        }

if (qntpar->step_size > 62)
        {
        qntpar->step_size = 62;
        }
}
```



```
/*-------------------------------------------------------------------------------------------------------------*/
/*
        Atul Puri
        Copyright AT&T Bell Labs
        Sept 1991

        vlc_coeff_adap.c

        Adaptive VLC coding of coefficients and Intra DC prediction
*/
/*-------------------------------------------------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include "zzz.h"


/* VLTBLI0 and VLTBLI1: EOB 2-bits */

static int vltbli0[32][40] = {
{ 3, 4, 5, 6, 6, 7, 8, 8, 9, 9, 9,11,11,11,13,13,13,13,13,14,14,14,14,14,15,15,15,15,15,15,16,16,16,16,16,16,16,17,17,17},
{ 5, 7, 9,11,11,13,14,14,16,16,16,16,16,16,17,17,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 6, 9,13,13,14,16,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,11,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,13,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,13,16,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,14,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{11,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{11,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
```

```
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{16,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20}
};

/*vltbli1=vltblp0 */

static int vltbli1[32][40] ={
{ 3, 5, 6, 7, 9, 9,11,11,13,13,13,13,13,14,14,14,14,14,15,15,15,15,15,15,15,15,15,15,15,15,16,16,16,16,16,16,16,17,17,17},
{ 4, 6, 9,11,13,14,14,14,16,16,16,16,16,16,17,17,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 5, 8,11,14,14,16,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 6, 9,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,11,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,11,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,14,16,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,14,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{11,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{11,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{16,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20}
};

/* VLTBLI2: EOB 3-bits */

static int vltbli2[32][40] ={
{ 3, 4, 4, 6, 6, 7, 8, 8, 9, 9, 9,11,11,11,13,13,13,13,13,14,14,14,14,14,15,15,15,15,15,15,16,16,16,16,16,16,16,17,17,17},
```

```
{ 4, 7, 9,11,11,13,14,14,16,16,16,16,16,16,17,17,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 6, 9,13,13,14,16,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,11,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,13,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,13,16,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,14,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{11,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{11,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{16,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20}
};

/* VLTBLI3: EOB 4-bits */

static int vltbli3[32][40] ={
{ 3, 3, 5, 6, 6, 7, 8, 8, 9, 9, 9,11,11,11,13,13,13,13,13,14,14,14,14,14,15,15,15,15,15,15,16,16,16,16,16,16,16,17,17,17},
{ 4, 7, 9,11,11,13,14,14,16,16,16,16,16,16,17,17,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 6, 9,13,13,14,16,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,11,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,13,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,13,16,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,14,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{11,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{11,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
```

```
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{16,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20}
};

/* VLTBLP0 and VLTBLP1: EOB 2-bits */

static int vltblp0[32][40] ={
{ 3, 5, 6, 7, 9, 9,11,11,13,13,13,13,13,14,14,14,14,14,15,15,15,15,15,15,15,15,15,15,15,16,16,16,16,16,16,17,17,17},
{ 4, 6, 9,11,13,14,14,14,16,16,16,16,16,16,17,17,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 5, 8,11,14,14,16,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 6, 9,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,11,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,11,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,14,16,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,14,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{11,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{11,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{16,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20}
};

/* vltblp1 = vltbl10 */

static int vltblp1[32][40] ={
{ 3, 4, 5, 6, 6, 7, 8, 8, 9, 9, 9,11,11,11,13,13,13,13,13,14,14,14,14,14,15,15,15,15,15,15,16,16,16,16,16,16,16,17,17,17},
{ 5, 7, 9,11,11,13,14,14,16,16,16,16,16,16,17,17,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 6, 9,13,13,14,16,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,11,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
```

```
{ 7,13,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,13,16,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,14,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{11,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{11,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{16,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20}
};

/* VLTBLP2: EOB 3-bits */

static int vltblp2[32][40]={
{ 3, 4, 6, 7, 9, 9,11,11,13,13,13,13,13,14,14,14,14,14,15,15,15,15,15,15,15,15,15,15,15,15,16,16,16,16,16,16,16,17,17,17},
{ 4, 6, 9,11,13,14,14,14,16,16,16,16,16,16,17,17,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 4, 8,11,14,14,16,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 6, 9,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,11,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,11,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,14,16,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,14,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{11,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{11,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
```



```
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{16,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20}
};

/* VLTBLP3: EOB 4-bits */

static int vltblp3[32][40]={
{ 3, 4, 6, 7, 9, 9,11,11,13,13,13,13,13,14,14,14,14,14,15,15,15,15,15,15,15,15,15,15,15,15,16,16,16,16,16,16,16,17,17,17},
{ 3, 6, 9,11,13,14,14,14,16,16,16,16,16,16,17,17,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 5, 8,11,14,14,16,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 6, 9,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,11,13,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,11,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 7,14,16,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,14,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,14,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 8,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{ 9,17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{11,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{11,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{13,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{15,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{16,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20},
{17,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20}
};

static int    vlc_dcdif_lum[9] = {3, 2, 2, 3, 3, 4, 5, 6, 6};

static int    vlc_dcdif_chr[9] = {2, 2, 2, 3, 4, 5, 6, 7, 7};
```

```
vlc_2d2_adap(source, step_size, n_points,  n_bits,  parameter, type_macro, type_pnb_macro, block_index, index, mean_hdr, qntpar, error)
short   *source;                /* Pointer to source data to be VLC'd. INPUT */
int     *step_size;             /* Pointer to step size used by the quantizer routine. INPUT */
int     *type_macro;
int     *type_pnb_macro;
int     n_bits[4];              /* Number of bits used to code the coefficients. OUTPUT */
int     block_index, index;
int     n_points;               /* Number of coefficients to be coded in SOURCE. INPUT */
struct  codec   *parameter;
struct  q_stat  *qntpar;
struct  mean_mac        *mean_hdr;
int     *error;
        {

        int     qnt_index;
        int     i, cc, jk, level, cnt, run, rr, newi;
        int     sign;           /* Used to store the sign of the current source value if truncation is required */
        int     weight;
float   p_ind;
int     max_coeff_blk;
int     steparr[64];
int     *ptr_type, *ptr_type_pnb;

int     dc_prediction;
int     dc_dif, size;

int     itmp, jtmp;
FILE    *file_pointer;
static  int     first=1;
static  int     hist_amprun[64][32];
int     gof_id, seglen_id, frame_id;
int     search_id, range_id, intp_id, prevmc_id, origmc_id, fracmc_id;
float   avg_predf;
int     avg_predi;
int     tmp_mac_index;
int     u, ulimit;
short   *ptr_source;


max_coeff_blk = 64;


for (u=0; u < 4; u++)
        {
        n_bits[u] = 0;
        }
cnt = 0;
ptr_type = type_macro + block_index;
ptr_type_pnb = type_pnb_macro + block_index;

tmp_mac_index = block_index/6;

for(i=0;i<n_points;i++)
        {
```

```
/*---------RESET SIGN----------*/
sign = 0;

/*----------------------------LOAD STEP SIZE--------------------------------*/
if ((*ptr_type == 1) && (i == 0))
        {
        steparr[i] = 8; /* DC COEF of INTRA */
        }
else
        {
        steparr[i] = *step_size;
        }

/*----------------------------LOAD WEIGHT OF STEP---------------------------*/
if ((parameter->JPEG_QUANT==2) || ((parameter->JPEG_QUANT==1) && (*ptr_type==1)))
        {
        if ((index >= 0) && (index < 4))
                {
                weight = parameter->weight_zig_lum[i];
                }
        else
                {
                weight = parameter->weight_zig_chr[i];
                }
        }
else    /* (JPEG_QUANT==0) || (JPEG_QUANT==1 && *ptr_type==0) */
        {
        if (parameter->ENABLE_MATRIX)
                {
                weight = parameter->weight_zig_alt[i];
                }
        else
                {
                weight = 16;
                }
        }

/*----------------------------PROCESS RECONST'S-----------------------------*/
if ((*ptr_type == 1) && (i == 0))                       /* If Intra Block DC coef */
        {
        qnt_index = *source/steparr[i];
        if ((qnt_index < 0) || (qnt_index > 255))
                {
                exit(-1);
                }
        }
else                                                    /* Non DC coef INTRA, any coef INTER */
        {
        if (*source == 0)                               /* ZERO source */
                {
                qnt_index = 0;
                }
        else                                            /* NonZERO source */
```

```
{
if (*source < 0)
    {
    sign = 1;
    *source = - *source;
    }

if (*ptr_type == 1)                    /* Non DC coef INTRA */
    {
    if (parameter->JPEG_QUANT==1 || parameter->JPEG_QUANT==2)
        {
        p_ind = 0.833;
        }
    else                               /* CCITT */
        {
        p_ind = 0.0;
        }
    }
else
    {
    if (parameter->JPEG_QUANT==2)      /* Real JPEG */
        {
        p_ind = 1.0;
        }
    else                               /* Pseudo JPEG and CCITT */
        {
        p_ind = 0.0;
        }
    }
quant_for(*source, &qnt_index, steparr[1]/2, weight, p_ind, parameter, error);
}
}
/* ENSURE LIMITS OF QUANT INDEX */
if(qnt_index < 0 )
    {
    exit(-1);
    }

if ((*ptr_type == 1) && (i == 0))
    {
    if (parameter->JPEG_QUANT)
        {
        if ((index >= 0) && (index < 4))        /* LUM BLOCK */
            {
            /* reset prediction at start of slice in quant3.c */
            if ((((tmp_mac_index%45) == 0) || (*(ptr_type-6) != 1)) /* first macro of slice, prev macro non-intra*/
                {
                if ((index==0) || (index==1))
                    {
                    dc_prediction = mean_hdr->dc_pred_ltop;
```

```
mean_hdr->dc_pred_ltop = qnt_index;
if (index == 0)
    {
    mean_hdr->dc_pred_lbot = mean_hdr->dc_pred_ltop;
    }
else
    {
    dc_prediction = mean_hdr->dc_pred_lbot;
    mean_hdr->dc_pred_lbot = qnt_index;
    }
}
else
{
if ( *ptr_type_pnb ==  *(ptr_type_pnb-6) )
    {
    if ((index == 0) || (index == 1))
        {
        dc_prediction = mean_hdr->dc_pred_ltop;
        mean_hdr->dc_pred_ltop = qnt_index;
        }
    else
        {
        dc_prediction = mean_hdr->dc_pred_lbot;
        mean_hdr->dc_pred_lbot = qnt_index;
        }
    }
else
    {
    if ((index == 0) || (index == 1))
        {
        if (index == 0)
            {
                avg_predf = ((float)(mean_hdr->dc_pred_ltop + mean_hdr->dc_pred_lbot))/2
                roundp(avg_predf,&avg_pred1);
                mean_hdr->dc_pred_ltop = avg_pred1;
            }
        dc_prediction = mean_hdr->dc_pred_ltop;
        mean_hdr->dc_pred_ltop = qnt_index;
        if (index == 0)
            {
            mean_hdr->dc_pred_lbot = mean_hdr->dc_pred_ltop;
            }
        }
    else
        {
        dc_prediction = mean_hdr->dc_pred_lbot;
        mean_hdr->dc_pred_lbot = qnt_index;
        }
    }
}
else
```

```
                {
                if (index == 4)                                 /* CHR BLOCK */
                        {
                        dc_prediction = mean_hdr->dc_pred_cb;
                        mean_hdr->dc_pred_cb = qnt_index;
                        }

                if (index == 5)                                 /* CHR BLOCK */
                        {
                        dc_prediction = mean_hdr->dc_pred_cr;
                        mean_hdr->dc_pred_cr = qnt_index;
                        }
                }
        mean_hdr->dc_predictor[index] = dc_prediction;  /* Note....used in vlc_2d2adap_bits_()...only... */
        dc_dif = qnt_index - dc_prediction;
        size_dc_dif(dc_dif, &size, parameter, error);

        if ((index >= 0) && (index < 4))                /* LUM BLOCK */
                {
                n_bits[0] = n_bits[1] = n_bits[2]  = n_bits[3] = (size + vlc_dcdif_lum[size]);
                }
        else                                            /* CHR BLOCK */
                {
                n_bits[0] = n_bits[1] = n_bits[2]  = n_bits[3] = (size + vlc_dcdif_chr[size]);
                }
        }
}
else
{
/* COMPUTE RUN */
level = qnt_index;
if ((i == 0) && (level != 0))
        {
        run = 0;
        }
else
        {
        if (level == 0)
                cnt++;
        else
                {
                run = cnt;
                cnt = 0;
                }
        }

/* 2D VLC CODE (RUN, LEVEL) using all 4 tables  */
if (qntpar->intra_frm == 1)
        ulimit = 4;
else if (qntpar->pred_frm == 1)
        ulimit = 4;
else if (qntpar->intp_frm == 1)
        ulimit = 1;

for (u=0; u < ulimit; u++)
        {
        if (level != 0)
```

```
{
if (((level >= 1)&&(level <= 40)) && ((run >= 0)&&(run <= 31)))
        {
        switch (u)
                {
        case 0:
                if (qntpar->intra_frm == 1)
                        n_bits[0] += vltbli0[run][level-1];
                else if (qntpar->pred_frm == 1)
                        n_bits[0] += vltblp0[run][level-1];
                else if (qntpar->intp_frm == 1)
                        n_bits[0] += vltblp0[run][level-1];
                break;
        case 1:
                if (qntpar->intra_frm == 1)
                        n_bits[1] += vltbli1[run][level-1];
                else if (qntpar->pred_frm == 1)
                        n_bits[1] += vltblp1[run][level-1];
                break;
        case 2:
                if (qntpar->intra_frm == 1)
                        n_bits[2] += vltbli2[run][level-1];
                else if (qntpar->pred_frm == 1)
                        n_bits[2] += vltblp2[run][level-1];
                break;
        case 3:
                if (qntpar->intra_frm == 1)
                        n_bits[3] += vltbli3[run][level-1];
                else if (qntpar->pred_frm == 1)
                        n_bits[3] += vltblp3[run][level-1];
                break;
                }
        }
else
        {
        if (((level > 40)&&(level <=128)) || ((run > 31)&&(run <=63)))
                {
                switch (u)
                        {
                case 0:
                        n_bits[0] += 20;
                        break;
                case 1:
                        n_bits[1] += 20;
                        break;
                case 2:
                        n_bits[2] += 20;
                        break;
                case 3:
                        n_bits[3] += 20;
                        break;
                        }
```

```
    }
    else
    {
    if (parameter->VLC_EXTEND)
        {
            if ((level > 128)&&(level <=255))
                {
                switch (u)
                    {
                    case 0:
                        n_bits[0] += 28;
                        break;
                    case 1:
                        n_bits[1] += 28;
                        break;
                    case 2:
                        n_bits[2] += 28;
                        break;
                    case 3:
                        n_bits[3] += 28;
                        break;
                    }
                }
            else
                {
                    exit(-1);
                }
            }
        else
            {
            exit(-1);
            }
        }
    }
}
}
}
}
}
source++;
}
}
```

```
/*-----------------------------------------------------------------------------------------------------------*/
/*
        Atul Puri
        Copyright AT&T Bell Labs
        Sept 1991

        load_qntpar.c

        Loads Inter/Intra matrices and MSCALE parameters
*/
/*-----------------------------------------------------------------------------------------------------------*/

#include <stdio.h>
#include "zzz.h"

load_qntpar(qntpar, error)
struct  q_stat  *qntpar;
int     *error;
        {
        register int    i;
        FILE    *fp_qntpar;

        /* open qnt_file to read quant. parameters */
        if ((fp_qntpar = fopen("qntpar_file", "r")) == NULL)
                {
                fprintf(stderr, "In LOAD_QNTPAR.....unable to open qntpar_file....quitting..!!\n");
                exit(0);
                }

        /* Read jpeg_lum_frm[] and jpeg_lum_fld[], weighting matrix for INTRA, frame and field blocks */
        for (i=0; i < 64; i++)
                {
                fscanf(fp_qntpar, "%d", &qntpar->jpeg_lum_frm[i]);
                }
        for (i=0; i < 64; i++)
                {
                fscanf(fp_qntpar, "%d", &qntpar->jpeg_lum_fld[i]);
                }

        /* copy jpeg_lum_frm[] matrix to jpeg_chr_frm[] and jpeg_lum_fld[] matrix to jpeg_chr_fld[] */
        for (i=0; i < 64; i++)
                {
                qntpar->jpeg_chr_frm[i] = qntpar->jpeg_lum_frm[i];
                qntpar->jpeg_chr_fld[i] = qntpar->jpeg_lum_fld[i];
                }

        /* Read ccit_alt_frm[]  and ccit_alt_fld[], weighting matrices for NON-INTRA, frame and field blocks */
        for (i=0; i < 64; i++)
                {
                fscanf(fp_qntpar, "%d", &qntpar->ccit_alt_frm[i]);
                }
        for (i=0; i < 64; i++)
                {
                fscanf(fp_qntpar, "%d", &qntpar->ccit_alt_fld[i]);
                }

        /* Read mscaleb[] and mscalep[], local scale factors for mquant of INTP frames */
        for (i=0; i < 4; i++)
                {
                fscanf(fp_qntpar, "%f", &qntpar->mscaleb[i]);
                }
        for (i=0; i < 4; i++)
```

```
    {fscanf(fp_qntpar, "%f", &qntpar->scalep[1]);
     }

/* Read scaleb_mac and scalep_mac, global scale factors for mquant of INTP frames */
fscanf(fp_qntpar, "%f", &qntpar->scaleb_mac);
fscanf(fp_qntpar, "%f", &qntpar->scalep_mac);

/* Read step_var1[] and var_lnt1[].  step sizes and variance thresholds corresponding to activity classes for INTRA frame */
for (i=0; i < 8; i++)
        {fscanf(fp_qntpar, "%d", &qntpar->step_var1[i]);
        }

for (i=0; i < 8; i++)
        {fscanf(fp_qntpar, "%d", &qntpar->var_lnt1[i]);
        }

/* Read step_varp[] and var_lntp[], step sizes and variance thresholds corresponding to activity classes for PRED frame */
for (i=0; i < 8; i++)
        {fscanf(fp_qntpar, "%d", &qntpar->step_varp[i]);
        }

for (i=0; i < 8; i++)
        {fscanf(fp_qntpar, "%d", &qntpar->var_lntp[i]);
        }

    /* close qntpar file */
    fclose (fp_qntpar);
    }

#include <stdio.h>
#include "zzz.h"

vlc_mvadap_pred (val_map_adap, index, type_pnb, parameter, mv_bits)

int     *val_map_adap;
int     index;
struct codec *parameter;
int     *type_pnb;
int     *mv_bits;

        {
        int     mod, quo;
        int     low, high, range, tmp_diff;
        int     i, n_points;
        int     dist ;
        int     *ptr_vec;
        int     *ptr_type;
        int     curr_vec_px1, curr_vec_py1, curr_vec_px2, curr_vec_py2, diff_vec, macros_in_slice;

        int     MV_LMT;

        static  int first = 1;
        static  FILE *p_coufil;

        static  int reg_16x16frm_px, reg_16x16frm_py;
        static  int reg_16x8frm_px1, reg_16x8frm_py1, reg_16x8frm_px2, reg_16x8frm_py2;
        static  int reg_16x8fld_px1, reg_16x8fld_py1, reg_16x8fld_px2, reg_16x8fld_py2;
```

```
macros_in_slice = 45;
*mv_bits = 0;

ptr_vec = vml_map_adap + 4*index;
ptr_type = type_pnb + index;

curr_vec_px1 = *ptr_vec;
curr_vec_py1 = *(ptr_vec + 1);

curr_vec_px2 = *(ptr_vec + 2);
curr_vec_py2 = *(ptr_vec + 3);

/* "dist" is now indicated by "forward_f" .... */
dist = parameter->forward_f;
if (dist < 1)
    {
    fprintf (stderr, "vlc_mvl_pred: dist %d too small. STOP\n", dist) ;
    exit (-1);
    }

if( (index % macros_in_slice) == 0)
    {
    reg_16x16frm_px = reg_16x16frm_py = 0;
    reg_16x8frm_px1 = reg_16x8frm_py1 = reg_16x8frm_px2 = reg_16x8frm_py2 = 0;
    reg_16x8fld_px1 = reg_16x8fld_py1 = reg_16x8fld_px2 = reg_16x8fld_py2 = 0;
    }

/* Begin MV processing ... */
switch (*ptr_type)
    {
case 0:
case 1:
    diff_vec = curr_vec_px1 - reg_16x16frm_px;
    mv_send (diff_vec, dist, mv_bits,parameter->symbits_ind);
    diff_vec = curr_vec_py1 - reg_16x16frm_py;
    mv_send (diff_vec, dist, mv_bits, parameter->symbits_ind);
    reg_16x16frm_px = curr_vec_px1;
    reg_16x16frm_py = curr_vec_py1;
    break;
case 2:
case 3:
    diff_vec = curr_vec_px1 - reg_16x8frm_px1;
    mv_send (diff_vec, dist, mv_bits, parameter->symbits_ind);
    diff_vec = curr_vec_py1 - reg_16x8frm_py1;
    mv_send (diff_vec, dist, mv_bits, parameter->symbits_ind);
    diff_vec = curr_vec_px2 - reg_16x8frm_px2;
    mv_send (diff_vec, dist, mv_bits, parameter->symbits_ind);
    diff_vec = curr_vec_py2 - reg_16x8frm_py2;
    mv_send (diff_vec, dist, mv_bits, parameter->symbits_ind);
    reg_16x8frm_px1 = curr_vec_px1;
    reg_16x8frm_py1 = curr_vec_py1;
    reg_16x8frm_px2 = curr_vec_px2;
    reg_16x8frm_py2 = curr_vec_py2;
    break;
case 4:
case 5:
    diff_vec = curr_vec_px1 - reg_16x8fld_px1;
    mv_send (diff_vec, dist, mv_bits, parameter->symbits_ind);
    diff_vec = curr_vec_py1 - reg_16x8fld_py1;
    mv_send (diff_vec, dist, mv_bits, parameter->symbits_ind);
    diff_vec = curr_vec_px2 - reg_16x8fld_px2;
```

```
        mv_send (diff_vec, dist, mv_bits, parameter->symbits_ind);
        diff_vec = curr_vec_py2 - reg_16x8fld_py2;
        mv_send (diff_vec, dist, mv_bits, parameter->symbits_ind);
        reg_16x8fld_px1 = curr_vec_px1;
        reg_16x8fld_py1 = curr_vec_py1;
        reg_16x8fld_px2 = curr_vec_px2;
        reg_16x8fld_py2 = curr_vec_py2;
        break;

    case 6:
    case 7:

        reg_16x16frm_px = reg_16x16frm_py = 0;
        reg_16x8frm_px1 = reg_16x8frm_px2 = reg_16x8frm_py2 = 0;
        reg_16x8fld_px1 = reg_16x8fld_py1 = reg_16x8fld_px2 = reg_16x8fld_py2 = 0;
        *mv_bits += 0;
        break;

    default:
        fprintf(stderr, "In VLC_MVADAP...Illegal ptr_type=%d,...quitting..\n", *ptr_type);
        exit(0);
        }

    terminate::
    }

#include <math.h>
#include <stdio.h>
#include "zzz.h"
#include "myalloc.h"

mot_pred_adap(prev_frm, curr_frm, pred_frm, val_map, val_map_frmsub, val_map_fldsub, type_mcadap, parameter)
short   *prev_frm;
short   *curr_frm;
short   *pred_frm;
int     *val_map;
int     *val_map_frmsub;
int     *val_map_fldsub;
int     *type_mcadap;
struct codec_parameter;

{
    register int i,j,k,s,p;
    int     x_blocks, y_blocks, blocks_in_frm;
    int     mot_bl_size;

    int     prev_frm_x, prev_frm_y;
    int     curr_frm_x, curr_frm_y;

    int     *ptr_x_vec;
    int     *ptr_y_vec;

    short   **curr_array, **prev_fld0, **prev_fld1;

    short   **prev_blint, **prev_blint_fld0, **prev_blint_fld1;

    int     lum_ind;

    short   frameblk[16][16], fld0blk[8][16], fld1blk[8][16];

    float newflt;

    int     i, temp_dif, min_ind, sum_sqdif;
    int     count_frmmac_mac, count_frmsub_mac, count_fldsub_mac;
```

```
float   avg_sqdif[3], ref0, ref2, T0, T2;
short   predblk[3][16][16];
count_frmmac_mac = count_frmsub_mac = count_fldsub_mac = 0;
for(i=0;i<y_blocks;i++)                 /* do for all y blocks */
        {
        for(j=0;j<x_blocks;j++)         /* do for all x blocks */
                {
                /*-------------------------------------------------------------------------------*/
                /*                      STEP 1.         PREDICT 16x16 FRAME BLOCK                */
                /*-------------------------------------------------------------------------------*/

                ptr_x_vec = vel_map;
                ptr_y_vec = vel_map + 1;
    curr_frm_x = j*mot_bl_size;
    curr_frm_y = i*mot_bl_size;
    if (parameter->MOT_EST_FRAC)
            {
            prev_frm_x = 2*curr_frm_x + (*ptr_x_vec);
            prev_frm_y = 2*curr_frm_y + (*ptr_y_vec);
            for(s = 0; s < 2*mot_bl_size; s += 2)
                    {
                    for(p = 0; p < 2*mot_bl_size; p += 2)
                            {
                            predblk[0][s/2][p/2] = prev_blint[s + prev_frm_y][p + prev_frm_x];
                            }
                    }
            }
    else    /* INTEGER PEL MOT COMP */
            {
            prev_frm_x = curr_frm_x + (*ptr_x_vec);
            prev_frm_y = curr_frm_y + (*ptr_y_vec);
            for(s = 0; s < mot_bl_size; s++)
                    {
                    for(p = 0; p < mot_bl_size; p++)
                            {
                            predblk[0][s][p] = prev_blint[s + prev_frm_y][p + prev_frm_x];
                            }
                    }
            }
    vel_map++;
    vel_map++;
    for (k=0; k < 2; k++)
            {
            /*-------------------------------------------------------------------------------*/
            /*                      STEP 2.         PREDICT 16x8 FRAME BLOCK                 */
            /*-------------------------------------------------------------------------------*/

            ptr_x_vec = vel_map_frmsub;
            ptr_y_vec = vel_map_frmsub + 1;
            curr_frm_x = j*mot_bl_size;
            curr_frm_y = i*mot_bl_size + k*mot_bl_size/2;
```

```
if (parameter->HOT_EST_FRAC)
    {
    prev_frm_x = 2*curr_frm_x + (*ptr_x_vec);
    prev_frm_y = 2*curr_frm_y + (*ptr_y_vec);

    for(s = 0; s < 2*mot_bl_size/2; s += 2)
        {for(p = 0; p < 2*mot_bl_size; p += 2)
            {
            predblk[1][s/2 + k*mot_bl_size/2][p/2] = prev_blint[s + prev_frm_y][p + prev_frm_x];
            }/* end for p */
        }/* end for s */
    }
else
    {
    prev_frm_x = curr_frm_x + (*ptr_x_vec);
    prev_frm_y = curr_frm_y + (*ptr_y_vec);

    for (s = 0; s < mot_bl_size/2; ++s)
            {for (p = 0; p < mot_bl_size; ++p)
                {
                predblk[1][s + k*mot_bl_size/2][p] = prev_array[s + prev_frm_y][p + prev_frm_x];
                }/* end for p */
            }/* end for s */
    }
val_map_frmsub++;
val_map_frmsub++;

/*----------------------------------------------------------------------*/
/*              STEP 3.     PREDICT 16x8 FIELD BLOCK                    */
/*----------------------------------------------------------------------*/

ptr_x_vec = val_map_fldsub;
ptr_y_vec = val_map_fldsub + 1;

curr_frm_x = 1*mot_bl_size;
curr_frm_y = 1*mot_bl_size/2;

if (parameter->HOT_EST_FRAC)
    {
    prev_frm_x = 2*curr_frm_x + (*ptr_x_vec);
    prev_frm_y = 4*curr_frm_y + (*ptr_y_vec);

    for(s = 0; s < 2*mot_bl_size; s += 4)
        {for(p = 0; p < 2*mot_bl_size; p += 2)
            {
            if (k == 0)
                {fld0blk[s/4][p/2] = prev_blint_fld0[s + prev_frm_y][p + prev_frm_x];
                }
            else
                {fld1blk[s/4][p/2] = prev_blint_fld1[s + prev_frm_y][p + prev_frm_x];
                }
            }
        }
    }
```

```
    }
else
    {
    prev_frm_x = curr_frm_x + (*ptr_x_vec);
    prev_frm_y = 2*curr_frm_y + (*ptr_y_vec);

    for (s = 0; s < mot_bl_size; s += 2)
        {for (p = 0; p < mot_bl_size; ++p)
            {if (k == 0)
                {fld0blk[s/2][p] = prev_fld0[s + prev_frm_y][p + prev_frm_x];
                }
            else
                {fld1blk[s/2][p] = prev_fld1[s + prev_frm_y][p + prev_frm_x];
                }
            }
        }
    }
vel_map_fldsub++;
vel_map_fldsub++;
/*-------------------------------------------------------------------*/
}/* end for k, two field/frame split*/

for (s=0; s < mot_bl_size; s++)
    {
    for (p=0; p < mot_bl_size; p++)
        {
        if (s%2 == 0)
            {
            predblk[2][s][p] = fld0blk[s/2][p];
            }
        else
            {
            predblk[2][s][p] = fld1blk[s/2][p];
            }
        }
    }
/*--------------------------------------------------------------------*/
/*                                                                    */
/*                STEP 4.      ADAPT BETWEEN THREE PREDICTIONS        */
/*                                                                    */
/*--------------------------------------------------------------------*/

/* Step 4(a) compute all three differences */
curr_frm_x = 1*mot_bl_size;
curr_frm_y = 1*mot_bl_size;

for (l=0; l < 3; l++)
    {
    avg_sqdif[l] = 0.0;
    sum_sqdif = temp_dif = 0;
    for (s=0; s < mot_bl_size; s++)
        {for (p=0; p < mot_bl_size; p++)
            {
            temp_dif = curr_array[s+curr_frm_y][p+curr_frm_x] - predblk[l][s][p];
            sum_sqdif += abs(temp_dif);
            }
```

```
        }
    avg_sqdif[1] = sum_sqdif;
    }

/* Step 4(b) compare differences determine min_ind */
T0 = 1.0;
ref0 = ((float)avg_sqdif[0])/T0;

T2 = 1.0;
ref2 = ((float)avg_sqdif[2])/T2;

if ((avg_sqdif[1] < ref0) || (avg_sqdif[2] < ref0))
    {
    if (avg_sqdif[1] < ref2)
        {
        min_ind = 1;
        count_frmsub_mac++;
        }
    else
        {
        min_ind = 2;
        count_fldsub_mac++;
        }
    }
else
    {
    min_ind = 0;
    count_frmmac_mac++;
    }

/* Step 4(c) Transfere to prediction frame */
for (s=0; s < mot_bl_size; s++)
    {
    for (p=0; p < mot_bl_size; p++)
        {
        pred_frm = predblk[min_ind][s][p];
        pred_frm++;
        }
    }

/* Step 4(d) save mc adaptation info */
*type_mcadap = min_ind;
type_mcadap++;
}/* end for j */
}/* end for i */
}

#include <math.h>
#include <stdio.h>
#include "zzz.h"


find_scanmode(type_pnb, mac_index, scanmode, parameter, qntpar, error)
int     *type_pnb;
int     mac_index;
int     *scanmode;
```

```
struct codec *parameter;
struct q_stat *qntpar;

{
        register int i,j,k,s,p,itmp;
        int     x_blocks, y_blocks, blocks_in_frm;
        int     mot_bl_size;
        int     *ptr_type;


        ptr_type = type_pnb + mac_index;

        if (qntpar->intra_frm == 1)
                {
                if ((*ptr_type >=0) && (*ptr_type < 2))
                        {
                        if ((*ptr_type % 2) == 0)       /* frame */
                                {
                                *scanmode = 2;
                                }
                        else                            /* field */
                                {
                                *scanmode = 3;
                                }
                        }
                else
                        {
                        fprintf(stderr,"In find_scanmode...illegal..type_pnb=%d for...Intra picture..quitting\n", *ptr_type);
                        exit(0);
                        }
                }
        else if (qntpar->pred_frm == 1)
                {
                if ((*ptr_type >=0) && (*ptr_type < 8))
                        {
                        if ((*ptr_type % 2) == 0)       /* frame */
                                {
                                *scanmode = 2;
                                }
                        else                            /* field */
                                {
                                *scanmode = 3;
                                }
                        }

          else
                {
                fprintf(stderr,"In find_scanmode...illegal..type_pnb=%d for...Pred picture..quitting\n", *ptr_type);
                exit(0);
                }
          }
  else if (qntpar->intp_frm == 1)
          {
```

```
        if ((*ptr_type >=0) && (*ptr_type < 16))
                {
                if ((*ptr_type % 2) == 0)       /* frame */
                        {
                        *scanmode = 2;
                        }
                else                            /* field */
                        {
                        *scanmode = 3;
                        }
                }
        else
                {
                fprintf(stderr,"In find_scanmode...illegal..type_pnb=%d for...Interp picture..quitting\n", *ptr_type);
                exit(0);
                }
        }

    )/* end find_scanmode */

#include <stdio.h>
#include "zzz.h"

vlc_mvadap_intp(vel_map_adap, index, type_pnb, parameter, mv_bits)

int     *vel_map_adap;
int     index;

struct codec *parameter;
int     *type_pnb;

int     *mv_bits;

        {
        int     mod, quo;
        int     low, high, range, tmp_diff;
        int     i, n_points;
        int     dist ;
        int     *ptr_vec;
        int     *ptr_type;
        int     curr_vec_x1, curr_vec_y1, curr_vec_x2, curr_vec_y2, diff_vec, macros_in_slice;

        int     MV_LMT;

        static  int first = 1;
        static  FILE *p_coufil;

        static  int reg_16x16frm_px, reg_16x16frm_py;
        static  int reg_16x16frm_nx, reg_16x16frm_ny;
        static  int reg_16x8frm_px1, reg_16x8frm_py1, reg_16x8frm_px2, reg_16x8frm_py2;
        static  int reg_16x8frm_nx1, reg_16x8frm_ny1, reg_16x8frm_nx2, reg_16x8frm_ny2;
        static  int reg_16x8fld_px1, reg_16x8fld_py1, reg_16x8fld_px2, reg_16x8fld_py2;
        static  int reg_16x8fld_nx1, reg_16x8fld_ny1, reg_16x8fld_nx2, reg_16x8fld_ny2;

        macros_in_slice = 45;
        *mv_bits = 0;
```

```
    ptr_vec =  val_map_adap + 4*index;
    ptr_type = type_mb + index;

    curr_vec_x1 = *ptr_vec;
    curr_vec_y1 = *(ptr_vec + 1);

    curr_vec_x2 = *(ptr_vec + 2);
    curr_vec_y2 = *(ptr_vec + 3);

    /* Some checking ... */
    dist = parameter->forward_f;
    if( (dist < 1) || (dist > 6))
            {
                fprintf (stderr, "vlc_mvadap_intp forward_f %d out of range. STOP\n", dist);
                exit (-1) ;
                }

    dist = parameter->backward_f;
    if( (dist < 1) || (dist > 6))
        {
        fprintf (stderr, "vlc_mvadap_intp: backward_f %d out of range. STOP\n", dist);
        exit (-1) ;
        }

    if( (index % macros_in_slice) == 0)
            {
            reg_16x16frm_px = reg_16x16frm_py = 0;
            reg_16x16frm_nx = reg_16x16frm_ny = 0;
            reg_16x8frm_px1 = reg_16x8frm_py1 = reg_16x8frm_px2 = reg_16x8frm_py2 = 0;
            reg_16x8frm_nx1 = reg_16x8frm_ny1 = reg_16x8frm_nx2 = reg_16x8frm_ny2 = 0;
            reg_16x8fld_px1 = reg_16x8fld_py1 = reg_16x8fld_px2 = reg_16x8fld_py2 = 0;
            reg_16x8fld_nx1 = reg_16x8fld_ny1 = reg_16x8fld_nx2 = reg_16x8fld_ny2 = 0;
            }

    /* Begin MV processing ... */
    switch (*ptr_type)
        {
        case 0:
        case 1: /* 16x16 frm p */
                dist = parameter->forward_f;
                diff_vec = curr_vec_x1 - reg_16x16frm_px;
                mv_send (diff_vec, dist, mv_bits,parameter->symbits_ind);
                diff_vec = curr_vec_y1 - reg_16x16frm_py;
                mv_send (diff_vec, dist, mv_bits,parameter->symbits_ind);
                reg_16x16frm_px = curr_vec_x1;
                reg_16x16frm_py = curr_vec_y1;
                break;
        case 2:
        case 3: /* 16x16 frm n */
                dist = parameter->backward_f;
                diff_vec = curr_vec_x1 - reg_16x16frm_nx;
                mv_send (diff_vec, dist, mv_bits,parameter->symbits_ind);
                diff_vec = curr_vec_y1 - reg_16x16frm_ny;
                mv_send (diff_vec, dist, mv_bits,parameter->symbits_ind);
                reg_16x16frm_nx = curr_vec_x1;
                reg_16x16frm_ny = curr_vec_y1;
                break;
        case 4:
        case 5: /* 16x16 frm b */
                dist = parameter->forward_f;
                diff_vec = curr_vec_x1 - reg_16x16frm_px;
```

```
        mv_send (diff_vec, dist, mv_bits,parameter->symbits_ind);
        diff_vec = curr_vec_y1 - reg_16x16frm_py;
        mv_send (diff_vec, dist, mv_bits,parameter->symbits_ind);
        dist = parameter->backward_f;
        diff_vec = curr_vec_x2 - reg_16x16frm_nx;
        mv_send (diff_vec, dist, mv_bits,parameter->symbits_ind);
        diff_vec = curr_vec_y2 - reg_16x16frm_ny;
        mv_send (diff_vec, dist, mv_bits,parameter->symbits_ind);
        reg_16x16frm_px = curr_vec_x1;
        reg_16x16frm_py = curr_vec_y1;
        reg_16x16frm_nx = curr_vec_x2;
        reg_16x16frm_ny = curr_vec_y2;
        break;
    case 6:
    case 7: /* 16x8 frm pn */
        dist = parameter->forward_f;
        diff_vec = curr_vec_x1 - reg_16x8frm_px1;
        mv_send (diff_vec, dist, mv_bits,parameter->symbits_ind);
        diff_vec = curr_vec_y1 - reg_16x8frm_py1;
        mv_send (diff_vec, dist, mv_bits,parameter->symbits_ind);
        reg_16x8frm_nx1 = reg_16x8frm_nx2 = reg_16x8frm_px2 = 0;
        reg_16x8frm_ny1 = reg_16x8frm_py2 = 0;
        break;
    default:
        fprintf(stderr, "In vlc_mvadap_intp...illegal ptr_type=%d,...quitting..\n", *ptr_type);
        exit(0);
        }
    }

#include <math.h>
#include <stdio.h>
#include "zzz.h"

frame_field(curr_frm,adap_frm,type_codadap,type_mcadap,type_ifdadap,parameter)
short  *curr_frm;
short  *adap_frm;
int    *type_codadap;
int    *type_mcadap;
int    *type_ifdadap;
struct codec *parameter;
{
    register int i,j,k,s,p;
    int     x_blocks, y_blocks, blocks_in_frm;
    int     mot_bl_size;

    int     prev_frm_x, prev_frm_y;
    int     curr_frm_x, curr_frm_y;

    short   **curr_array;
    int     lum_ind;

    double  sum, sumsq, sumhorsq, sumversq;
    float   npels_frmmc, npels_fldmc;
    double  var, varhor, varver, avg, avgsq;
    double  phor, pverfrm, pverfld;
```

```
double  pverfld_slice[90];
int     count_frmmac, count_fldmac;
int     cnt_16x16frmmc_frmcod, cnt_16x16frmmc_fldcod, cnt_16x8frmmc_frmcod, cnt_16x8frmmc_fldcod;
int     cnt_16x8fldmc_frmcod, cnt_16x8fldmc_fldcod, cnt_intra_frmcod, cnt_intra_fldcod;
static  int     first = 1;

FILE    *file_pointer;
int     gof_id, seglen_id, frame_id;
int     search_id, range_id, intp_id, prevmc_id, origmc_id, fracmc_id;


cnt_16x16frmmc_frmcod = cnt_16x16frmmc_fldcod = cnt_16x8frmmc_frmcod = cnt_16x8frmmc_fldcod = 0;
cnt_16x8fldmc_frmcod = cnt_16x8fldmc_fldcod = cnt_intra_frmcod = cnt_intra_fldcod = 0;

count_frmmac = count_fldmac = 0;
npels_frmmac = 256.0, npels_fldmac = 128.0;
for(i=0;i<y_blocks;i++)                 /* do for all y blocks */
        {
        for(j=0;j<x_blocks;j++)         /* do for all x blocks */
                {
                curr_frm_x = j*mot_bl_size;
                curr_frm_y = i*mot_bl_size;

                /*-------------------------------------------------------------------------------------------------*/
                /*                      STEP 1.         FIND horiz and vertical frame correlation                  */
                /*-------------------------------------------------------------------------------------------------*/

            sumsq = sum = 0.0;
            avg = avgsq = var = 0.0;
            for(s = 0; s < mot_bl_size; s++)
                    {
                    for(p = 0; p < mot_bl_size; p++)
                            {
                            sum += curr_array[s+curr_frm_y][p+curr_frm_x];
                            sumsq += (curr_array[s+curr_frm_y][p+curr_frm_x]*curr_array[s+curr_frm_y][p+curr_frm_x]);
                            }
                    }

            avg = sum / npels_frmmac;
            avgsq = sumsq / npels_frmmac;
            var = avgsq - (avg*avg);

            sumhorsq = sumversq = 0.0;
            varhor = varver = phor = pverfrm = 0.0;
            for(s = 0; s < mot_bl_size; s++)
                    {
                    for(p = 0; p < mot_bl_size; p++)
                            {
                            if ((p+curr_frm_x+1) >= (x_blocks*mot_bl_size))
                                    {
                                    sumhorsq += ((curr_array[s+curr_frm_y][p+curr_frm_x] - avg) *
                                                 (curr_array[s+curr_frm_y][p+curr_frm_x] - avg));
                                    }
                            else
                                    {
                                    sumhorsq += ((curr_array[s+curr_frm_y][p+curr_frm_x] - avg) *
                                                 (curr_array[s+curr_frm_y][p+curr_frm_x+1] - avg));
                                    }

                            if ((s+curr_frm_y+1) >= (y_blocks*mot_bl_size))
                                    {
```

```
                        sumversq += ((curr_array[s+curr_frm_y][p+curr_frm_x] - avg) *
                                     (curr_array[s+curr_frm_y][p+curr_frm_x] - avg));
                        }
                    else
                        {
                        sumversq += ((curr_array[s+curr_frm_y][p+curr_frm_x] - avg) *
                                     (curr_array[s+curr_frm_y+1][p+curr_frm_x] - avg));
                        }
                    }
            }
    varhor = sumhorsq/npels_frmmac;
    phor = varhor/var;

    varver = sumversq/npels_frmmac;
    pverfrm = varver/var;

    fprintf(file_pointer, "%3.2f%3.2f ", phor, pverfrm);

    /*-----------------------------------------------------------------------------*/
    /*                  STEP 2.        FIND vertical field correlation              */
    /*-----------------------------------------------------------------------------*/
    sumsq = sum = 0.0;
    avg = avgsq = var = 0.0;
    for(s = 0; s < mot_bl_size; s += 2)
            {
            for(p = 0; p < mot_bl_size; p++)
                    {
                    sum += curr_array[s+curr_frm_y][p+curr_frm_x];
                    sumsq += (curr_array[s+curr_frm_y][p+curr_frm_x]*curr_array[s+curr_frm_y][p+curr_frm_x]);
                    }
            }

   avg = sum / npels_fldmac;
   avgsq = sumsq /npels_fldmac;
   var = avgsq - (avg*avg);

   sumversq = 0.0;
   varver = pverfld = 0.0;
   for(s = 0; s < mot_bl_size; s += 2)
           {
           for(p = 0; p < mot_bl_size; p++)
                   {
                   if ((s+curr_frm_y+2) >= (y_blocks*mot_bl_size))
                           {
                           sumversq += ((curr_array[s+curr_frm_y][p+curr_frm_x] - avg) *
                                        (curr_array[s+curr_frm_y][p+curr_frm_x] - avg));
                           }
                   else
                           {
                           sumversq += ((curr_array[s+curr_frm_y][p+curr_frm_x] - avg) *
                                        (curr_array[s+curr_frm_y+2][p+curr_frm_x] - avg));
                           }
                   }
           }
   varver = sumversq/npels_fldmac;
   pverfld= varver/var;

   pverfld_slice[j] = pverfld;

   /*-----------------------------------------------------------------------------*/
   /*                 STEP. 3    Compare vertical frame and field correlation       */
   /*-----------------------------------------------------------------------------*/
```

```
if (pverfrm >= pverfld)
        {
        count_frmmac++;
        *type_codadap = 0;
        }
else
        {
        count_fldmac++;
        *type_codadap = 1;
        }

/*-------------------------------------------------------------------------*/
/*              STEP. 4    Copy 16x16 frame mac or field macs depending on correlation   */
/*-------------------------------------------------------------------------*/
if (*type_codadap == 1)  /* Field Mode */
        {
        for (k=0; k < 2; k++)   /* field0 followed by field 1 */
                {
                for(s = 0; s < mot_bl_size; s += 2)
                        {
                        for(p = 0; p < mot_bl_size; p++)
                                {
                                *adap_frm = curr_array[s+k+curr_frm_y][p+curr_frm_x];
                                adap_frm++;
                                }
                        }
                }
        }
else                    /* Frame Mode */
        {
        for(s = 0; s < mot_bl_size; s++)
                {
                for(p = 0; p < mot_bl_size; p++)
                        {
                        *adap_frm = curr_array[s+curr_frm_y][p+curr_frm_x];
                        adap_frm++;
                        }
                }
        }
  type_codadap++;
  type_mcadap++;
  type_ifdadap++;
  }/* end for j */

    }/* end for i */

}/* end frame_field */
```

We claim:

1. An apparatus for encoding digital video signals, comprising:

a means for receiving a digital video input signal comprising a succession of digital representations related to picture elements making up at least one frame of a video image, the frame comprising a plurality of interlaced fields;

a means for coding groups of digital representations related to frames of picture elements;

a means for coding groups of digital representations related to interlaced fields in the frames; and

a means responsive to the digital video input signal for producing a field frame coding type signal which directs a selected one, but not both, of the frame coding means or the field coding means to code the digital video input signal.

2. The encoding apparatus of claim 1, in which the fields comprise alternating horizontal scan lines of the frames.

3. An apparatus for compressing digital video signals, comprising:

a means for receiving a digital video input signal comprising a succession of digital signals representing picture elements which make up at least one frame of a video image, the frame comprising a plurality of interlaced fields;

a means for producing a signal relating to an estimate of the digital video input signal;

a means responsive to the digital video input signal and the estimate of the digital video input signal for producing an error signal;

a circuit means responsive to the error signal for determining frequency coefficients of the error signal;

a means for quantizing the frequency coefficients;

a means for scanning the quantized frequency coefficients in predetermined order for producing a succession of frequency coefficient signals in the predetermined order;

a variable word length encoder responsive to the succession of the frequency coefficient signals for producing a compressed video signal bit stream; and

a means responsive to the digital video input signal prior to compression for producing a coding type signal for controlling compression of the digital video input signals by causing a selected one, but not both, of frame encoding and field encoding to be applied to the digital video input signal.

4. The apparatus of claim 3, in which the means for producing a coding type signal controls the predetermined order of scanning produced by the scanning means.

5. The apparatus of claim 3, in which the means for producing a coding type signal controls the means for producing a signal relating to an estimate of the digital video input signal.

6. The apparatus of claim 3, in which the means for producing a coding type signal controls the quantizing means.

7. The apparatus of claim 6, in which the estimate of the digital input video signal is produced in response to motion vectors determined by a motion estimating means responsive to the digital video input signal and at least one representation of another video input signal.

8. The apparatus of claim 3, in which the estimate of the digital input video signal is a motion compensated estimate of the digital input video signal.

9. The apparatus of claim 8, in which there are at least two modes of performing motion compensation.

10. The apparatus of claim 9, in which there is a frame motion compensation mode.

11. The apparatus of claim 9, in which there is a field motion compensation mode.

12. An apparatus for encoding digital video signals, comprising:

a means for receiving digital video input signals representing frames of picture elements, the frames comprising fields of non-contiguous picture elements;

a means for producing motion compensated estimates of the digital video input signals;

a means for producing motion vectors in response to signals representing frames of picture elements; and

a means for producing motion vectors in response to signals representing fields of picture elements;

a means for selecting one of the motion vectors produced in response to signals representing frames of picture elements and the motion vectors produced in response to signals representing field of picture elements for input to the estimate producing means;

the estimate producing means producing estimates based upon the selection produced by the selection means.

13. An apparatus for decoding a compressed digital video signal, comprising:

a means for receiving a compressed digital video bit stream; and

a means responsive to a motion compensation type signal for selectively and adaptively performing motion compensated decoding of frames of the compressed video bit stream.

14. The apparatus of claim 13, in which the decoding means comprises:

an adaptive inverse scanning means responsive to a coding type signal.

15. The apparatus of claim 13, in which the decoding means comprises:

a means responsive to a motion compensation type signal and selectively responsive to frame motion vectors and field motion vectors for producing an adaptive motion compensated estimate of a decoded video signal;

a means responsive to the compressed digital video bit stream for producing a decoded estimate error signal; and

a means responsive to the adaptive motion compensated estimate and the estimate error signal for producing a decoded video signal.

16. The apparatus of claim 13, in which the decoding means comprises:

a means responsive to a coding type signal for adaptively dequantizing the compressed digital video bit stream.

17. The apparatus of claim 13, in which the decoding means comprises:

a means for receiving a compressed digital video signal comprising at least one DC coefficient representation related to the video signal;

a means for producing an estimated DC coefficient in response to a coding type signal; and

a means for producing a decoded DC coefficient

signal in response to the DC coefficient representation and the estimated DC coefficient.

18. An apparatus for encoding digital video signals, comprising:

a means responsive to digital input video signals comprising a succession of digital representations of picture elements making up at least one video picture frame having a plurality of fields;

a means for producing a compressed code relating to the input video signals; and

an adaptive frame/field coding selector responsive to the digital input video signal prior to compression for producing a coding type signal which controls the production of compressed code by causing a selected one, but not both, of frame encoding and field encoding to be applied to the digital video input signals.

19. The apparatus of claim 18, in which the coding selector comprises:

a means for adaptively quantizing the digital input video signals in response to the coding type signal.

20. The apparatus of claim 18, further comprising:

a means selectively responsive to frame motion vectors and field motion vectors for producing an adaptive motion compensated estimate of the digital input video signal; and

a means for coding a signal related to the difference between the digital input video signal and the adaptive motion compensated estimate.

21. The apparatus of claim 18, further comprising:

a means for adaptively scanning stored values related to the representations in response to the coding type signal.

22. The apparatus of claim 18, further comprising:

a means for converting a plurality of the digital representations into a DC coefficient and a plurality of AC coefficients;

a means for adaptively producing a predicted DC coefficient in response to the coding type signal; and

a means responsive to the DC coefficient produced by the converting means and to the predicted DC coefficient for encoding the DC coefficient.

23. An apparatus for encoding digital video signals, comprising:

a means for receiving a digital video input signal comprising a succession of digital representations of picture elements making up at least one video frame, the frame comprising a plurality of fields;

a means for selectively encoding groups of digital representations in the input signal relating to one of frames and fields; and

a means responsive to the video input signal prior to encoding to ascertain a predetermined characteristic present in the input signal prior to encoding for producing a field/frame encoding type signal which directs the encoding means to encode a selected one, but not both, of the groups of digital representations relating to frames and fields in the input signal.

24. The apparatus of claim 23, in which the means for encoding comprises a means for selectively transforming first groups of digital representations relating to one of frames of picture elements and fields of picture elements into second groups of digital representations relating to one of frames of picture elements and fields of picture elements.

25. The apparatus of claim 24, in which the means for selectively transforming comprises a means for transforming groups of digital representations in the input video signal into groups of frequency coefficients related to the input video signal.

26. The apparatus of claim 24, in which the means for selectively transforming comprises a means for adaptively performing a discrete cosine transform of the input video signal in response to the frame/field encoding type signal.

27. The apparatus of claim 24, in which the predetermined characteristic comprises a correlation between predetermined portions of the digital video input signal.

28. The apparatus of claim 24, in which the encoding means comprises a means for performing motion compensation.

29. The apparatus of claim 24, in which the encoding means comprises a means for quantizing the digital video input signal.

30. The apparatus of claim 24, in which the encoding means comprises a means for performing variable word length encoding.

31. An apparatus for encoding digital video signals, comprising:

a means for receiving a digital video input signal comprising a succession of digital representations of picture elements making up at least one video frame, the frame comprising a plurality of fields;

a means for performing adaptive motion compensated encoding of groups of digital representations in the input signal relating to one of frames and fields in the input signal; and

a means responsive to the video input signal prior to encoding for producing a motion compensation type signal for controlling the adaptive motion compensated encoding means.

32. An apparatus for encoding digital video signals, comprising:

a means for receiving digital video input signals; and

a means for performing variable word length encoding adaptively in response to the video input signals.

33. The apparatus of claim 32, further comprising:

a means of transforming the digital video input signals into transform coefficients;

the encoding means being responsive to the transform coefficients for performing adaptive variable word length encoding.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

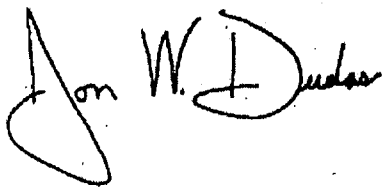
PATENT NO. : 5,227,878
DATED : July 13, 1993
INVENTOR(S) : Atul Puri and Rangarajan Aravind

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 90,
Line 38, "compressed video bit stream." should read as -- compressed digital video bit stream and fields of the compressed video bit stream. --.

Signed and Sealed this

Twenty-fifth Day of October, 2005



JON W. DUDAS
Director of the United States Patent and Trademark Office

US005500678A

United States Patent [19]

Puri

[11] Patent Number: 5,500,678

[45] Date of Patent: Mar. 19, 1996

[54] **OPTIMIZED SCANNING OF TRANSFORM COEFFICIENTS IN VIDEO CODING**

[75] Inventor: **Atul Puri**, Riverdale, N.Y.

[73] Assignee: **AT&T Corp.**, Murray Hill, N.J.

[21] Appl. No.: **215,532**

[22] Filed: **Mar. 18, 1994**

[51] **Int. Cl.**[6] **H04N 7/50**

[52] **U.S. Cl.** **348/408**; 348/405

[58] **Field of Search** 348/408, 405, 348/419; H04N 7/133, 7/50

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,227,878 7/1993 Puri 348/416

Primary Examiner—Howard W. Britton

Attorney, Agent, or Firm—Eugene S. Indyk; Mark K. Young

[57] **ABSTRACT**

An improved scanning apparatus and method which allows for increased coding efficiency over conventional zigzag scans is disclosed. The invention advantageously allows for total compatibility with the MPEG-1 standard, and accomodates video sequences which may be composed of both the progressive and interlaced format frames.

19 Claims, 7 Drawing Sheets

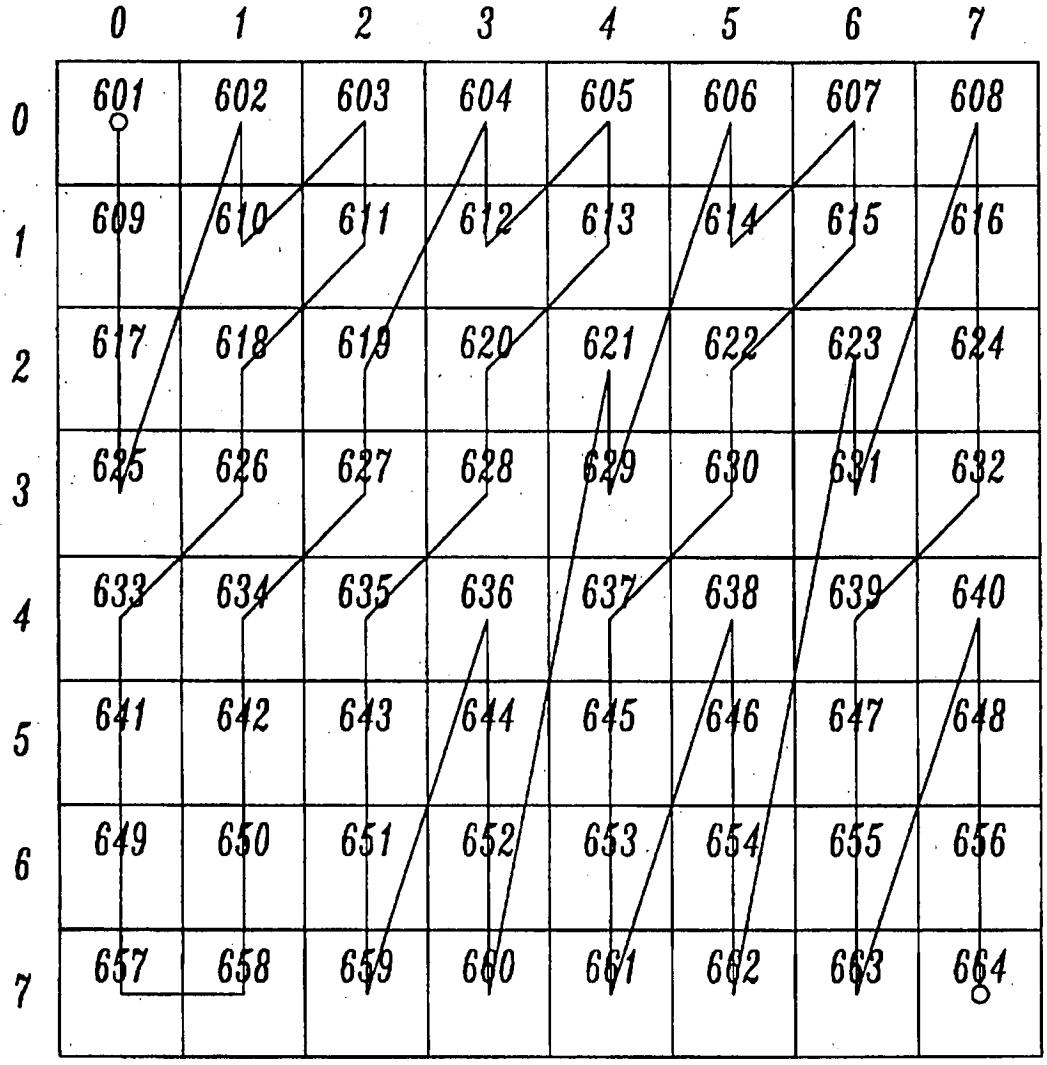


FIG. 1
(PRIOR ART)

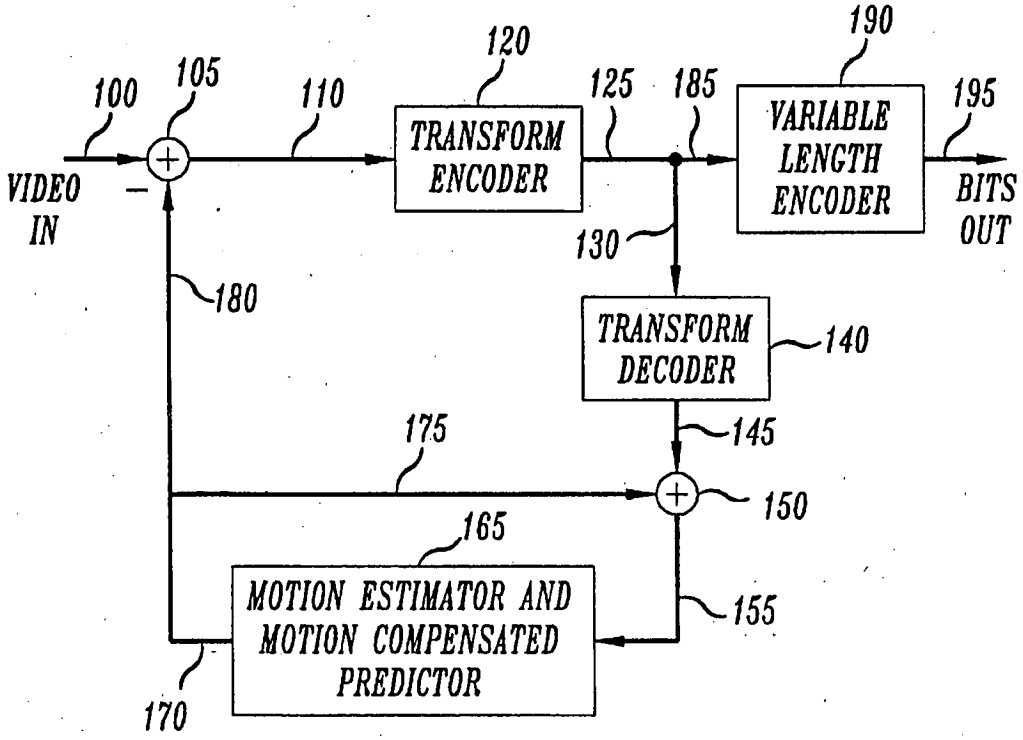


FIG. 2
(PRIOR ART)

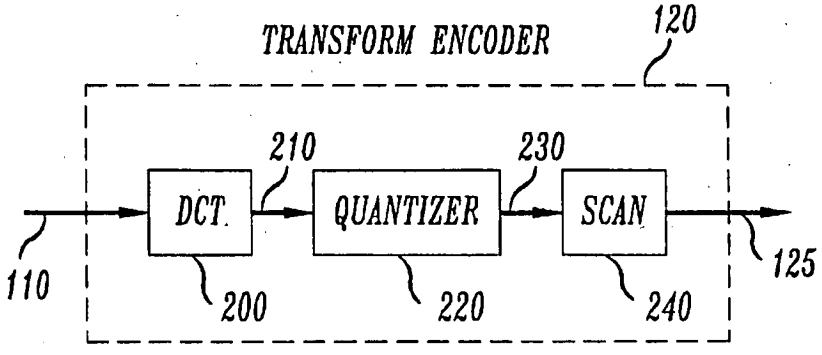


FIG. 3
(PRIOR ART)

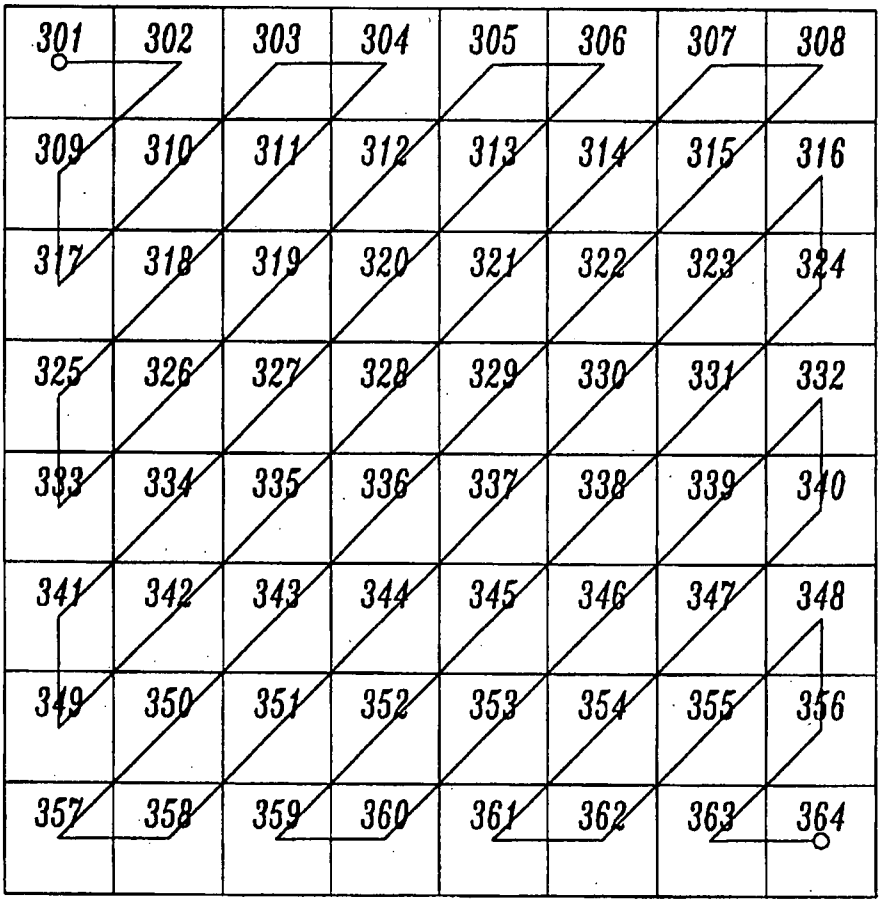


FIG. 4
(PRIOR ART)

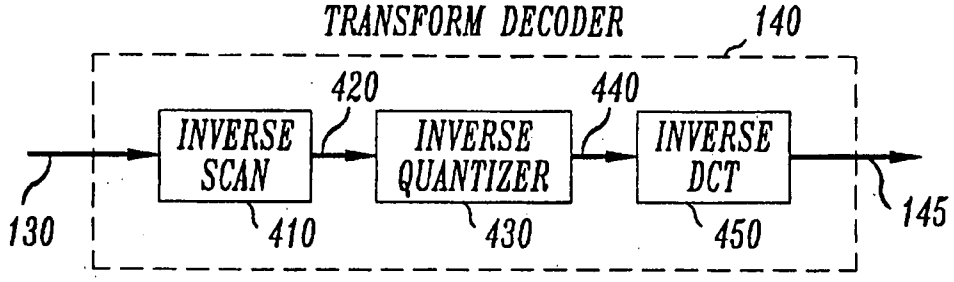


FIG. 5
(PRIOR ART)

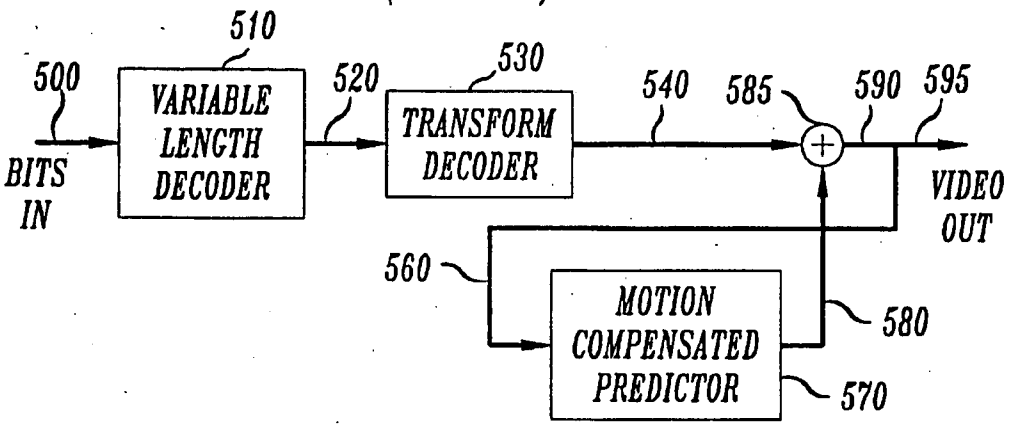


FIG. 6

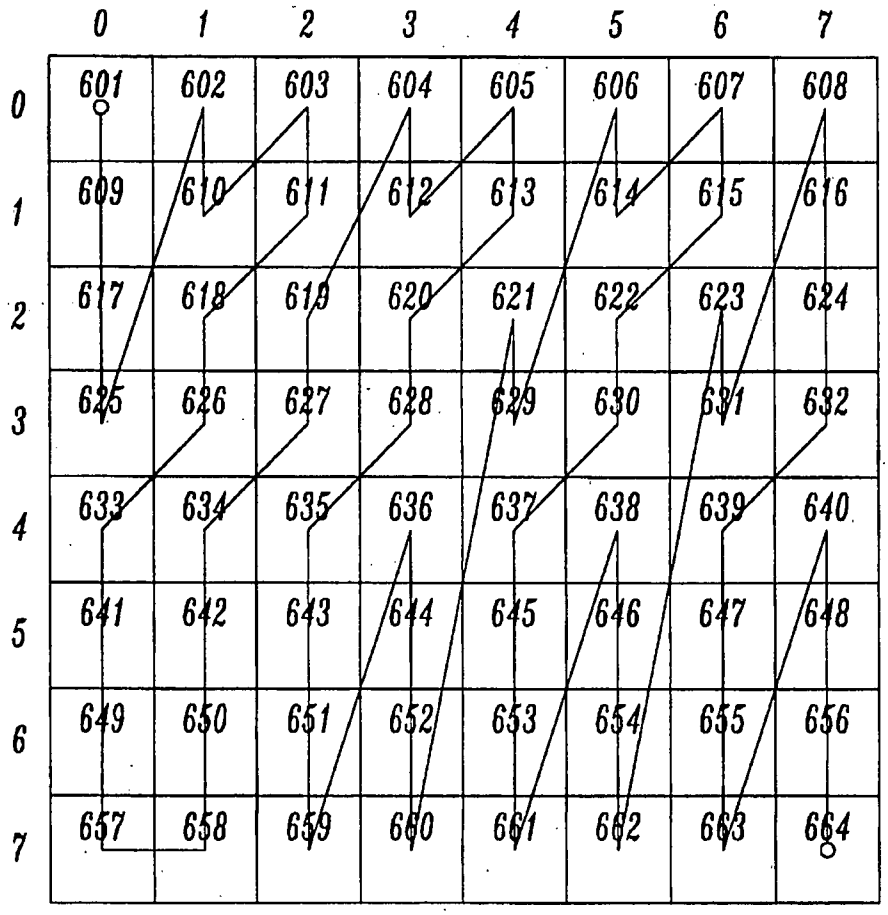


FIG. 7

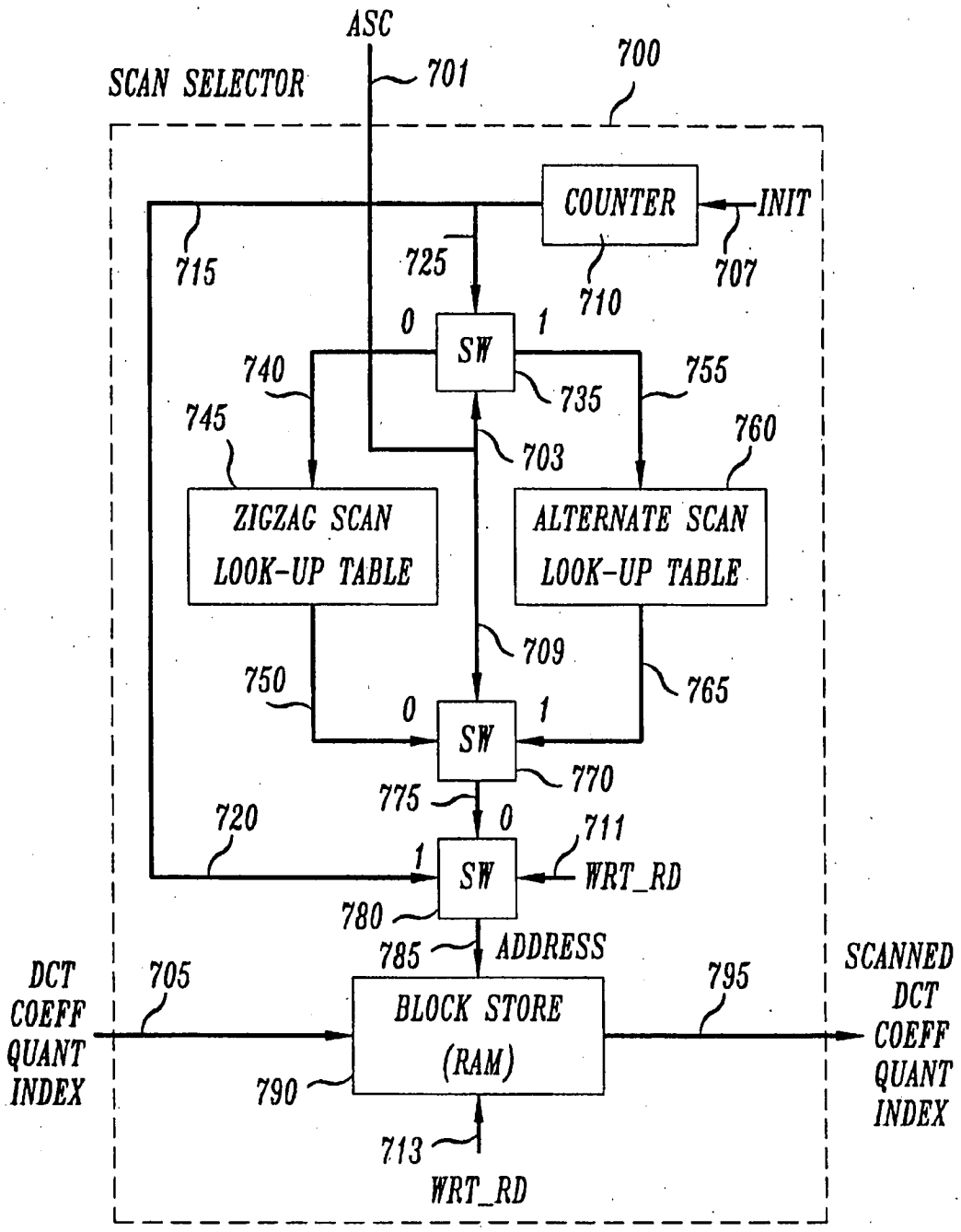

ASC
SCAN SELECTOR
701
700
COUNTER
INIT
715
725
710
707
0
1
SW
740
735
755
745
760
703
ZIGZAG SCAN
LOOK-UP TABLE
ALTERNATE SCAN
LOOK-UP TABLE
709
750
765
0
1
SW
770
775
0
720
711
1
SW
WRT_RD
780
785
ADDRESS
DCT
COEFF
QUANT
INDEX
705
BLOCK STORE
(RAM)
795
SCANNED
DCT
COEFF
QUANT
INDEX
790
713
WRT_RD

FIG. 8

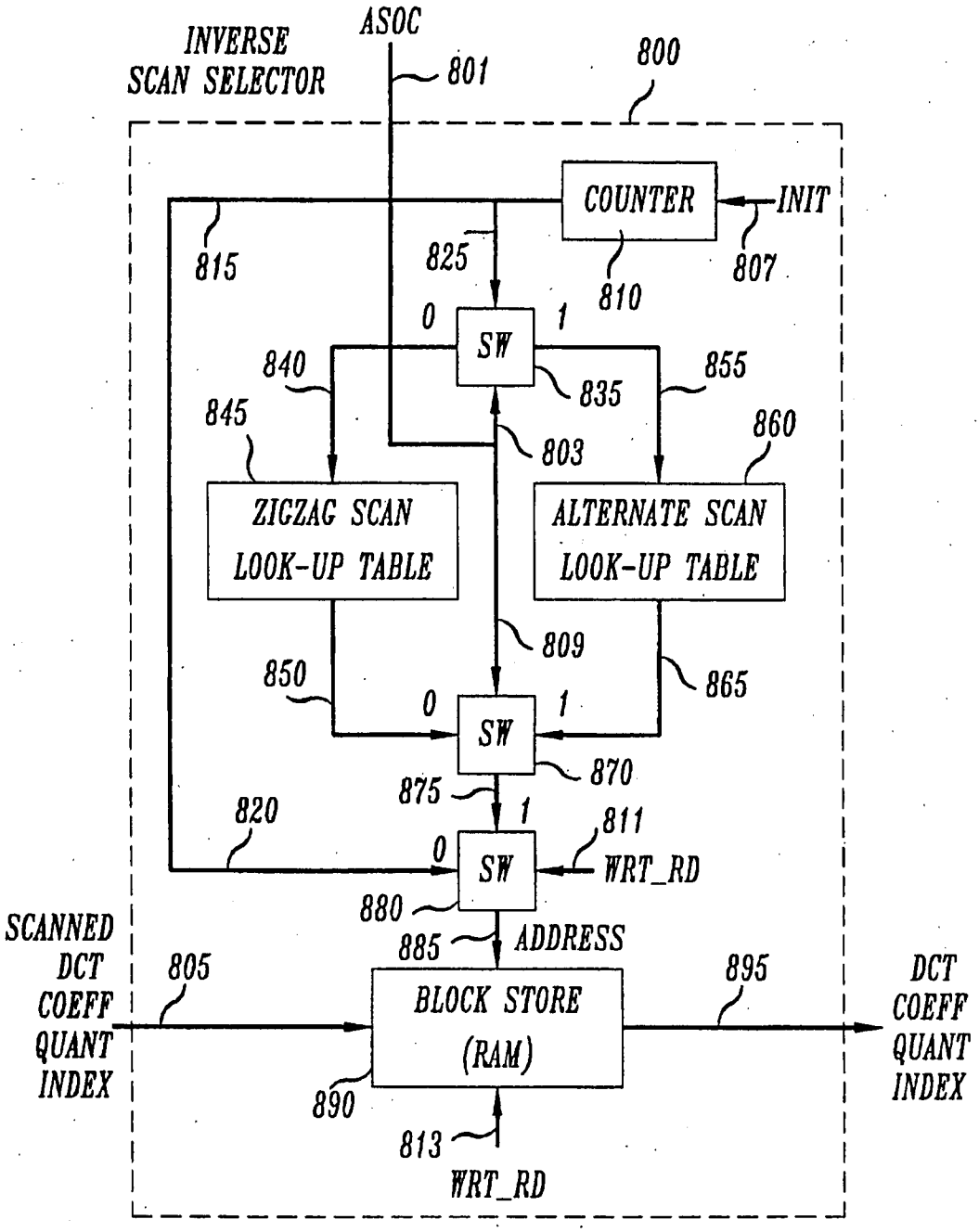

INVERSE SCAN SELECTOR
ASOC
801
800
COUNTER
INIT
815
825
807
810
0
1
SW
840
855
835
845
860
803
ZIGZAG SCAN LOOK-UP TABLE
ALTERNATE SCAN LOOK-UP TABLE
809
850
865
0
1
SW
870
875
820
1
811
0
SW
WRT_RD
880
SCANNED DCT COEFF QUANT INDEX
885
ADDRESS
805
895
BLOCK STORE (RAM)
DCT COEFF QUANT INDEX
890
813
WRT_RD

FIG. 9

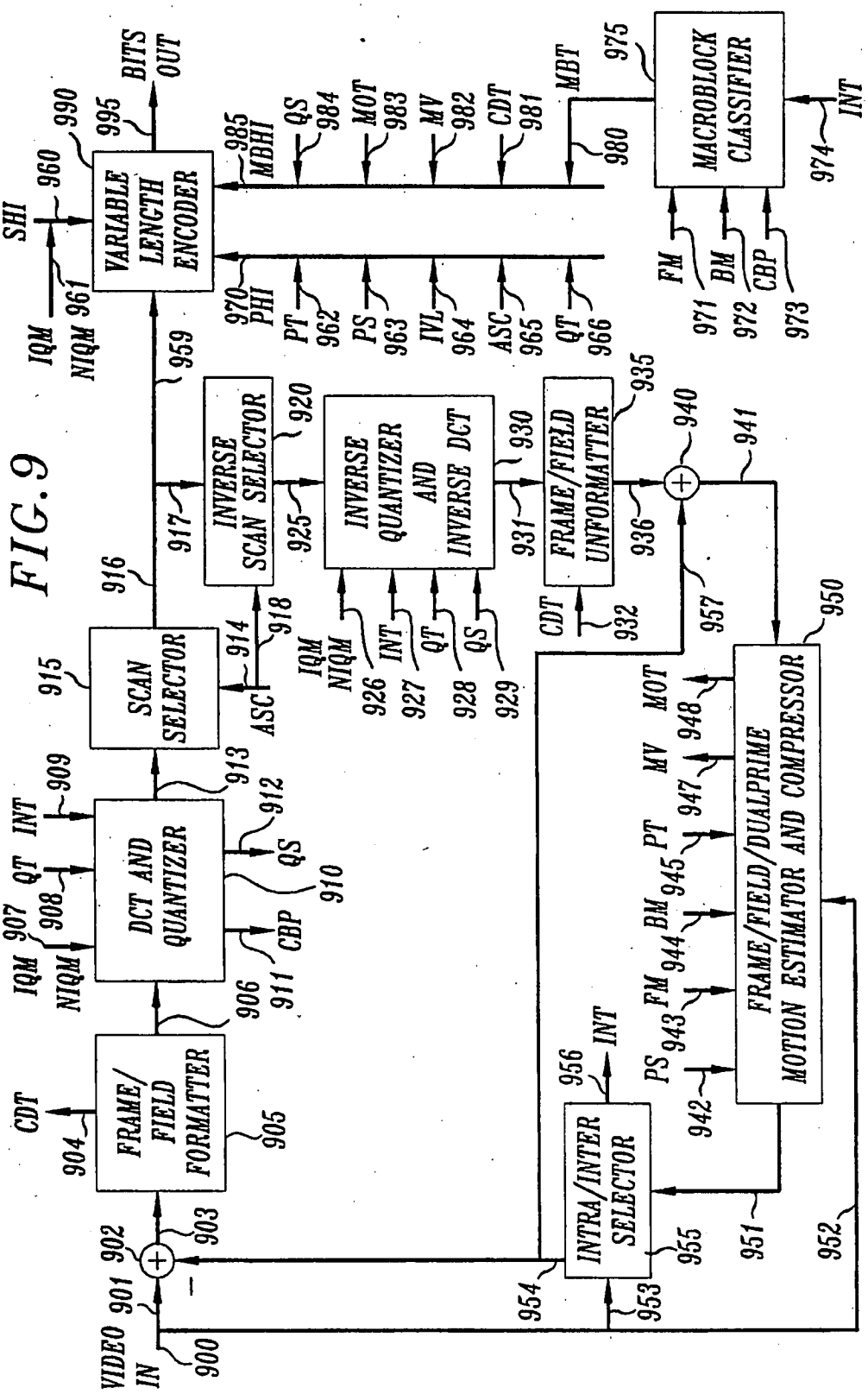
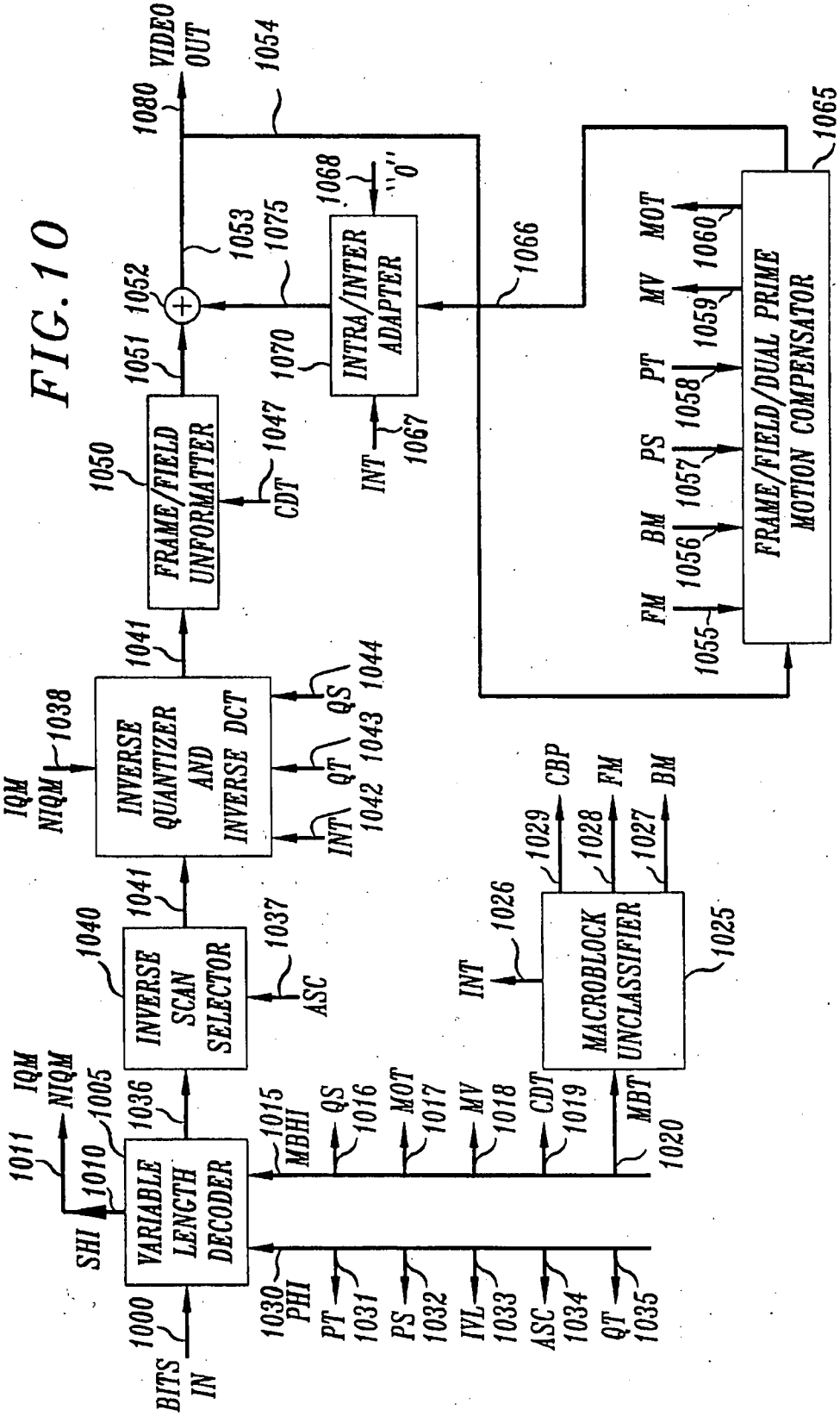

VIDEO IN
900
901
902
903
FRAME/ FIELD FORMATTER
905
CDT
904
DCT AND QUANTIZER
906
910
911
912
913
907
908
909
IQM
NIQM
QT
INT
CBP
QS
SCAN SELECTOR
915
914
916
ASC
INVERSE SCAN SELECTOR
917
918
920
INVERSE QUANTIZER AND INVERSE DCT
925
930
926
927
928
929
IQM
NIQM
INT
QT
QS
FRAME/FIELD UNFORMATTER
931
935
936
940
941
CDT
932
957
950
INTRA/INTER SELECTOR
953
954
955
956
INT
FRAME/FIELD/DUALPRIME MOTION ESTIMATOR AND COMPRESSOR
942
943
944
945
947
948
PS
FM
BM
PT
MV
MOT
951
952
VARIABLE LENGTH ENCODER
959
960
961
SHI
IQM
NIQM
970
PHI
962
PT
963
PS
964
IVL
965
ASC
966
QT
971
FM
972
BM
973
CBP
974
MACROBLOCK CLASSIFIER
975
980
981
982
983
984
985
MBT
CDT
MV
MOT
QS
MBHI
990
995
BITS OUT
INT

FIG. 10


BITS IN
1000
VARIABLE LENGTH DECODER
SHI
1010
IQM NIQM
1011
1005
1036
INVERSE SCAN SELECTOR
1040
ASC
1037
1041
INVERSE QUANTIZER AND INVERSE DCT
IQM NIQM
1038
INT QT QS
1042 1043 1044
1041
FRAME/FIELD UNFORMATTER
1050
CDT
1047
1051 1052
1053 1075
1070
INTRA/INTER ADAPTER
INT
1067
1068
"0"
1066
1080 VIDEO OUT
1054
1065
FRAME/FIELD/DUAL PRIME MOTION COMPENSATOR
FM BM PS PT MV MOT
1055 1056 1057 1058 1059 1060
MACROBLOCK UNCLASSIFIER
1025
INT
1026
CBP FM BM
1029 1028 1027
MBT
1020
CDT
1019
MV
1018
MOT
1017
QS
1016
MBHI
1015
PHI PT PS IVL ASC QT
1030 1031 1032 1033 1034 1035

OPTIMIZED SCANNING OF TRANSFORM COEFFICIENTS IN VIDEO CODING

TECHNICAL FIELD

This invention relates to coding of video signals. More particularly, this invention relates to optimized scanning of transform coefficients for improved coding efficiency.

BACKGROUND

Transform coding is an efficient image compression scheme that typically involves segmenting an image into blocks of pixels, taking discrete cosine transforms ("DCTs") of the blocks of pixels to obtain blocks of DCT coefficients, quantizing these coefficients, and coding the quantized coefficients by an entropy coder. Interframe coding schemes utilizing motion compensation and transform coding of motion compensated interframe differences, by taking DCTs of blocks of difference pixels, quantizing the DCT coefficients and entropy coding the quantized DCT coefficients, may also be employed.

Interframe coding employing motion compensation and DCT coding has become widely recognized as a particularly efficient coding scheme for video compression and forms the core of the Comité Consulatif International Télégraphique et Téléphonique Recommendation H.261-Video Codec for Audiovisual Services at 64 Kbit/s, Geneva, August, 1990 ("CCITT H.261") and the Motion Pictures Expert Group Phase 1 ("MPEG-1") video compression standards. The MPEG-1 standard is set forth in International Standards Organization ("ISO") Committee Draft 11172-2, "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to 1.5 Mbits/s," November, 1991. The CCITT H.261 standard primarily addresses coding of video conferencing scenes of Common Intermediate Format resolution at bitrates of 64 kbit/s to 2 Mbit/s; the MPEG-1 standard can be efficiently used for coding all types of video scenes in 1 to 2 Mbit/s range. The MPEG-1 standard incorporates additional features, for example, group-of-pictures and motion compensated bidirectional prediction. However, because the MPEG-1 standard was initially applied to progressive format video, it is not optimized for coding of interlaced video conforming to the Comité Consultatif International des Radiocommunications Recommendation 601 ("CCIR 601") format which is similar to that used for broadcast and cable television. The CCITT H.261 and MPEG-1 coding standards specify that the DCT coefficients must be ordered before they are encoded. This ordering of DCT coefficients prior to encoding is called "scanning." For progressive format video, a conventional zigzag scan of DCT coefficients mentioned in the standards allows for ordering of the coefficients relative to their significance from low frequency to high frequency which results in events that can be efficiently coded by a two-dimensional variable length coder. Unfortunately, conventional zigzag scans do not allow for efficient coding of interlaced format video signals.

One approach to video coding using alternatives to the zigzag scan is discussed in U.S. Pat. No. 5,227,878 by A. Puri et al. which refers to adaptive coding on a macroblock basis using frame or field coding.

SUMMARY

Applicant has developed a novel scanning apparatus and method which allows for increased coding efficiency over conventional zigzag scans. The invention advantageously allows for total compatibility with the MPEG-1 standard, and accomodates video sequences which may be composed of both the progressive and interlaced format frames.

The discussion in this Summary and the following Brief Description of the Drawings, Detailed Description, and drawings only deals with examples of this invention and is not to be considered in any way a limitation on the scope of the exclusionary rights conferred by a patent which may issue from this application. The scope of such exclusionary rights is set forth in the claims at the end of this application.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a simplified block diagram of an illustrative interframe motion compensated transform encoder.

FIG. **2** is a block diagram of some details of an illustrative transform encoder.

FIG. **3** shows the order in which DCT coefficients are scanned using a zigzag scan conforming to the MPEG-1 and CCITT H.261 standards.

FIG. **4** is a block diagram of some details of an illustrative transform decoder.

FIG. **5** is a simplified block diagram of an illustrative interframe motion compensated transform decoder.

FIG. **6** shows the order in which DCT coefficients are scanned in accordance with the invention.

FIG. **7** is a simplified block diagram of a forward scan operation with scan selection, arranged in accordance with the invention.

FIG. **8** is a simplified block diagram of an inverse scan operation that uses scan selection at an encoder, arranged in accordance with the invention.

FIG. **9** is a simplified block diagram of an illustrative encoder employing an alternative to conventional zigzag scans and a scan adaptation in accordance with the invention.

FIG. **10** is a simplified block diagram of an illustrative decoder employing an alternative to conventional zigzag scans and a scan adaptation in accordance with the invention.

DETAILED DESCRIPTION

FIG. **1** is the simplified block diagram of an illustrative generalized interframe encoder. Video frames enter one-by-one at bus **100**. Subtracter **105** differences an input frame of video on bus **100** with a prediction frame available on bus **180**. The resulting prediction error signal appears on bus **110** and is transform encoded by the transform encoder **120** which outputs quantized and scanned transform coefficients on bus **125** which are simultaneously available on buses **130** and **185**. Variable length encoder **190** entropy encodes these coefficients and multiplexes with other encoded overhead (not shown) and outputs a stream of bits on bus **195** for transport over a transmission channel or to a storage disk for eventual delivery to a decoder. The coefficients also get fed back on bus **130** to transform decoder **140** which outputs, on bus **145**, an approximation of the prediction error signal on bus **110**. A prediction signal available on bus **175** is summed to an approximated prediction error signal on bus **145** in adder **150**, resulting in locally reconstructed output on bus **155** which is fed to motion estimator and motion compensated predictor **165** whose output is the prediction signal at bus **170**. Those skilled in the art will appreciate that operation of the motion estimator and motion compensated predictor need not be described in detail here. However, it is

necessary to explicitly state that motion compensated estimator and predictor **165** may not only compute motion estimates required for forward prediction, as in the CCITT H.261 standard, but also compute motion estimates required for forward and backward predictions, as in the MPEG-1 standard. The prediction signal at bus **170** is simultaneously made available at bus **180** to one of the inputs to the subtracter **105** as well as at bus **175** to one of the inputs to the adder **150**.

FIG. **2** shows details of transform encoder **120** shown in FIG. **1**. The input signal is segmented into two-dimensional non-overlapping blocks of pixels, typically, into 8×8 blocks which are fed to bus **110** and are transformed to corresponding 8×8 blocks of coefficients by the DCT 200. The coefficients are then sent over bus **210** to quantizer **220**. The quantized DCT two-dimensional coefficient blocks on bus **230** are then scanned to a one dimensional sequence of coefficients with the property that the more important coefficients are scanned first and the runs of coefficients quantized to zero are maximized. This is accomplished by scan **240**. The scanned quantized DCT coefficients are output on bus **125** and are now ready for efficient entropy coding. Depending on the signal at the input **110**, this type of transform encoder can be used to encode the original input signal or the prediction error signal. The conventional technique for scanning blocks of two-dimensional quantized DCT coefficients into a one dimensional sequence of coefficients for every block is called the zigzag scan.

FIG. **3** shows the order in which the quantized transform coefficients forming a two dimensional block are scanned to produce a one dimensional sequence of quantized transform coefficients when performing the zigzag scan. Starting with the DC coefficient located at **301**, low frequency coefficients located at **302**, **309**, **317**, **310**, **303**,**304**, **311**, **318**, **32S**, **333**,**326**, **318**, **312**, **305**,**306**, **313**,**320**, **327**,**334**, **341** and so on along the zigzag scan path until the last coefficient located at **364** is scanned. The output of the scan on bus **125** consists of pairs of runs of zero quantized coefficients followed by the amplitude of next non- zero coefficient. These pairs are entropy encoded in a variable length encoder **190**.

The details of the transform decoder **140** employed in FIG. **1** are shown in FIG. **4**. This decoder reverses the operations performed by the transform encoder **120**. Bus **130** at the input to the transform decoder consists of pairs of runs of zero quantized coefficients followed by the amplitude of next non-zero coefficient. Inverse scan **410** restores a two-dimensional block of quantized DCT coefficients by following the sequence of run, amplitude pairs available on bus **130**, representing the output of the scan **240**. These two-dimensional blocks of quantized coefficients are input to inverse quantizer **430** via bus **420**. The output of quantizer **440** is inverse transformed in inverse DCT 450 to produce two-dimensional blocks of pixels which are output on bus **145**. Then, inverse scan **410** employs the same zigzag scan order of **300** as the forward scan, which basically undoes the scan operation.

FIG. **5** is a simplified block diagram of an illustrative generalized interframe decoder. An incoming stream of bits from a transport medium, such as a transmission channel or a storage disk, is input over bus **500** to variable length decoder **510** which demultiplexes and decodes the bit patterns codes representing different overhead (not shown) and the encoded events representing quantized DCT zero coefficient runs and non-zero amplitudes. The latter are output on bus **520** and fed to transform decoder **530**. The transform decoder **530** is an exact replica of the transform decoder **140** in the local decoding loop of the generalized interframe encoder of FIG. **1**. As will be appreciated by those skilled in the art, in the absence of errors on the transport medium, the input on bus **520** of the transform decoder **530** and the output of transform decoder on bus **540**, carry the exact corresponding signals as input on bus **130** of the transform decoder **140** (FIG. **1**) and its output on bus **145** (FIG. **1**). Next, the decoded blocks of pixels representing approximated prediction error signal on bus **540** are summed in the adder **585** with the prediction signal **580** at the output of motion compensated predictor **570**. The output of the adder **585** appears on the bus **590** and represents the decoded video at bus **595**. The same output is also fed back to the motion compensated predictor **570** via bus **560** and the prediction generated by **570** is output on bus **580** and forms one input to the adder **585**. Those skilled in the art will appreciate that operation of the motion compensated predictor need not be described in detailed herein. As above, it can produce not only the forward prediction, as in the CCITT H.261 standard, but also the forward and the backward predictions, as in MPEG-1 standard. Thus, the operation of motion compensation loop of the local encoder and decoder are practically identical, except that the decoder does not compute motion vectors since they are extracted from the decoded bitstream.

The zigzag scan **300** shown in FIG. **3** is only optimal when scanning blocks of pixels containing relatively flat areas or non-oriented texture. Since many simple video scenes contain a relatively high percentage of low detail areas containing directionally insensitive texture, and only a small percentage of blocks contain edges or directionally aligned structure, the zigzag scan works well overall. However, in many applications such as with normal television, interlaced format video that provides a higher spatio-temporal resolution while striking a compromise between the vertical and the temporal resolution, must be used. In interlaced format video, each frame consists of two fields each of half vertical resolution of a frame but occurring at twice the frame rate. Unlike the simple, also known the progressive, format used in video conferencing, this type of format results in many artificial edges due to interlaced nature of each frame. Thus, the alternative scan in accordance with the invention works better than conventional zigzag scans since it takes into account such characteristics of interlaced video.

FIG. **6** shows the order in which the quantized transform coefficients forming a two-dimensional block are scanned to produce a one-dimensional sequence of quantized transform coefficients when performing an alternative to conventional zigzag scans in accordance with the invention. Starting with the DC coefficient located at **601**, low frequency coefficients located at **609**, **617**,**625**,**602**, **610**, **603**, **611**, **618**, **626**, **633**, **641**, **649**, **657**, **658**, **650**, **642**, **634**, **627**, **619**, **604** and so on along the alternate scan path, as shown, until the last coefficient located at **664** is scanned.

FIG. **7** is a simplified block diagram of an illustrative example of a scan selector **700** that allows selection between the conventional zigzag scan and the new alternative scan described above, in accordance with the invention. Quantized DCT coefficient blocks corresponding to blocks of pixels enter at bus **705**. Counter **710** is initialized by control signal "init" on bus **707**, the count on bus **715** at the output of the counter is made available on bus **720** which feeds the switch **780**. At this time the "wrt_rd" control signal at switch **780** is set to "1" connecting the bus **720** to bus **785** and the incoming quantized DCT coefficient is stored in block store **790** at address on bus **785**. This process is

repeated until all coefficients of a block are stored in the block store **790**.

Next, depending on the value of the alternate scan control signal "asc" on bus **701**, either the zigzag scan lookup table **745** or the alternate scan lookup table **760** is utilized. Counter **710** is initialized by control signal "init" on line **707**. The output of the this counter on bus **715** provides a count of the coefficient being scanned. This signal is now available on bus **725** and applied to switch **735**. Now, assume that control signal "asc" is set to "1" selecting the alternate scan. Switch **735** connects bus **725** with bus **755**. Depending on the coefficient being processed, using look up table **760**, a corresponding address along the scan appears on bus **765** through switch **770** to bus **775**. Alternatively, if control signal "asc" is set to "0", the conventional zigzag scan is selected. Switch **735** connects bus **725** to bus **740** and corresponding contents of lookup table **745** appear on bus **750** and appear through switch **770** on bus **775**. At this time, control signal "wrt_rd" on line **711** is set to "0" and, thus switch **780** connects bus **775** to bus **785**. The address on bus **785** is now applied to the block store to retrieve the next coefficient along the one-dimensional scan. This process is repeated until all coefficients in a two-dimensional coefficient block in block store are ordered as a one dimensional sequence of coefficients facilitating the maximization of zero runs of coefficients and non-zero amplitude values along the scan. This process is repeated for every incoming coefficient block on line **705**.

FIG. **8** is a simplified block diagram of an illustrative example of an inverse scan selector that allows selection between the conventional zigzag scan and the alternate scan for the purpose of reversing the scanning operation performed by the scan selector **700**. Inverse scan selector **800** is similar in form and operation as scan selector **700**, with the exception that some control signals are interpreted differently. Here, an incoming 1-dimensional scanned block of quantized DCT coefficients enters at bus **805**, the counter **810** is initialized by the control signal "init" on bus **807** and the count on bus **815** at the output of the counter is made available on bus **820** which feeds the switch **870**. At this time the "wrt_rd" control signal at switch **880** is set to "0" connecting the output of bus **820** to bus **885**. This process is repeated until all coefficients of a block are stored in b**10**ck store **890**.

Next, depending on the value of the alternate scan control signal "asc" on bus **801**, either the zigzag scan lookup table **845** or the alternate scan lookup table **860** is utilized. Counter **810** is initialized by control signal "init" on line **807**. The output of the this counter on bus **815** provides count of the coefficient being scanned. This signal is available on bus **825** and applied to switch **835**. Now, assume that control signal "asc" is set to "1" selecting the alternate scan. Switch **835** now connects bus **825** with bus **855**. Depending on the coefficient being processed, using look up table **860**, a corresponding address along the scan appears on bus **865** and through the switch **870** to bus **875**. Alternatively, if control signal "asc" is set to "0", the zigzag scan would be selected. Switch **835** connects bus **825** to bus **840** and corresponding contents of lookup table **845** appear on bus **850**, and appear through switch **870** on bus **875**. At this time, control signal "wrt_rd" on line **811** is set to "1" and thus, switch **880** connects the bus **875** to bus **885**. The address on bus **885** is now applied to the block store to retrieve the next coefficient along the two-dimensional block. This process is repeated until all the coefficients in a one-dimensional sequence of coefficient in block store are ordered as a two-dimensional block of coefficients. This process is repeated for every incoming coefficient sequence on line **805**.

As described so far, this illustrative example of the invention could be used, for example, to adapt the choice of scan either on a block, a macroblock, a slice or on a picture basis. However, the overhead necessary to communicate scan selection information to the decoder may not be insignificant compared to the savings if scan selection on a macroblock or block basis is adopted. Furthermore, the encoder complexity may increase in case, a posterior selection of scan based on bit counting on a block, macro block or a slice basis is performed. Also, as stated earlier, one of the primary reasons for bits savings in scan selection is linked to the nature of the video source. For interlaced sources alternative scan outperforms the zigzag scan, and since a video sequence may be composed of interlaced and progressive frames, a picture based scan selection between the zigzag scan an the alternate scan has be chosen as a compromise in the MPEG-2 video standard.

Various control/data signals necessary for understanding the operation of some of the elements in FIG. **9** and FIG. **10** are listed below:

- pt: picture types allowed are I, P, B as described by MPEG-1.
- ps: picture structure (Frame or field).
- ivl: vlc selection for a picture.
- qt: quantizer type table selected for a picture.
- asc: alternate scan selected for a picture.
- mv: motion vector/s of a macro block mot motion type (Frame or field or dual prime or submacrob**10**ck) for a macro block.
- cdt: coding type (Frame or field) for a macroblock.
- fro: forward motion direction of a macroblock.
- bin: backward motion direction of a macroblock
- qs: quantizer scale chosen for a macroblock.
- cbp: coded block pattern of a macroblock in the intra/inter coding mode for a macroblock.
- shi: sequence header information.
- phi: picture header information.
- mbhi: macroblock header information
- iqm: intra quantizer matrix for a sequence or part of a sequence
- niqm: nonintra quantizer matrix for a sequence or part of a sequence

FIG. **9** is a simplified block diagram of an encoder employing an alternate scan and a scan adaptation in accordance with the invention. The structure of this encoder is similar in form and operation to the generalized interframe encoder shown in FIG. **1**. Although the encoder in FIG. **9** shows several extra operations and a few low level data/control signals, the portions of the circuit not directly related to our invention are not covered in great detail herein as they will be readily apparent to those skilled in the art.

Video enters at bus **900** and is input via bus **901** to subtracter **902** where it is differenced with its prediction signal at bus **958** and the difference appears on bus **903** and enters frame/field formatter **905** where it is retained as frame b**10**cks or converted to field blocks on a macroblock basis. The output of the frame/field formatter appears on bus **906** and undergoes DCT transform and quantization in DCT and quantizer **910**. For quantization, on a picture basis, a selection from two quantization tables is allowed by the "qt" signal. The intra or inter quantization characteristics are selected by the "in the" signal on a macrob**10**ck by

macrob10ck basis and corresponding quantization weighting matrix "iqm" and "niqm" are applied. The output of operations in DCT and quantizer **910** is quantized DCT coefficients on bus **913**, and signals representing coded b10cks in a macro blo0ck 'cbp' on line **911** and quantizer step size "qs" on line **912**. Next, quantized DCT coefficients enter scan selector **915**, which is similar in form and operation to scan selector **700** (FIG. 7) which allows selection on a picture basis from among the zigzag and the alternate scan. The zigzag scan is selected for progressive frames and the alternate scan for interlaced pictures. The signal controlling the scan selection is "asc" on bus **914** which is discussed above. The scanned quantized coefficients appear on bus **916** at the output of scan selector **915** and are variable length encoded in variable length encoder **990** and multiplexed with encoded data representing various types of overhead. Such overhead information could include, for example, sequence header information "shi" at bus **960**, picture header information "phi" at bus **970** and macroblock header information "mbhi" at bus **985** into MPEG-2 bitstream Ibrmat for storage or transmission. The sequence header information on bus **960** consists of intra and inter quantization weighting matrices on line **961**. The picture header information on bus **970** carries picture type "pt" on line **962**, picture structure "ps" on line **963**, intra VLC selection "ivl" on line **964**, alternate scan selection "asc" on line **965** and quantizer type "qt" on line **966**. The macroblock header information on bus **985** carries quantizer step "qs" on line **984**, motion type "mot" on line **983**, motion vector "mv" on line **982**, coding type "cdt" on line **981**, and macroblock type on line **980**. The macroblock type is obtained from classifier **975** as a result of input control signals, forward motion "fm" on line **97 1**, backward motion "bm" on line **972**, coded block pattern "cbp" on line **973** and intra/inter decision "in the" on line **974**. The bitstream representing coded data and overhead appears on bus **995** for transport or storage.

The local decoding loop at the encoder receives scanned quantized coefficients on bus **917** simultaneously with variable word length encoder **990**. These coefficients are inverse scanned in inverse scan selector **920** which requires alternate scan control "asc" to decide which scan to use and output on bus **925** to inverse quantizer and inverse DCT **930**. All but one of the control signals input or output from the DCT and quantizer **910** are input control signals to inverse quantizer and inverse DCT. Linc **926** carries quantizer matrices, line **927** carries "in the", line **928** carries "qt", and line **929** carries "qs". The output of inverse DCT **930** are reconstructed prediction error blocks on bus **931** and undergo frame/field unformatting in field/frame unformatter **935** to invert the formatting operation of **905**.

To the reorganized prediction error blocks on bus **936**, motion compensated prediction on bus **957** is summed in adder **940** resulting in local reconstructed output on bus **941**, which in turn is used for calculation of motion compensated prediction in **950**. There are several choices regarding motion compensation methods in **950** the best one is indicated by motion type "mot" on line **948**, the corresponding motion vectors "my" are output on **947**. Several control signals such as picture structure on line "**942**", forward motion "fm" on line **943**, backward motion "bm" on line **944** and picture type "pt" on line **945** control the operation of motion estimator and motion compensator **950**, whose output is the motion compensated prediction signal on bus **951**. This signal is compared with original on bus **953** to decide which would be more efficient to code in intra/inter selector **955** which yields a control signal "in the" on linc **956** and prediction signal on bus **954** which appears simultaneously on buses **957** and **958**.

The decoder of FIG. 10 performs a subset of operations performed by the local decoder in the encoder shown in FIG. **9**. A bitstream at bus **1000** feeds a variable length decoder **1040**, where depending on the context, code words representing control data forming sequence header information, macroblock header information, DCT coefficient data are identified and their values are decoded. The sequence header information "shi" is available on bus **1010** and comprises quantizer matrices, iqm and niqm on line **1011**. The picture header information "phi" is available on bus **1030** and comprises picture type "pt" on line **1031**, picture structure "ps" on line **1032**, intra vlc select "ivl" on **1033**, alternate scan select "asc" on line **1034** and quantizer type "qt", on line **1035**. Also included is macroblock header information on bus **1015** which comprises quantizer step size "qs" on line **10 16**, motion type 'mot' on line **1017**, motion vectors "mv" on line **1018**, coding type "cdt" on line **1019** and macroblock type "mbt" on line **1020**. The macroblock type "mbt" on line **1020** is analyzed in **1025** to yield intra/inter decision "in the" on line **1026**, coded block pattern "cbp" on bus **1029**, forward motion "fm" on line **1028** and backward motion "bm" signal on line **1027**. The decoded DCT coefficient data on bus **1036** is input to inverse scan selector **1040** which is controlled by the alternate scan "asc" signal on line **1037** and either performs either an alternative scan or zigzag scan as required. The output of inverse scan **1040** are a normally ordered two-dimensional DCT coefficient blocks and are input to inverse quantizer and inverse DCT **1045** which uses quantization matrices on line **1038**, intra/inter signal in the on bus **1042**, quantizer type signal "qt" on bus **1043** and quantizer step size "qs" on bus **1044** to perform inverse quantization. The output of inverse quantizer and inverse DCT on bus **1051** is blocks of pixels and to that a motion compensated prediction signal on bus **1075** is summed in adder **1052**, the resulting signal on bus **1053** is reconstructed video on bus **1053**, **1054** and **1080**. The reconstructed video on bus **1054** is used to generate motion compensated prediction in motion compensator **1065** which is controlled by signals 'fm' on line **1055**, "bm" on line **1056**, "ps" on line **1057**, "pt" on line **1058**, "mv" on line **1059** and "mot" on line **1060**. A difference with respect to motion estimator and compensator **950** at the encoder is that these signals do not need to be generated but instead are extracted from various header information layers in the bitstream. The output of motion compensator **1056** is available on bus **1066**, and is either applied as is to bus **1075** via intra/inter adapter **1070**, or is zeroed out when "in the" on line **1067** controlling intra/inter adapter is "1" and output on bus **1075**.

I claim:

1. A method of encoding a video signal, comprising the steps of:

generating a set of frequency coefficient signals, the set representing the video signal, and corresponding to an NxM matrix, wherein each of the frequency coefficient signals corresponds to a predetermined horizontal coordinate and a predetermined vertical coordinate in the matrix;

scanning a first subset of the frequency coefficient signals within the set in a predetermined first subset scanning order, as represented by the following list of coordinate pairs, each pair representing a horizontal and vertical coordinate in the matrix, to create an ordered set of frequency coefficient signals:

(0, 0), (0, 1), (0, 2), (0, 3), (1, 0), (1, 1), (2, 0), (2, 1), (1, 2), (1,

3), (0, 4), (0, 5), (0, 6), (0, 7) (1, 7);

and

generating an encoded video signal, the encoded video signal including the ordered set of frequency coefficient signals.

2. The method of claim 1 wherein the scanning step is performed in response to a frame format associated with the video signal.

3. The method of claim 2 in which the frame format is an interlaced frame format.

4. The method of claim 1 further including a step of scanning a second subset of the frequency coefficient signals within the set in a predetermined second subset scanning order such that the ordered set includes frequency coefficient signals in the second subset, wherein the second subset scanning order is represented by the following list of coordinate pairs:

(1, 6), (1, 5), (1, 4), (2, 3), (2, 2), (3, 0), (3, 1), (4, 0), (4, 1), (3, 2), (3, 3), (2, 4), (2, 5), (2, 6), (2, 7).

5. The method of claim 4 further including a step of scanning a third subset of the frequency coefficient signals within the set in a predetermined third subset scanning order such that the ordered set includes the scanned frequency coefficient signals in the third subset, wherein the third subset scanning order is represented by the following list of coordinate pairs:

(3, 4), (3, 5), (3, 6), (3, 7), (4, 2), (4, 3), (5, 0), (5, 1), (6, 0), (6, 1), (5, 2), (5, 3), (4, 4), (4, 5), (4, 6) (4, 7).

6. The method of claim 5 further including a step of scanning a fourth subset of the frequency coefficient signals within the set in a predetermined fourth subset scanning order such that the ordered set includes the scanned frequency coefficient signals in the fourth subset, wherein the fourth subset scanning order is represented by the following list of coordinate pairs:

(5, 4), (5, 5), (5, 6), (5, 7), (6, 2), (6, 3), (7, 0), (7, 1), (7, 2), (7, 3), (6, 4), (6, 5), (6, 6), (6, 7), (7, 4), (7, 5), (7, 6), (7, 7).

7. A method for encoding a video signal, comprising the steps of:

generating a set of frequency coefficient signals, the set representing the video signal, wherein the set corresponds to an N×M matrix and each of the frequency coefficient signals corresponds to a predetermined horizontal coordinate and a predetermined vertical coordinate in the matrix;

alternatively selecting between a first scanning order and a second scanning order in response to a frame format associated with the video signal;

scanning the set of frequency coefficient signals according to the selected scanning order to cream an ordered set of frequency coefficient signals; and

generating an encoded video signal, the encoded video signal including the ordered set of frequency coefficient signals.

8. The method of claim 7 in which the first scanning order comprises a zigzag scanning order.

9. The method of claim 7 in which the second scanning order includes a first subset scanning order represented by the following list of coordinate pairs, each pair representing a horizontal and vertical coordinate in the matrix:

(0, 0), (0, 1), (0, 2), (0, 3), (1, 0), (1, 1), (2, 0), (2, 1), (1, 2), (1, 3), (0, 4), (0, 5), (0, 6), (0, 7) (1, 7).

10. The method of claim 9 in which the second scanning order further includes a second subset scanning order matrix such that the ordered set includes frequency coefficient signals in a second subset, the second subset scanning order represented by the following list of coordinate pairs, each pair representing a horizontal and vertical coordinate in the matrix:

(1, 6), (1, 5), (1, 4), (2, 3), (2, 2), (3, 0), (3, 1), (4, 0), (4, 1), (3, 2), (3, 3), (2, 4), (2, 5), (2, 6), (2, 7).

11. The method of claim 10 in which the second scanning order further includes a third subset scanning order matrix such that the ordered set includes frequency coefficient signals in a third subset, the third subset scanning order represented by the following list of coordinate pairs, each pair representing a horizontal and vertical coordinate in the matrix:

(3, 4), (3, 5), (3, 6), (3, 7), (4, 2), (4, 3), (5, 0), (5, 1), (6, 0), (6, 1), (5, 2), (5, 3), (4, 4), (4, 5), (4, 6) (4, 7).

12. The method of claim 11 in which the second scanning order further includes a fourth subset scanning order matrix such that the ordered set includes frequency coefficient signals in the fourth subset, the fourth subset scanning order represented by the following list of coordinate pairs, each pair representing a horizontal and vertical coordinate in the matrix:

(5, 4), (5, 5), (5, 6), (5, 7), (6, 2), (6, 3), (7, 0), (7, 1), (7, 2), (7, 3), (6, 4), (6, 5), (6, 6), (6, 7), (7, 4), (7, 5), (7, 6), (7, 7).

13. An apparatus for encoding a video signal, comprising:

a discrete cosine transform coefficient generator for generating a set of frequency coefficient signals, the set representing the video signal and corresponding to an N×M matrix, wherein each of the frequency coefficient signals corresponds to a predetermined horizontal coordinate and a predetermined vertical coordinate in the matrix;

a scanner for scanning a first subset of the frequency coefficient signals within the set in a predetermined first subset scanning order, as represented by the following list of coordinate pairs, each pair representing a horizontal and vertical coordinate in the matrix, to create an ordered set of frequency coefficient signals:

(0, 0), (0, 1), (0, 2), (0, 3), (1, 0), (1, 1), (2, 0), (2, 1), (1, 2), (1, 3), (0, 4), (0, 5), (0, 6), (0, 7) (1, 7);

and

a means for generating an encoded video signal, the encoded video signal including the ordered set of frequency coefficient signals.

14. The apparatus of claim 13 in which the scanner performs the scanning in response to a frame format associated with the video signal.

15. The apparatus of claim 14 in which the frame format is an interlaced frame format.

16. The apparatus of claim 13 in which the scanner further including a means for scanning a second subset of the frequency coefficient signals within the set in a predetermined second subset scanning order such that the ordered set includes frequency coefficient signals in the second subset, wherein the second subset scanning order is represented by the following list of coordinate pairs:

(1, 6), (1, 5), (1, 4), (2, 3), (2, 2), (3, 0), (3, 1), (4, 0), (4, 1), (3,

2), (3, 3), (2, 4), (2, 5), (2, 6), (2, 7).

17. The apparatus of claim **16** in which the scanner further includes a means for scanning a third subset of the frequency coefficient signals within the set in a predetermined third subset scanning order such that the ordered set includes the scanned frequency coefficient signals in the third subset, wherein the third subset scanning order is represented by the following list of coordinate pairs:

(3, 4), (3, 5), (3, 6), (3, 7), (4, 2), (4, 3), (5, 0), (5, 1), (6, 0), (6, 1), (5, 2), (5, 3), (4, 4), (4, 5), (4, 6) (4, 7).

18. The apparatus of claim **17** in which the scanner further includes a means for scanning a fourth subset of the frequency coefficient signals within the set in a predetermined fourth subset scanning order such that the ordered set includes the scanned frequency coefficient signals in the fourth subset, wherein the fourth subset scanning order is represented by the following list of coordinate pairs:

(5, 4), (5, 5), (5, 6), (5, 7), (6, 2), (6, 3), (7, 0), (7, 1), (7, 2), (7, 3), (6, 4), (6, 5), (6, 6), (6, 7), (7, 4), (7, 5), (7, 6), (7, 7).

19. An apparatus for encoding a video signal, comprising:

a discrete cosine transform generator for generating a set of frequency coefficient signals, the set representing the video signal and corresponding to an N×M matrix, wherein each of the frequency coefficient signals corresponds to a predetermined horizontal coordinate and a predetermined vertical coordinate in the matrix;

a scan selector for alternatively selecting between a first scanning order and a second scanning order in response to a frame format associated with the video signal;

a scanner for scanning the set of frequency coefficient signals according to the selected scanning order to create an ordered set of frequency coefficient signals; and

a means for generating an encoded video signal, the encoded video signal including the ordered set of frequency coefficient signals.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 5,500,678
APPLICATION NO. : 08/215532
DATED : March 19, 1996
INVENTOR(S) : Atul Puri

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 9, line 55, "cream" should read --create--.

Signed and Sealed this

Twenty-ninth Day of May, 2007

JON W. DUDAS
Director of the United States Patent and Trademark Office

US005136377A

United States Patent [19]

Johnston et al.

[11] Patent Number: 5,136,377

[45] Date of Patent: Aug. 4, 1992

[54] ADAPTIVE NON-LINEAR QUANTIZER

[75] Inventors: James D. Johnston, Warren; Scott C. Knauer, Mountainside; Kim N. Matthews, Watchung; Arun N. Netravali, Westfield; Eric D. Petajan, Watchung; Robert J. Safranek, New Providence; Peter H. Westerink, Newark, all of N.J.

[73] Assignee: AT&T Bell Laboratories, Murray Hill, N.J.

[21] Appl. No.: 626,279

[22] Filed: Dec. 11, 1990

[51] Int. Cl.[5] H04N 7/12; H04N 7/01; H04N 7/04

[52] U.S. Cl. 358/136; 358/140; 358/141

[58] Field of Search 358/135, 136, 140, 141, 358/12

[56] References Cited

U.S. PATENT DOCUMENTS

4,051,530	9/1977	Kuroda et al.	358/136
4,202,011	5/1980	Koga	358/136
4,460,923	7/1984	Hirano et al.	358/136
4,562,468	12/1985	Koga	358/136
4,672,441	6/1987	Hoelzlwimmer et al.	358/136
4,689,672	8/1987	Furukawa et al.	358/136

OTHER PUBLICATIONS

Patent application Ser. No. 07/495525 filed Mar. 19, 1990 entitled "A High Definition Television Arrangement Employing Motion Compensated Prediction Error Signals", by S. C. Knauer et al.

Patent application Ser. No. 07/495523 filed Mar. 19, 1990 entitled "A High Definition Television Arrangement with Signal Selections Adapted to the Available Transmission Capacity", by S. C. Knauer et al.

Patent application Ser. No. 07/495507 filed Mar. 19, 1990 entitled "A High Definition Television Arrangement Including Noise Immunity Means", by S. C. Knauer et al.

Patent application Ser. No. 07/495559 filed Mar. 19, 1990 entitled "A Television Signal Arrangement Where Selected Signals are Encoded Digitally", by S. C. Knauer et al.

"Digital Pictures" by A. N. Netravali and B. G. Haskell, Plenum Press, 1988, p. 334 et seq.

"A Visual Model Weighted Cosine Transform for Image Compression and Quality Assessment", by Nill in IEEE Transactions on Communications, vol. COM-33, No. 6, Jun., 1985, pp. 551-557.

"Image quality measure based on a human visual system model", by Saghri et al., Optical Engineering, vol. 28, No. 7, Jul. 1989, pp. 813-818.

"Adaptive Quantization of Picture Signals using Spatial Masking", by Netravali et al., Proceedings of IEEE, vol. 65, Apr. 1977, pp. 536-548.

"Design of Statistically Based Buffer Control Policies for Compressed Digital Video", Zdepski et al., 1989 IEEE Conference, pp. 1343-1349.

Primary Examiner—John K. Peng

Attorney, Agent, or Firm—Henry T. Brendzel

[57] ABSTRACT

A quantizer, with quantization control that is sensitive to input signal characteristics and to output buffer fullness responds to an input signal that is divided into blocks and DCT transformed. The transformed signal is analyzed to develop a brightness correction and to evaluate the texture of the image and the change in texture in the image. Based on these, and in concert with the human visual perception model, perception threshold signals are created for each subband of the transformed signal. Concurrently, scale factors for each subband of the transformed signal are computed, and a measure of variability in the transformed input signal is calculated. A measure of the fullness of the buffer to which the quantizer sends its encoded results is obtained, and that measure is combined with the calculated signal variability to develop a correction signal. The correction signal modifies the perception threshold signals to develop threshold control signals that are applied to the quantizer. The scale factors are also applied to the quantizer, as well as a global target distortion measure.

31 Claims, 11 Drawing Sheets

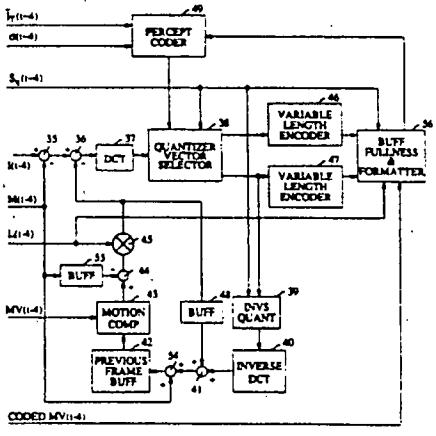


FIG. 1

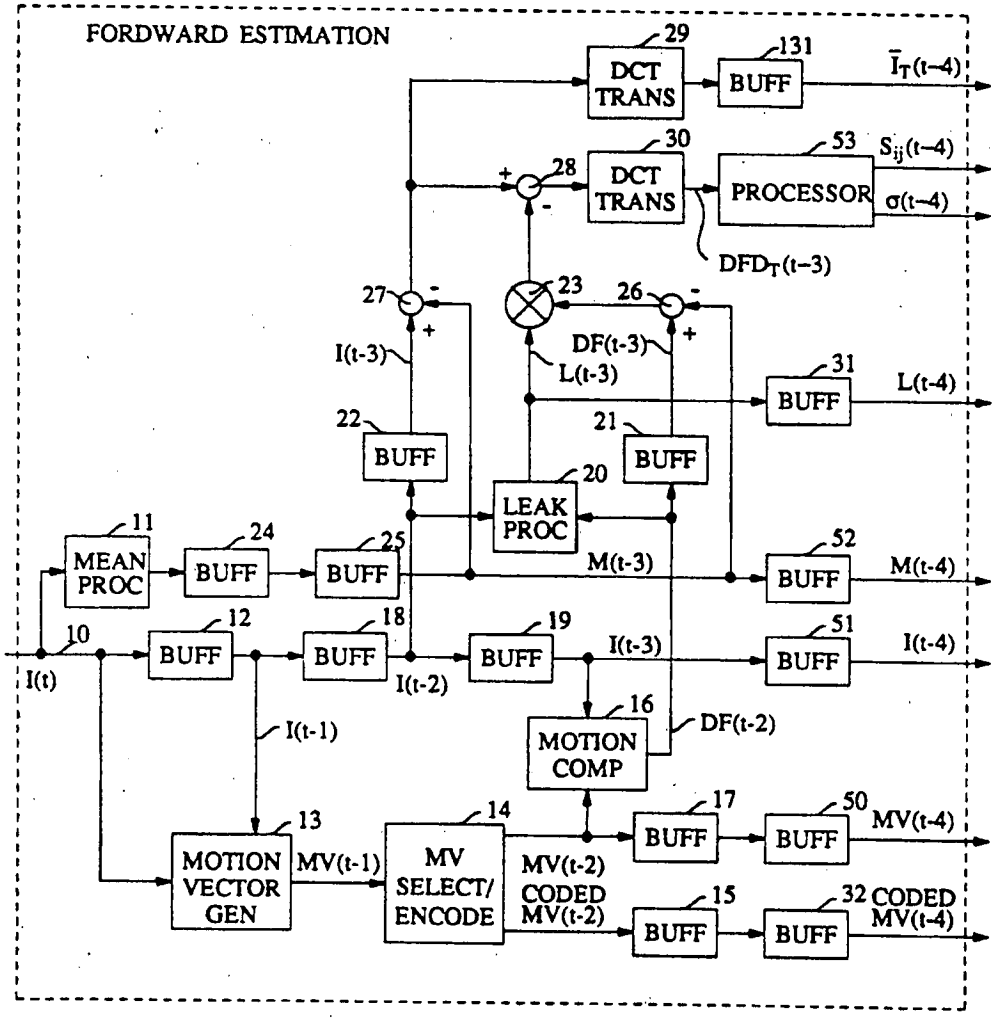

FORDWARD ESTIMATION
DCT TRANS
29
BUFF
131
$\bar{I}_T(t-4)$
DCT TRANS
30
28
PROCESSOR
53
$S_{ij}(t-4)$
$\sigma(t-4)$
$DFD_T(t-3)$
27
23
26
I(t-3)
DF(t-3)
L(t-3)
BUFF
31
L(t-4)
22
BUFF
21
BUFF
LEAK PROC
20
MEAN PROC
11
BUFF
24
BUFF
25
M(t-3)
BUFF
52
M(t-4)
10
I(t)
BUFF
12
BUFF
18
BUFF
19
I(t-3)
BUFF
51
I(t-4)
I(t-2)
I(t-1)
MOTION COMP
16
DF(t-2)
MOTION VECTOR GEN
13
MV(t-1)
MV SELECT/ ENCODE
14
MV(t-2)
CODED MV(t-2)
BUFF
17
BUFF
50
MV(t-4)
BUFF
15
BUFF
32
CODED MV(t-4)

FIG. 2

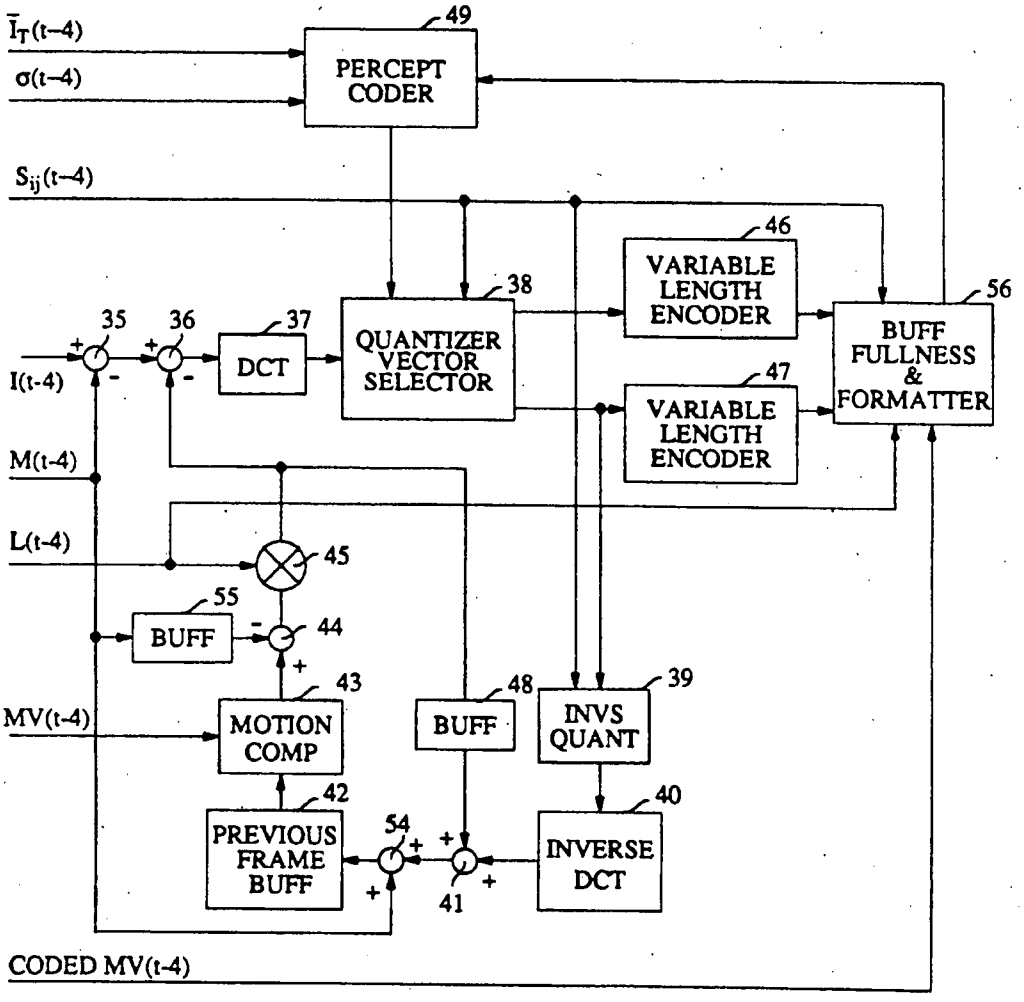

$\bar{I}_T(t-4)$
σ(t-4)
$S_{ij}(t-4)$
I(t-4)
M(t-4)
L(t-4)
MV(t-4)
CODED MV(t-4)
PERCEPT CODER
49
DCT
37
QUANTIZER VECTOR SELECTOR
38
VARIABLE LENGTH ENCODER
46
VARIABLE LENGTH ENCODER
47
BUFF FULLNESS & FORMATTER
56
35
36
45
BUFF
55
44
43
MOTION COMP
BUFF
48
INVS QUANT
39
INVERSE DCT
40
PREVIOUS FRAME BUFF
42
54
41

FIG. 3

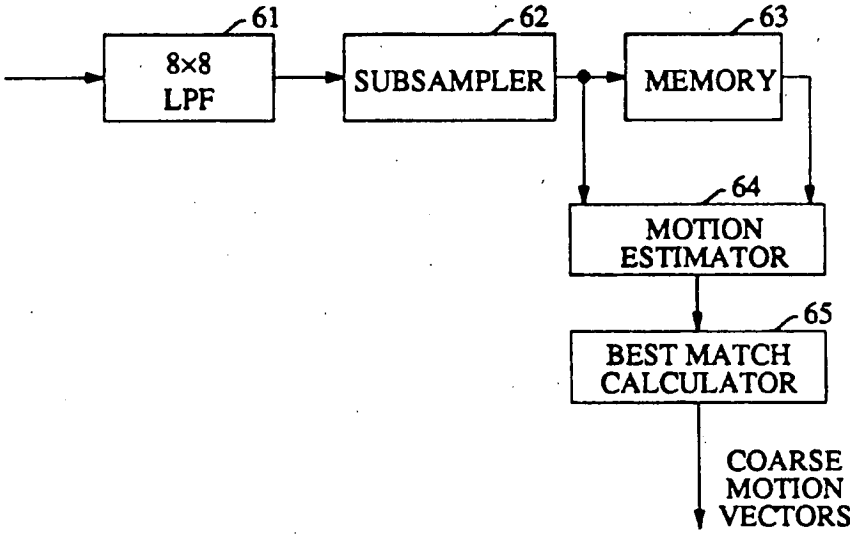

61
8×8
LPF
62
SUBSAMPLER
63
MEMORY
64
MOTION
ESTIMATOR
65
BEST MATCH
CALCULATOR
COARSE
MOTION
VECTORS

FIG. 4

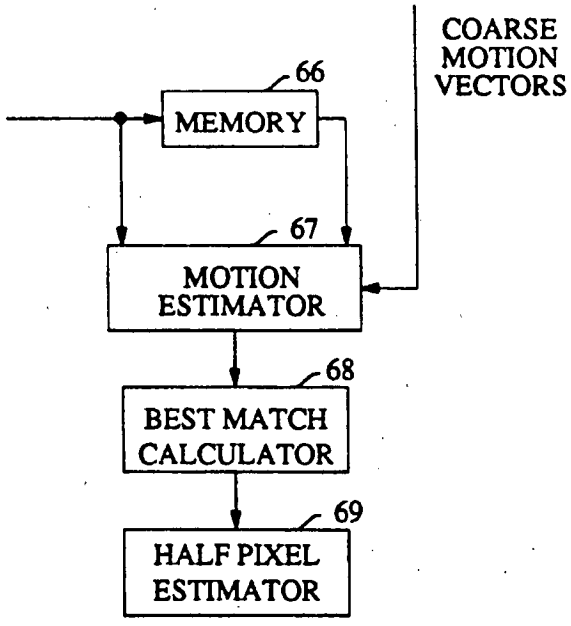

COARSE
MOTION
VECTORS
66
MEMORY
67
MOTION
ESTIMATOR
68
BEST MATCH
CALCULATOR
69
HALF PIXEL
ESTIMATOR

FIG. 5

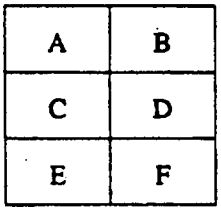


FIG. 6

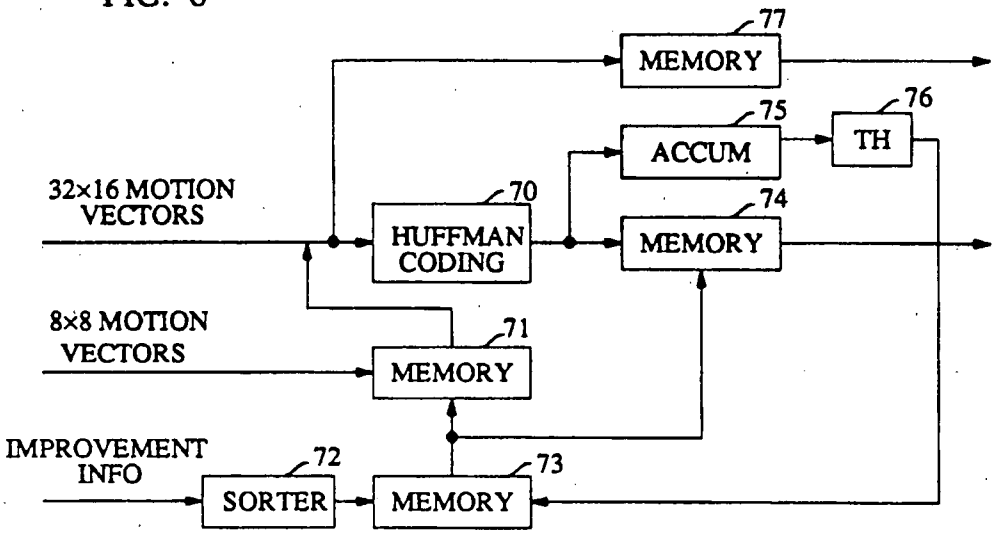


FIG. 7

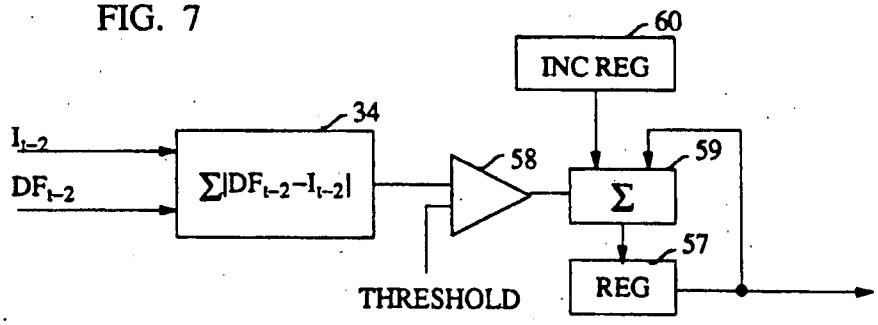


FIG. 8

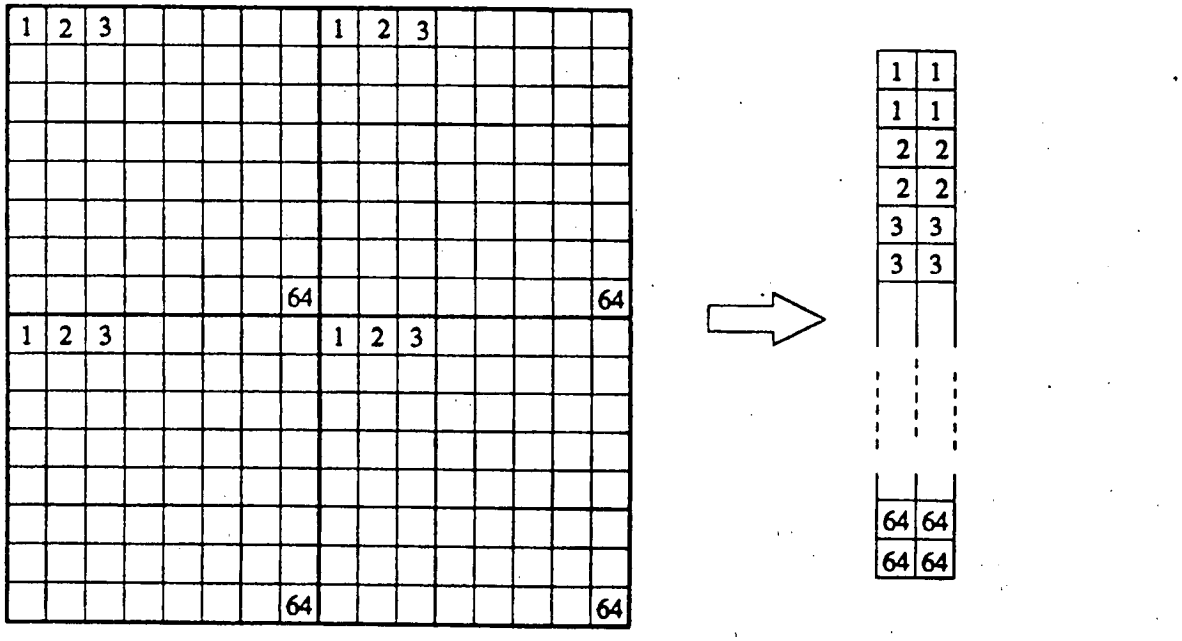


FIG. 9

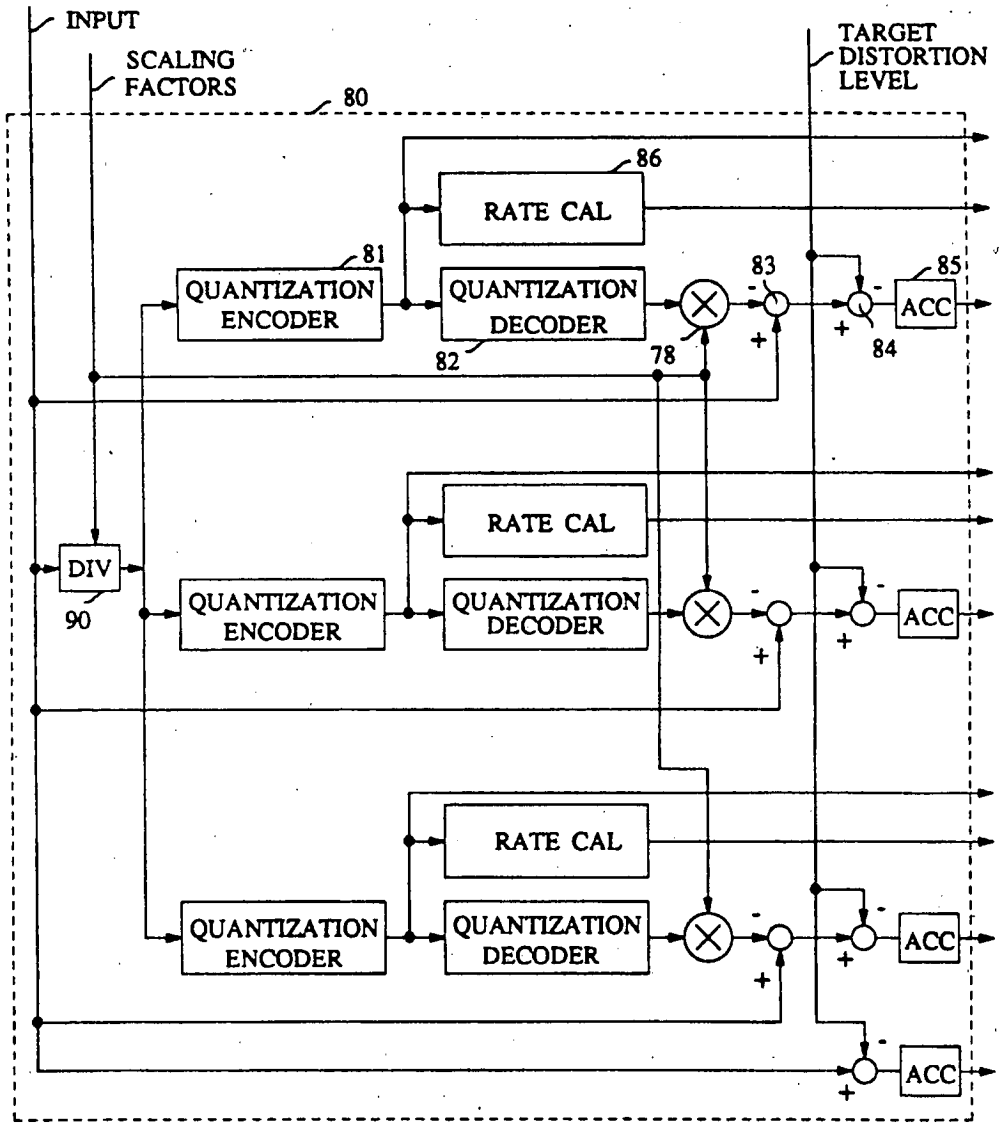

INPUT
SCALING FACTORS
80
TARGET DISTORTION LEVEL
86
RATE CAL
81
QUANTIZATION ENCODER
QUANTIZATION DECODER
82
83
85
ACC
84
78
DIV
90
RATE CAL
QUANTIZATION ENCODER
QUANTIZATION DECODER
ACC
RATE CAL
QUANTIZATION ENCODER
QUANTIZATION DECODER
ACC
ACC

FIG. 10

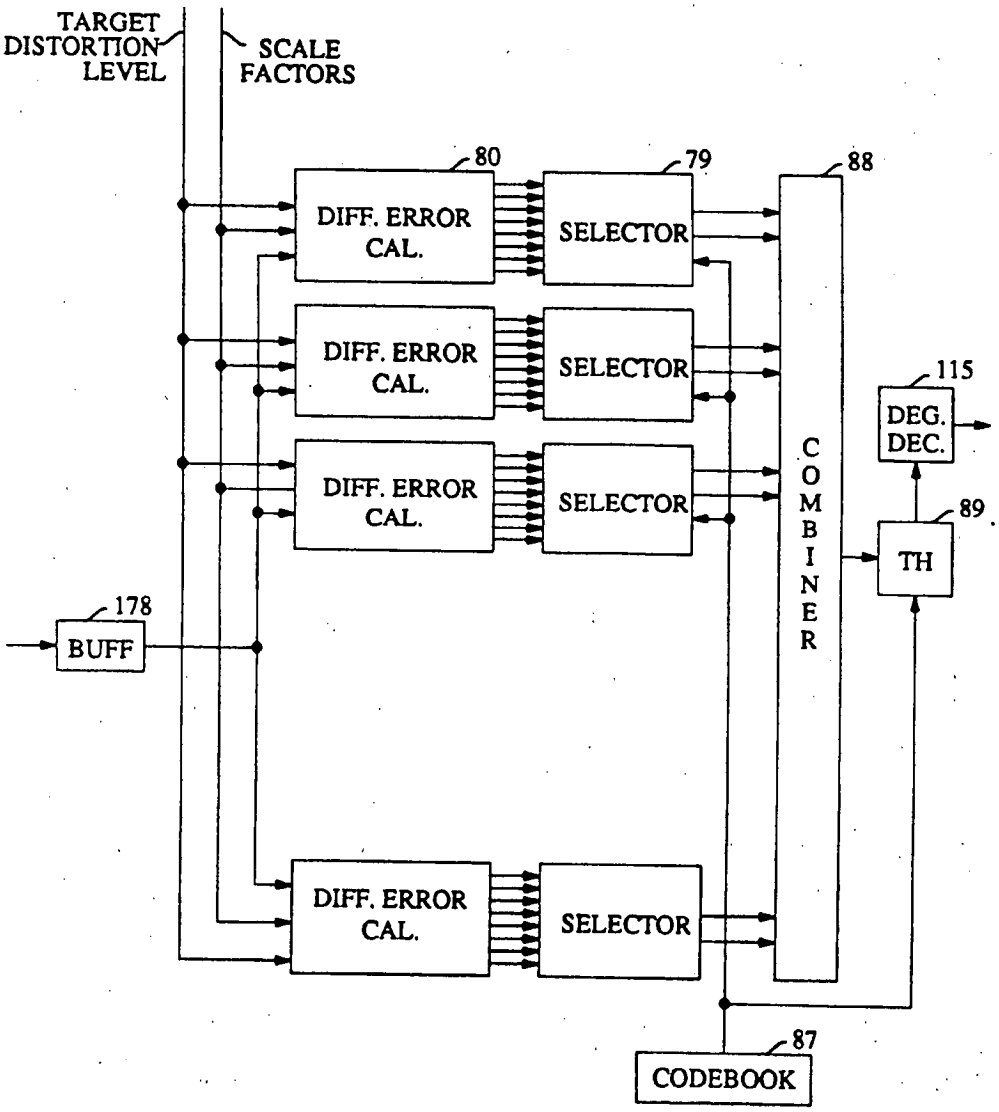

TARGET DISTORTION LEVEL
SCALE FACTORS
80
79
88
DIFF. ERROR CAL.
SELECTOR
DIFF. ERROR CAL.
SELECTOR
DIFF. ERROR CAL.
SELECTOR
115
DEG. DEC.
89
TH
COMBINER
178
BUFF
DIFF. ERROR CAL.
SELECTOR
87
CODEBOOK

FIG. 11

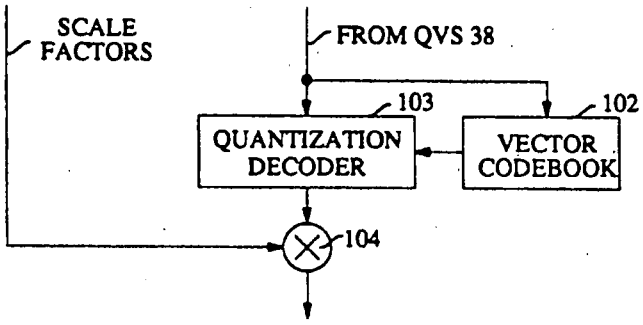


FIG. 12

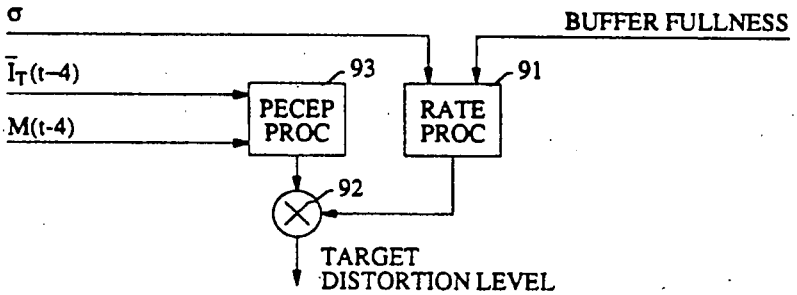


FIG. 13

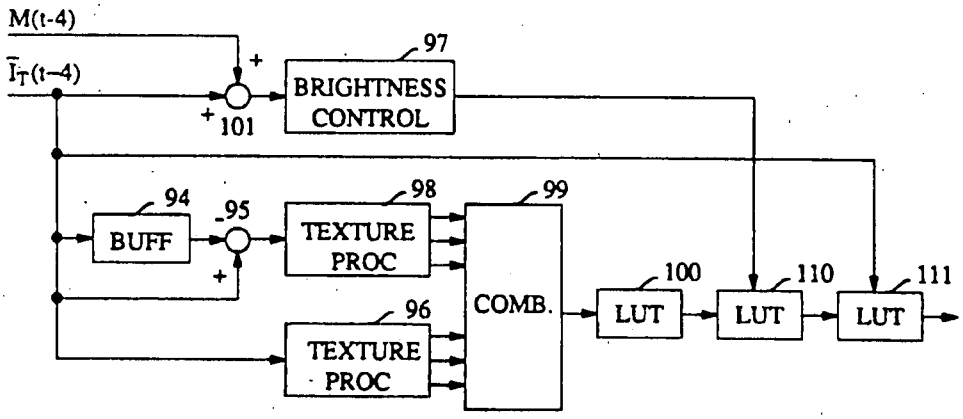


FIG. 14

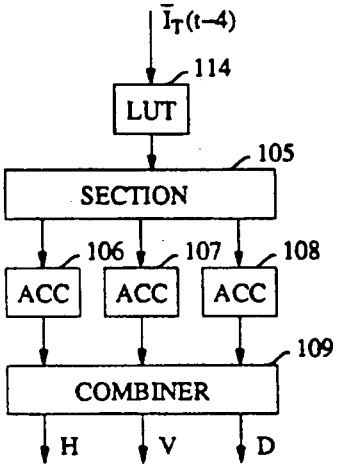

$\bar{I}_T(t-4)$
114
LUT
105
SECTION
106
107
108
ACC
ACC
ACC
109
COMBINER
H
V
D

FIG. 15

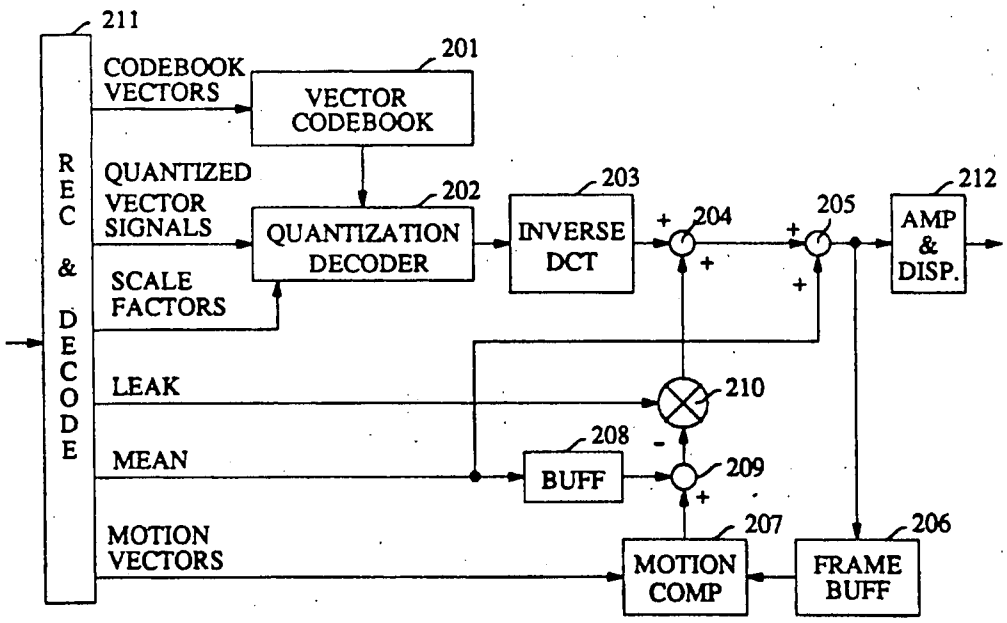

211
REC & DECODE
CODEBOOK VECTORS
201
VECTOR CODEBOOK
QUANTIZED VECTOR SIGNALS
SCALE FACTORS
202
QUANTIZATION DECODER
203
INVERSE DCT
204
205
212
AMP & DISP.
LEAK
210
MEAN
208
BUFF
209
MOTION VECTORS
207
MOTION COMP
206
FRAME BUFF

FIG. 16

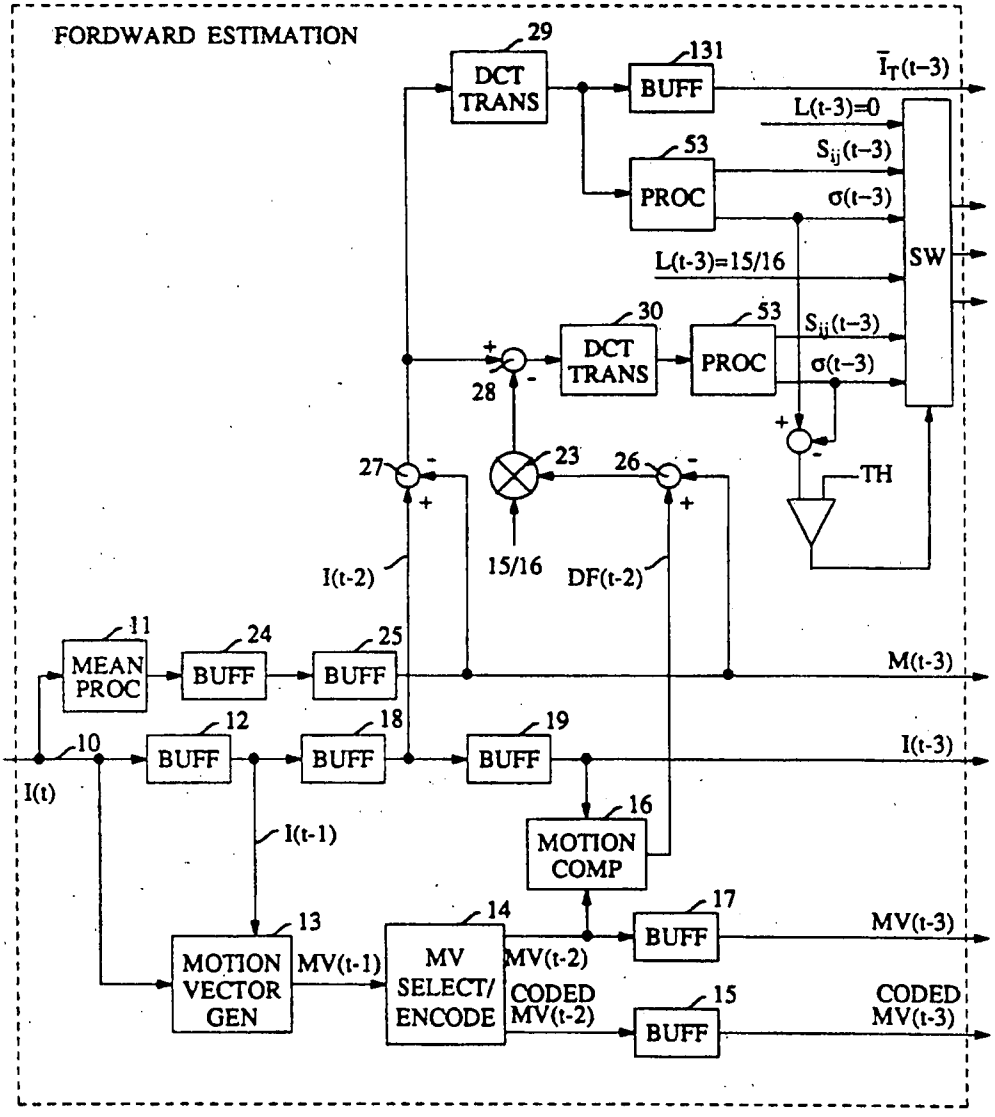
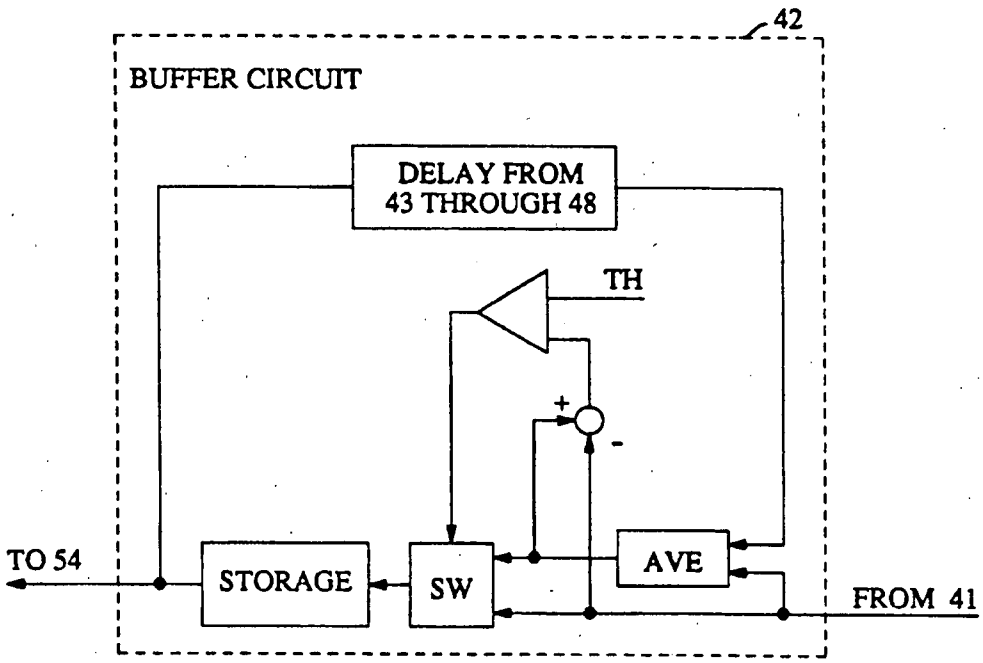

FORDWARD ESTIMATION
DCT TRANS
29
BUFF
131
$\bar{I}_T(t-3)$
L(t-3)=0
PROC
53
$S_{ij}(t-3)$
σ(t-3)
SW
L(t-3)=15/16
DCT TRANS
30
PROC
53
$S_{ij}(t-3)$
σ(t-3)
28
TH
27
23
26
15/16
I(t-2)
DF(t-2)
MEAN PROC
11
BUFF
24
BUFF
25
M(t-3)
I(t)
10
BUFF
12
BUFF
18
BUFF
19
I(t-3)
I(t-1)
MOTION COMP
16
MOTION VECTOR GEN
13
MV(t-1)
MV SELECT/ ENCODE
14
MV(t-2)
BUFF
17
MV(t-3)
CODED MV(t-2)
BUFF
15
CODED MV(t-3)

FIG. 17


42
BUFFER CIRCUIT
DELAY FROM
43 THROUGH 48
TH
+
-
TO 54
STORAGE
SW
AVE
FROM 41

ADAPTIVE NON-LINEAR QUANTIZER

CROSS-REFERENCE TO RELATED APPLICATION

This application is related to four other applications filed on this date. Their titles are:

1. A High Definition Television Coding Arrangement With Graceful Degradation
2. An Adaptive Leak HDTV Encoder
3. HDTV Encoder with Forward Estimation and Constant Rate Motion Vectors
4. HDTV Receiver

BACKGROUND OF THE INVENTION

This invention relates to encoding of signals, and more specifically, to encoding of signals with a controlled quantizer.

When image signals are digitized and linearly quantized, a transmission rate of about 100 Mbits per second is necessary for images derived from standard TV signals. For HDTV, with its greater image size and greater resolution, a much higher transmission rate would be is required. When terrestrial transmission is desired, and when the option of allocating greater bandwidth per channel is unavailable, it is necessary to compress the HDTV signal within the allocated bandwidth. Having an allocated bandwidth is tantamount to having a certain number of information bits that can be transmitted in each selected time interval, such as an image frame. All of the compression techniques of HDTV signals must of necessity consider this bit budget.

In order to reduce the bit rate, or to fit the encoded signal of an image frame within the allocated bit budget, various coding schemes have been studied. One is the so-called "differential pulse code modulation" (DPCM) coding approach. By this method, the value of a particular pixel at any moment is predicted on the basis of values of pixels which have been already coded. The necessary number of bits is reduced by coding the difference (error) between the predicted value and the value of the particular pixel at that moment. According to this differential pulse code modulation, it is possible to reduce the number of bits per pixel by approximately half.

Still, that number is much larger than can be accepted for terrestrial transmission, so a second look is typically taken at the quantizer itself. Clearly, if the step size of the quantizer is made large enough, the number of bits generated can be reduced to an acceptable level. Alas, the quantization error resulting therefrom would yield a picture that is far from acceptable.

Experiments show, and it is quite logical, that the effect of quantization error is different for different types of picture and, expectedly, artisans have tried to tailor quantizers to the pictures being encoded.

U.S. Pat. No. 4,802,232 issued Jan. 31, 1989, for example, describes one such attempt. In accordance with the described approach the picture to be encoded is divided into regions and each region of the input image is classified into one of a preset number of classes. The signal of each class is quantized in a specified manner, and the manner of quantization of the different classes, of course, differ. The limitation of this approach is the hard decisions boundaries between the classes.

Nill in "A Visual Model Weighted Cosine Transform for Image Compression and Quality Assessment", IEEE Transactions on Communications, Vol. COM-33, No. 6, June, 1985, pg. 551–557 and Saghri et al. "Image quality measure based on a human visual system model", Optical Engineering, Vol. 28, No. 7, July 1989, pg. 813–818, describe very similar approaches. These approaches utilize fixed global thresholds for each frequency band, where the thresholds are set based on a model of the human visual systems (HVS) modulation transfer function (MTF), and the transform utilized. The limitation of this approach is that the quantization cannot adapt to local characteristics of the image.

In "Adaptive Quantization of Picture Signals using Spatial Masking", Proceedings of IEEE, Vol. 65, April 1977, pg. 536–548, Netravali et al. describe a means of designing non-uniform quantizers to incorporate spatial masking and brightness correction for predictive coding systems. The limitation of their approach, again, is that the quantization cannot adapt to local characteristics of the image.

In "Design of Statistically Based Buffer Control Policies for Compressed Digital Video", Zdepski et al., an IEEE conference, 1989, pg. 1343–1349, describe an approach where the quantizer in the DPCM loop interacts with an adaptive mode control circuit. The circuit measures the number of bits generated by the quantizer and, based on preselected thresholds, decides for the next frame on one of eight possible quantizer step sizes. The selected step size is employed for the next frame. A similar approach is described in "Digital Pictures" by A. N. Netravali and B. G. Haskell, Plenum Press, 1988, pg. 537 et seq.

The deficiency of these methods is that the quantization steps are altered in discrete jumps and in that no accounting is made of a global distortion target.

It is an object of this invention, therefore, to develop means for controlling the quantizer so that the number of bits generated per frame is, on the average, within the allocated bit budget.

It is another object of this invention to control the quantizer in a manner that is sensitive to the characteristics of the input signal to which human viewers are sensitive.

It is still another object of this invention to control the quantizer in a manner that spreads as evenly as possible the unavoidable quantization noise.

SUMMARY OF THE INVENTION

These and other objects are achieved with a quantizer control mechanism that is responsive to both the input signal and the fullness of the output buffer. More specifically, the input image is divided into blocks and the signal of each block is DCT transformed. The transformed signal is analyzed to develop a brightness correction and to evaluate the texture of the image and the change in texture in the image. Based on these, and in concert with the human visual perception model, perception threshold signals are created for each subband of the transformed signal.

Concurrently, scale factors for each subband of the transformed signal are computed, and a measure of variability in the transformed input signal is calculated. A measure of the fullness of the buffer to which the quantizer sends its encoded results is obtained, and that measure is combined with the calculated signal variability to develop a correction signal. The correction signal modifies the perception threshold signals to develop threshold control signals that are applied to the quan-

tizer. The scale factors are also applied to the quantizer, as well as a global target distortion measure.

The quantizer pre-multiplies the input signal by the scale factors and quantizes the signal in a manner that is sensitive to the threshold control signals. The quantization itself is performed a number of times by scanning a codebook that specifies the exact manner of quantizing of each of the subbands of the transformed signal. Accounting for the global target distortion, the best quantization result is selected and sent to the aforementioned buffer.

BRIEF DESCRIPTION OF THE DRAWING

FIG. 1 presents a block diagram of a forward estimation section of an HDTV digital encoder;

FIG. 2 presents a block diagram of an encoding loop section of an HDTV encoder that interacts with the forward estimation section of FIG. 1;

FIG. 3 depicts a hardware organization for a coarse motion vector detector;

FIG. 4 depicts a hardware organization for a fine motion vector detector that takes into account the output of the coarse motion vector detector;

FIG. 5 illustrates the spatial relationship of what is considered a "slice" of image data;

FIG. 6 shows one way for selecting a mix of motion vectors to fit within a given bit budget;

FIG. 7 presents one embodiment for evaluating a leak factor, α;

FIG. 8 illustrates the arrangement of a superblock that is quantized in QVS 38 block of FIG. 2;

FIG. 9 depicts how a set of selection error signals is calculated in preparation for codebook vector selection;

FIG. 10 presents a block diagram of QVS block 38;

FIG. 11 is a block diagram of inverse quantization block 39;

FIG. 12 presents the structure of perceptual coder 49;

FIG. 13 illustrates the structure of perceptual processor 93;

FIG. 14 is a block diagram of texture processors 96 and 98;

FIG. 15 presents a block diagram of a digital HDTV receiver;

FIG. 16 presents a modified forward estimation block that choses a leak factor from a set of two fixed leak factors; and

FIG. 17 a frame buffere circuit that includes a measure of temporal filtering.

DETAILED DESCRIPTION

The motion estimation principles of this invention start with the proposition that for a given bit budget, a much better overall prediction error level can be attained by employing blocks of variable sizes. Starting with large sized blocks that handle large image sections for which a good translation vector can be found, the motion vector selection in accordance with this invention then replaces, or adds, to some of motion vectors for the large blocks with motion vectors for the small blocks.

While these principles of the invention can be put to good use in the prediction of other types of signals, it is clear that it is quite useful in connection with the encoding of HDTV signals. However, in the HDTV application there is simply not enough time within the encoding loop to compute the best motion vectors and to then select the best mix of motion vectors. Accordingly, a forward estimation section is incorporated in the encoder, and that section creates as many of the signals used within the encoder as it is possible. The input image signal is, of course, appropriately delayed so that the need for the signals that are computed within the forward estimation section does not precede their creation.

In order to appreciate the operation of the motion vector selection and encoding processes improvement of this invention serves, the following describes the entire coder section of an HDTV transmitter. The principles of this invention are, of course, primarily described in connection with the motion vector estimation and selection circuits depicted of FIG. 1.

DETAILED DESCRIPTION

Quantization of the signals is one of the most important tasks that the HDTV transmitter must perform. The quantizer, however, is but a part of the entire encoding loop. Moreover, the encoding loop, generally, and the quantizer, in particular, require a number of signals that need a substantial period of time for computation. Fortunately, it is possible to compute these signals in a forward estimation section, which "off line" as far as the encoding loop is concerned.

To better understand the function and workings of the quantizer of this invention, the following describes the entire coder section of an HDTV transmitter, i.e., both the forward estimation section and the encoding loop. The detailed principles of this invention are primarily described in connection with the perceptual encoder circuit, the quantizer circuit, and the associated circuits.

In FIG. 1, the input signal is applied at line 10. It is a digitized video signal that arrives in sequences of image frames. This input signal is applied to frame-mean processor 11, to buffer 12, and to motion vector generator 13. The output of buffer 12 is also applied to motion vector generator block 13. Frame-mean processor 11 develops the mean value of each incoming frame. That value is delayed in buffers 24 and 25, and applied to a number of elements within FIG. 1, as described below. It is also sent to the encoding loop of FIG. 2 through buffer 52. Motion vector generator 13 develops motion vectors which are applied to motion vector selector/encoder 14 and, thereafter, through buffers 15 and 32, wherefrom the encoded motion vectors are sent to the encoding loop of FIG. 2. The unencoded output of motion vector selector/encoder 14 is also applied to motion compensator block 16, and to buffer 17 followed by buffer 50, wherefrom the unencoded motion vectors are sent to the encoding loop of FIG. 2.

The output of buffer 12 is applied to buffer 18 and thereafter to buffers 19 and 51, wherefrom it is sent to the encoding loop of FIG. 2. The output of buffer 18 is applied to buffer 22 and to leak factor processor 20, and the output of buffer 19 is applied to motion compensator circuit 16. The output of motion compensator 16 is applied to buffer 21 and to leak factor processor 20.

The frame-mean signal of buffer 25 is subtracted from the output of buffer 21 in subtracter 26 and from the output of buffer 22 in subtracter 27. The outputs of subtracter 26 and leak processor 20 are applied to multiplier 23, and the output of multiplier 23 is applied to subtracter 28. The output of leak processor 20 is also sent to the encoding loop of FIG. 2 via buffer 31. Element 28 subtracts the output of multiplier 23 from the output of subtracter 27 and applies the result to DCT transform circuit 30. The output of transform circuit 30

is applied to processor 53 which computes scale factors S_{ij} and signal standard deviation σ and sends its results to FIG. 2. The output of subtracter 27 is applied to DCT transform circuit 29, and the output of DCT circuit 29 is sent to the encoding loop of FIG. 2.

To get a sense of the timing relationship between the various elements in FIG. 1, it is useful to set a benchmark, such as by asserting that the input at line 10 corresponds to the image signal of frame t; i.e., that the input signal at line 10 is frame I(t). All of the buffers in FIG. 1 store and delay one frame's worth of data. Hence, the output of buffer 12 is I(t-1), the output of buffer 18 is I(t-2), the output of buffer 19 is I(t-3), and the output of buffer 51 is I(t-4).

Motion vector generator 13 develops motion vectors M(t) that (elsewhere in the encoder circuit and in the decoder circuit) assist in generating an approximation of frame I(t) based on information of frame I(t-1). It takes some time for the motion vectors to be developed (an internal delay is included to make the delay within generator 13 equal to one frame delay). Thus, the output of generator 13 (after processing delay) corresponds to a set of motion vectors MV(t-1). Not all of the motion vectors that are created in motion vector generator 13 are actually used, so the output of generator 13 is applied to motion vector selector/encoder 14 where a selection process takes place. Since the selection process also takes time, the outputs of selector/encoder 14 are MV(t-2) and the CODED MV(t-2) signals, which are the motion vectors, and their coded representations, that assist in generating an approximation of frame I(t-2) based on information of frame I(t-3). Such an I(t-2) signal is indeed generated in motion compensator 16, which takes the I(t-3) signal of buffer 19 and the motion vectors of selector/encoder 14 and develops therefrom a displaced frame signal DF(t-2) that approximates the signal I(t-2). Buffers 17 and 50 develop MV(t-4) signals, while buffers 15 and 32 develop the CODED MV(t-4) signals.

As indicated above, processor 11 develops a frame-mean signal. Since the mean signal cannot be known until the frame terminates, the output of processor 11 relates to frame t-1. Stated differently, the output of processor 11 is M(t-1) and the output of buffer 25 is M(t-3).

Leak factor processor 20 receives signals I(t-2) and DF(t-2). It also takes time to perform its function (and internal delay is included to insure that it has a delay of exactly one frame), hence the output signal of processor 20 corresponds to the leak factor of frame (t-3). The output of processor 20 is, therefore, designated L(t-3). That output is delayed in buffer 31, causing L(t-4) to be sent to the encoding loop.

Lastly, the processes within elements 26–30 are relatively quick, so the transformed image ($\bar{I}_T$) and displaced frame difference (DFD_T) outputs of elements 29 and 30 correspond to frame I_T(t-3) and DFD_T(t-3), respectively, and the output of processor 53 corresponds to S_{ij}(t-4) and σ(t-4).

FIG. 2 contains the encoding loop that utilizes the signals developed in the forward estimation section of FIG. 1. The loop itself comprises elements 36, 37, 38, 39, 40, 41, 54, 42, 43, 44 and 45. The image signal I(t-4) is applied to subtracter 36 after the frame-mean signal M(t-4) is subtracted from it in subtracter 35. The signal developed by subtracter 36 is the difference between the image I(t-4) and the best estimation of image I(t-4) that is obtained from the previous frame's data contained in the encoding loop (with the previous frame's frame-mean excluded via subtracter 44, and with a leak factor that is introduced via multiplier 45). That frame difference is applied to DCT transform circuit 37 which develops 2-dimensional transform domain information about the frame difference signal of subtracter 36. That information is encoded into vectors within quantizer-and-vector-selector (QVS) 38 and forwarded to encoders 46 and 47. The encoding carried out in QVS 38 and applied to encoder 47 is reversed to the extent possible within inverse quantizer 39 and applied to inverse DCT circuit 40.

The output of inverse DCT circuit 40 approximates the output of subtracter 36. However, it does not quite match the signal of subtracter because only a portion of the encoded signal is applied to element 39 and because it is corrupted by the loss of information in the encoding process of element 38. There is also a delay in passing through elements 37, 38, 39, and 40. That delay is matched by the delay provided by buffer 48 before the outputs of buffer 48 and inverse DCT transform circuit 40 are combined in adder 41 and applied to adder 54. Adder 54 adds the frame-mean signal M(t-4) and applies the results to buffer 42. Buffer 42 complements the delay provided by buffer 48 less the delay in elements 43, 44 and 45 (to form a full one frame delay) and delivers it to motion compensator 43.

Motion compensator 43 is responsive to the motion vectors MV(t-4). It produces an estimate of the image signal I(t-4), based on the approximation of I(t-5) offered by buffer 42. As stated before, that approximation is diminished by the frame-mean of the previous frame, M(t-5), through the action of subtracter 44. The previous frame's frame-mean is derived from buffer 55 which is fed by M(t-4). The results of subtracter 44 are applied to multiplier 45 which multiplies the output of subtracter 44 by the leak factor L(t-4). The multiplication results form the signal to the negative input of subtracter 36.

It may be noted in passing that the action of motion compensator 43 is linear. Therefore, when the action of buffer 42 is also linear—which means that it does not truncate its incoming signals—then adder 54 and subtracter 44 (and buffer 55) are completely superfluous. They are used only when buffer 42 truncates its incoming signal to save on the required storage.

In connection with buffer 42, another improvement is possible. When the processing within elements 36, 37, 38, 39, and 40 and the corresponding delay of buffer 48 are less than the vertical frame retrace interval, the output of buffer 42 can be synchronized with its input, in the sense that pixels of a frame exit the buffer at the same time that corresponding pixels of the previous frame exit the buffer. Temporal filtering can then be accomplished at this point by replacing buffer 42 with a buffer circuit 42 as shown in FIG. 17. In buffer circuit 42, the incoming pixel is compared to the outgoing pixel. When their difference is larger than a certain threshold, the storage element within circuit 42 is loaded with the average of the two compared pixels. Otherwise, the storage element within buffer 42 is loaded with the incoming pixel only.

QVS 38 is also responsive to perceptual coder 49 and to S_{ij}(t-4). That coder is responsive to signals $\bar{I}_T$(t-4) and σ(t-4). Signals S_{ij}(t-4) are also sent to inverse quantization circuit 39 and to buffer fullness and formatter (BFF) 56. BFF block 56 also receives information from encoders 46 and 47, the leak signal L(t-4) and the

CODED MV(t-4) information from buffer 32 in FIG. 1. BFF block 56 sends fullness information to perceptual coder 49 and all if its received information to subsequent circuitry, where the signals are amplified, appropriately modulated and, for terrestrial transmission, applied to a transmitting antenna.

BFF block 56 serves two closely related functions. It packs the information developed in the encoders by applying the appropriate error correction codes and arranging the information, and it feeds information to perceptual coder 49, to inform it of the level of output buffer fullness. The latter information is employed in perceptual coder 49 to control QVS 38 and inverse quantizer 39 and, consequently, the bit rate of the next frame.

The general description above provides a fairly detailed exposition of the encoder within the HDTV transmitter. The descriptions below delve in greater detail into each of the various circuits included in FIGS. 1 and 2.

FRAME-MEAN CIRCUIT 11

The mean, or average, signal within a frame is obtainable with a simple accumulator that merely adds the values of all pixels in the frame and divides the sum by a fixed number. Adding a binary number of pixels offers the easiest division implementation, but division by any other number is also possible with some very simple and conventional hardware (e.g., a look-up table). Because of this simplicity, no further description is offered herein of circuit 11.

MOTION VECTOR GENERATOR 13

The motion vector generator compares the two sequential images I(t) and I(t-1), with an eye towards detecting regions, or blocks, in the current image frame, I(t), that closely match regions, or blocks, in the previous image frame, I(t-1). The goal is to generate relative displacement information that permits the creation of an approximation of the current image frame from a combination of the displacement information and the previous image frame.

More specifically, the current frame is divided into $n \times n$ pixel blocks, and a search is conducted for each block in the current frame to find an $n \times n$ block in the previous frame that matches the current frame block as closely as possible.

If one wishes to perform an exhaustive search for the best displacement of an $n \times n$ pixel block in a neighborhood of a $K \times K$ pixel array, one has to test all of the possible displacements, of which there are $(K-n) \times (K-n)$. For each of those displacements one has to determine the magnitude of the difference (e.g., in absolute, RMS, or square sense) between the $n \times n$ pixel array in the current frame and the $n \times n$ portion of the $K \times K$ pixel array in the previous frame that corresponds to the selected displacement. The displacement that corresponds to the smallest difference is the preferred displacement, and that is what we call the motion vector.

One important issue in connection with a hardware embodiment of the above-described search process is the shear volume of calculations that needs to be performed in order to find the absolutely optimum motion vector. For instance, if the image were subdivided into blocks of 8×8 pixels and the image contains 1024×1024 pixels, then the total number of blocks that need to be matched would be 2^{14}. If an exhaustive search over the entire image were to be performed in determining the best match, then the number of searches for each block would be approximately 2^{20}. The total count (for all the blocks) would then be approximately 2^{34} searches. This "astronomical" number is just too many searches!

One approach for limiting the required number of searches is to limit the neighborhood of the block whose motion vector is sought. In addition to the direct reduction in the number of searches that must be undertaken, this approach has the additional benefit that a more restricted neighborhood limits the number of bits that are required to describe the motion vectors (smaller range), and that reduces the transmission burden. With those reasons in mind, we limit the search neighborhood in both the horizontal and vertical directions to ± 32 positions. That means, for example, that when a 32×16 pixel block is considered, then the neighborhood of search is 80×80 pixels, and the number of searches for each block is 2^{12} (compared to 2^{20}).

As indicated above, the prediction error can be based on a sum of squares of differences, but it is substantially simpler to deal with absolute values of differences. Accordingly, the motion vector generator herein compares blocks of pixels in the current frame with those in the previous frame by forming prediction error signals that correspond to the sum over the block of the absolute differences between the pixels.

To further reduce the complexity and size of the search, a two-stage hierarchical motion estimation approach is used. In the first stage, the motion is estimated coarsely, and in the second stage the coarse estimation is refined. Matching in a coarse manner is achieved in the first stage by reducing the resolution of the image by a factor of 2 in both the horizontal and the vertical directions. This reduces the search area by a factor of 4, yielding only 2^{12} blocks in a 1024×1024 image array. The motion vectors generated in the first stage are then passed to the second stage, where a search is performed in the neighborhood of the coarse displacement found in the first stage.

FIG. 3 depicts the structure of the first (coarse) stage in the motion vector generator. In FIG. 3 the input signal is applied to a two-dimensional, 8 pixel by 8 pixel low-pass filter 61. Filter 61 eliminates frequencies higher than half the sampling rate of the incoming data. Subsampler 62 follows filter 61. It subsamples its input signal by a 2:1 factor. The action of filter 61 insures that no aliasing results from the subsampling action of element 62 since it eliminates signals above the Nyquist rate for the subsampler. The output of subsampler 62 is an image signal with half as many pixels in each line of the image, and half as many lines in the image. This corresponds to a four-fold reduction in resolution, as discussed above.

In FIG. 1, motion vector generator 13 is shown to be responsive to the I(t) signal at input line 10 and to the I(t-1) signal at the output of buffer 12. This was done for expository purposes only, to make the operation of motion vector 13 clearly understandable in the context of the FIG. 1 description. Actually, it is advantageous to have motion vector generator 13 be responsive solely to I(t), as far as the connectivity of FIG. 1 is concerned, and have the delay of buffer 12 be integrated within the circuitry of motion vector generator 13.

Consonant with this idea, FIG. 3 includes a frame memory 63 which is responsive to the output of subsampler 62. The subsampled I(t) signal at the input of frame memory 63 and the subsampled I(t-1) signal at the out-

put of frame memory **63** are applied to motion estimator **64**.

The control of memory **63** is fairly simple. Data enters motion estimator block **64** is sequence, one line at a time. With every sixteen lines of the subsampled I(t). memory **64** must supply to motion estimator block **64** sixteen lines of the subsampled I(t-1); except offset forward by sixteen lines. The 32 other (previous) lines of the subsampled I(t-1) that are needed by block **64** are already in block **64** from the previous two sets of sixteen lines of the subsampled I(t) signal that were applied to motion estimator block **64**.

Motion estimator **64** develops a plurality of prediction error signals for each block in the image. The plurality of prediction error signals is applied to best-match calculator **65** which identifies the smallest prediction error signal. The displacement corresponding to that prediction error signal is selected as the motion vector of the block.

Expressed in more mathematical terms, if a block of width w and height h in the current frame block is denoted by b(x,y,t), where t is the current frame and x and y are the north-west corner coordinates of the block. then the prediction error may be defined as the sum of absolute differences of the pixel values:

$$PE(x,y,w,h,r,s) = \sum_{i=x}^{x+w} \left[\sum_{j=y}^{y+h} |b(i-r,j-s,t-1) - b(i,j,t)| \right] \quad (1)$$

where r and s are the displacements in the x and y directions, respectively.

The motion vector that gives the best match is the displacement (r,s) that gives the minimum prediction error.

The selection of the motion vector is performed in calculator **65**. In cases where there are a number of vectors that have the same minimum error, calculator **65** selects the motion vector (displacement) with the smallest magnitude. For this selection purpose, magnitudes are defined in calculator **65** as the sum of the magnitudes of the horizontal and vertical displacement, i.e., $|r| + |s|$.

In the second stage of motion vector generator **13**, a refined determination is made as to the best displacement value that can be selected, within the neighborhood of the displacement selected in the first stage. The second stage differs from the first stage in three ways. First, it performs a search that is directed to a particular neighborhood. Second, it evaluates prediction error values for 8×8 blocks and a 4×2 array of 8×8 blocks (which in effect is a 32×16 block). And third, it interpolates the end result to ½ pixel accuracy.

FIG. **4** presents a general block diagram of the second stage of generator **13**. As in FIG. **3**, the input signal is applied to frame memory **66**. The input and the output of memory **66** are applied to motion estimator **67**, and the output of motion estimator **67** is applied to best match calculator **68**. Estimator **67** is also responsive to the coarse motion vector estimation developed in the first stage of generator **13**, whereby the estimator is caused to estimate motion in the neighborhood of the motion vector selected in the first stage of generator **13**.

Calculator **68** develops output sets with 10 signals in each set. It develops eight 8×8 block motion vectors, one 32×16 motion vector that encompass the image area covered by the eight 8×8 blocks, and a measure of the improvement in motion specification (i.e., a lower prediction error) that one would get by employing the eight 8×8 motion vectors in place of the associated 32×16 motion vector. The measure of improvement can be developed in any number of ways, but one simple way is to maintain the prediction errors of the 8×8 blocks, develop a sum of those prediction errors, and subtract the developed sum from the prediction error of the 32×16 motion vector.

The motion vector outputs of calculator **68** are applied in FIG. **4** to half pixel estimator **69**. Half pixel motion is deduced from the changes in the prediction errors around the region of minimum error. The simple approach used in estimator **69** is to derive the half pixel motion independently in the x and y directions by fitting a parabola to the three points around the minimum, solving the parabola equation, and finding the position of the parabola's minimum. Since all that is desired is ½ pixel accuracy, this process simplifies to performing the following comparisons:

$$\text{if } (p_{x-1} - p_x) < \frac{(p_{x+1} - p_x)}{3} \text{ then } x' = x - \frac{1}{2} \quad (2)$$

and

$$\text{if } (p_{x+1} - p_x) < \frac{(p_{x-1} - p_x)}{3} \text{ then } x' = x + \frac{1}{2} \quad (3)$$

where p_x is the prediction error at x, and x' is the deduced half pixel motion vector.

The searches in both stages of motion vector generator **13** can extend over the edges of the image to allow for the improved prediction of an object entering the frame. The values of the pixels outside the image should be set to equal the value of the closest known pixel.

The above describes the structure of motion vector generator **13**. All of the computational processes can be carried out with conventional processors. The processes that can most benefit from special purpose processors are the motion estimation processes of elements **64** and **67**; simply because of the number of operations called for. These processes, however, can be realized with special purpose chips from LSI Logic Corporation, which offers a video motion estimation processor (L64720). A number of these can be combined to develop a motion estimation for any sized block over and sized area. This combining of L64720 chips is taught in an LSI Logic Corporation Application Note titled "LG720 (MEP) Video Motion Estimation Processor".

MOTION VECTOR SELECTOR/ENCODER **14**

The reason for creating the 32×16 blocks is rooted in the expectation that the full set of motion vectors for the 8×8 blocks cannot be encoded in the bit budget allocated for the motion vectors. On the other hand, sending only 32×16 block motion vectors requires 28,672 bits—which results from multiplying the 14 bits per motion vector (7 bits for the horizontal displacement and 7 bits for the vertical displacement) by 32 blocks in the horizontal direction and 64 blocks in the vertical direction. In other words, it is expected that the final set of motion vectors would be a mix of 8×8 block motion vectors and 32×16 block motion vectors. It follows, therefore, that a selection must be made of the final mix of motion vectors that are eventually sent by the HDTV transmitter, and that selection must fit within a preassigned bit budget. Since the number of bits that define a motion vector depends on the efficacy of com-

pression encoding that may be applied to the motion vectors, it follows that the selection of motion vectors and the compression of the selected motion vectors are closely intertwined.

The basic philosphy in the encoding process carried out in selector/encoder **14** is to fill the bit budget with as much detailed information as is possible. That is, the aim is to use as many of the 8×8 motion vectors as is possible. Our approach is to start with the set of 32×16 block motion vectors, compress this sequence of motion vectors, develop a measure of the number of bits left over from the preassigned bit budget, and then enter an iterative process that allocates the leftover bits. To simplify the process, the 32×16 motion vectors are grouped into sets, and each set is coded as a unit. The sets, or slices, that we employ are 2 by 3 arrays, (six 32×16 motion vectors in all) as depicted in FIG. **5**.

The encoding process starts with encoder **14** (FIG. **1**) developing the codes for the first six 32×16 motion vectors described above and compressing those codes with a variable length code, such as a Huffman coder. The first code in the group specifies a first 32×16 motion vector in the slice in an absolute sense. Thereafter, the remaining 5 motion vectors are specified by the difference between the vectors and the first vector. Following the 32×16 motion vectors of the slice, a bit is included to indicate whether any of the 32×16 blocks in the slice are also specified by the more refined information of the 8×8 motion vectors. If so, a code is provided that specifies which of the blocks are so more refinely specified, and thereafter, up to 6 times 8, or 48, codes specify the included 8×8 motion vectors.

Thus, the encoded format of the slice (with reference to FIG. **5**) forms a packet as follows:

field	# of bits	description
1	1-16	motion vector C
2	1-16	motion vector A-C
3	1-16	motion vector B-C
4	1-16	motion vector D-C
5	1-16	motion vector E-C
6	1-16	motion vector F-C
7	1	subdivisions in slice
8	6	identify subdivisions
9+	1-14	8 × 8 motion vectors

FIG. **6** presents a block diagram of selector/encoder **14**. With each 32×16 motion vector that it receives from half pixel estimator **69**, it also receives the associated 8×8 motion vectors and the improvement information. The incoming 32×16 motion vectors are sent to Huffman coder **70**, the 8×8 motion vectors are sent to memory **71**, and the improvement information is sent to sorter **72**. Sorter **72** has an associated memory **73** for storing the relevant information about the improvement signals in sorted order, by the magnitude of the improvement; with the relevant information being the identity of the 32×16 block that is associated with the improvement value.

Huffman coder **70** develops a code for the first six fields of the slice packet and stores that information in memory **74**. Concurrently, it accumulates in accumulator **75** the bits generated in the coding process. The number of bits developed in accumulator **75** is compared to the preassigned bit budget (which is a fixed number) in threshold detector **76**. As long as that budget is not exceeded, threshold **76** sends a signal to memory **73** requesting it to access the identifier at the top of the sorted queue (and delete it from the queue). The identifier provided by memory **73** is applied to memory **71**, where it causes memory **71** to output the eight 8×8 motion vectors associated with the 32×16 block. The motion vectors delivered by memory **71** are applied to Huffman coder **70** which compresses that information, adds the compressed information to memory **74** (under control of the memory **73** output signal), and accumulates the compressed bits in accumulator **75**. This completes one iteration.

The operation of threshold circuit **76** and the entire iteration is repeated until the bit budget is exceeded. At that point, memory **74** contains the appropriate mix of coded motion vectors that are delivered to buffer **15** in FIG. **1**. As depicted in FIG. **6**, the uncoded motion vectors applied to Huffman circuit **70** are also applied to memory **77**. Those vectors are applied to motion compensator circuit **16** and to buffer **17** in FIG. **1**.

MOTION COMPENSATOR CIRCUITS

To develop its output signals, motion compensator circuit **16** (in FIG. **1**) merely needs to access data in buffer **19** under control of the motion vectors selected by motion vector selector/encoder **14**. This is easily achieved by having buffer **19** be a random access memory with multiplexed control. Output signals destined to motion compensator **16** are under control of the motion vectors, while output signals destined to buffer **51** are under control of a sequential counter.

Motion compensator circuit **43** (in FIG. **2**) is identical to motion compensator circuit **16**.

LEAK CIRCUIT

The leak circuit comprises leak processor **20** and multiplier **23**. Multiplier **23** modifies the signals of motion compensator circuit **16** prior to their application to subtracter **28**. Substracter **26** excludes the mean signal from considerations, in an effort to reduce the dynamic range of the signals considered in DCT transform circuits, **29** and **30**.

The processing within element **20** takes some time, of course, and therefore, FIG. **1** includes buffers **21** and **22**. Buffer **22** delays the image signal applied to subtracter **28**, and buffer **21** delays the motion vectors signals sent by motion compensator circuit **16**.

Directing attention to leak processor **20**, one way to look at the leak circuit is as a mechanism for reducing the DFD (displaced frame difference) developed at the output of subtracter **28**. The entire effort of developing good motion vectors, therefore, is to reduce the DFD out of subtracter **28**. To the extent that the leak circuit can reduce the DFD further, its employment is beneficial.

One way to reduce the DFD is to minimize the DFD as a function of the leak variable α. That is, the need is to determine α such that

$$\frac{\partial E\{(I - \alpha DF)^2\}}{\partial \alpha} = 0. \quad (4)$$

where I is the image frame signal of buffer **18**, DF is the displaced frame signal of motion compensator circuit **16**, α is the multiplicative leak value, and E{X} is the expected value of X. The solution to the above equation is

$$\alpha = \frac{E\{I \cdot DF\}}{E\{DF^2\}} \quad (5)$$

Thus, one embodiment for processor 20 merely computes α, in accordance with equation (5) in response to signals I(t-2) and DF(t-2). The computations performed in processor 20, in such a case, are simple multiplications and averaging (sum and divide), so they are not described any further herein. Suffice it to state that processor 20 may be a conventional arithmetic processor (e.g., a DSP integrated circuit).

Although the calculations performed in processor 20 are straight forward, they are still rather numerous (though not anywhere near as numerous as the calculations needed for selecting the motion vectors). A somewhat simpler calculation task can be assigned to processor 20 by observing the following.

Considering the limits to which the leak factor should be subjected, it is clear, for example, that the leak factor cannot be allowed to reach and stay at 1.0. Some leak of the actual image must always be present. Otherwise, a receiver that freshly tunes to the transmitter cannot construct the image signal because it lacks all historical information; i.e., it never has the correct "previous frame" information to which the motion vectors can be successfully applied. Also, a noise accepted by the receiver would never disappear. Thus, a maximum level must be set on the long term value of the leak factor; such as 15/16.

It is also clear that when there is a scene change, a leak factor of value 0 is best, because it completely discards the old scene and begins to build a new one. It may be noted that scene changes are relatively frequent in commercial TV programs. Setting α to 0 also helps in capturing the necessary historical information for freshly tuned-in receivers and for receivers who have corrupted historical information because of noise. Of course, the value of α should not be maintained at 0 for too long because that creates a heavy transmission burden.

In view of the above, in its simpler form, the process carried out by processor 20 need only determine the occurrence of a scene change and set α to 0 at every such occurrence. Thereafter, α may be incremented at a preselected rate with every frame so that after m frames, the value of α is allowed to reach α_{max} (e.g., 15/16). Of course, if there is another scene change within the m frames, α is again reset to 0 and the incrementing is restarted.

This simple approach for developing α can be implemented simply with a scene-change determining circuit, and an accumulator. The scene-change determining circuit may simply be a circuit that adds the magnitudes of the displaced frame difference signals; i.e., $\Sigma|I_{t-2}-DF_{t-2}|$. That provides a measure that, when compared to a threshold, determines whether a scene change has occurred. This is depicted in FIG. 7 where element 34 develops the sum signal $\Sigma|I_{t-2}-DF_{t-2}|$ and applies this signal to threshold circuit 58. The output of circuit 58 is applied to a disable lead of adder 59, which adds the value in threshold register 60 to the value in register 57. The output of register 57 is the leak factor, α.

A still another approach is to employ a fixed leak at the input to multiplier 23, and to develop a two level leak factor thereafter. By placing a processor 53 at the output of DCT transform 29 (in addition to the processor 53 at the output of DCT transform 30) two σ(t-4) signals are developed.

The leak factor that is sent to the encoding loop of FIG. 2 is selected based upon the two σ(t-4) signals developed. When there is a scene change, the two σ signals will not differ much because the frame will have a poor prediction, resulting in a high value of the DFD standard deviation. In fact, the deviation of the DFD might even be higher than the standard deviation of the original frame. In such a case (i.e., when the two σ signals differ by more than a chosen threshold), it is clear that a leak of 1 (no prediction) is to be selected and, accordingly, the leak factor $\alpha=0$ is sent to FIG. 2. When the two α signals do differ by more than the chosen threshold, then a fixed leak factor, such as $\alpha=15/16$ is sent to FIG. 2.

The block diagram of the forward estimation section of the encoder (i.e. FIG. 1), can be simplified somewhat by adopting this leak approach. This is shown in FIG. 16, where processor 20 is dropped, as well as a number of buffers. On the other hand, another processor 53 had to be added, a subtracter, a threshold device, and a selector that selected one of two sets of scale factors and standard deviation measures and either a leak factor of 0 or 15/16.

DCT TRANSFORM CIRCUITS

Transform circuits 29 and 30 (FIG. 1) and 37 and 40 (FIG. 2) develop two dimensional transform. Circuits 29, 30 and 37 convert time domain information into frequency domain information, and circuit 40 performs the inverse. The hardware for creating two dimensional transforms can follow the teachings of U.S. Pat. No. 4,829,465 issued May 9, 1989, titled "High Speed Cosine Transform".

PROCESSOR 53

Processor 53 computes scale factor signals S_{ij} and standard deviation signal σ for the frame. With an 8×8 DCT transform, there are 64 scale factor signals. The scale factor signals are developed by evaluating

$$S_{ij} = \sqrt{\frac{1}{K_1} \sum_{\text{all blocks}} s_{ij}^2} \quad (6)$$

where s_{ij} is the frequency ij coefficients produced by DCT element 30 and K_1 is the number of 8×8 blocks in the frame. The standard deviation, σ, is a single value for the frame and it corresponds to

$$\sigma = \sqrt{\frac{1}{K_2} \sum_{\text{all pixels}} S_{ij}^2} \quad (7)$$

where K_2 is the number of pixels in the frame. Of course, these calculations can be performed quite easily with a conventional processor, since they only require squaring, adding and taking the square root. Still, since the answer cannot be ascertained before the entire frame has been processed, the outputs of processor 53 are marked S_{ij}(t-4) and σ(t-4).

QUANTIZER-AND-VECTOR-SELECTOR (QVS) 38

DCT transform circuit 37 develops sets of 8×8 frequency domain coefficients. In order to simplify coding in QVS 38 which follows circuit 37, it is best to group

these sets; and one possible such grouping is a 2×2 array. This is illustrated in FIG. 8 where a 2×2 array of 8×8 sets of coefficients (left side of the FIG.) are combined to form a 64 cell superblock vector (right side of the FIG.), where each cell contains 4 members. A first level of simplification is introduced by placing a restriction that the coefficients in each cell can be quantized only with one quantizer. A degree of sophistication is introduced by allowing a different quantizer to be used for quantizing different cells. The number of different quantizers is limited to a preselected number, such as 4, to reduce the hardware and to simplify the encoding. This (side) information, which is the information that identifies the choices, or selections, must is sent to the receiver. The number of possible selections (patterns, or vectors) is 64^4. Defining this range of selections in binary form would require 128 bits of information, and that is still more bits than is desirable to use.

Consequently, a second level of simplification is introduced by arbitrarily electing to only employ a limited number of patterns with which to represent the superblock vector. That number may be, for example, 2048, which would require only 11 bits to encode. Variable length encoding might further reduce the amount. The 2048 selected patterns are stored in a vector codebook.

Viewing the task of choosing a codebook vector (i.e., quantization pattern) in somewhat more mathematical terms, for each transform coefficient that is quantized, a quantization error may be defined by

$$q=|x-Q(x)| \tag{7}$$

where Q(x) is the quantized value of the cell member x (the value obtained by first encoding x and then decoding x). For equal visibility of the errors throughout the image, it is desirable for this quantization error to be equal to some target distortion level, d, which is obtained from perceptual coder 49. It is expected, of course, that using the codebook vectors would not yield this target distortion level in all instances. Hence, we define a selection distortion error by

$$e=|q-d|. \tag{8}$$

The total selection error for a cell is the sum of the individual selection errors of equation 9 over the 2×2 array of the cell members; and the overall selection errors for the superblock vector is the sum of the total selection errors of the 64 individual cells.

The process for selecting the appropriate codebook vector considers each of the codebook vectors and selects the codebook vector that offers the lowest overall selection error. In cases where two different codebook vectors offer the same or almost the same overall selection error, the vector that is selected is the one that results in fewer bits when the superblock is quantized.

Before proceeding to describe in detail the block diagram of QVS 38, it is convenient to first describe the element within QVS 38 that evaluates the selection error. That element is depicted in FIG. 9.

FIG. 9 includes 3 parallel paths corresponding to each of the 3 nontrivial quantizers selected for the system ("drop the cell" is the fourth "quantizer", but it represents a trivial quantization schema). The paths are identical.

The scale factors derived in the forward estimation section of FIG. 1 are used to effectively match a set of standard quantizers to the data. This is done by first dividing the input signal in divider 90 by the scale factors. The scaled signals are then applied in each path to (quantizer) 81 and to subtracter 83. The quantized output data of encoder 81 is decoded in quantization decoder 82 and multiplied by the scale factors in multiplier 78. The result is the signal plus quantization noise. The quantization noise is isolated by subtracting the output of multiplier 78 from the original signal applied to divider 90. The subtraction is accomplished in subtracter 83. Of course, each of the encoder and decoder pairs in FIG. 9 employs a different quantizer.

The output of each subtracter 83 is the quantization error signal, |q| for the employed level of quantization. The global target distortion level, d, is subtracted from |q| in subtracter 84, and the results are added in accumulator 85. In accordance with equations 8 and 9, subtracter 83 and 84 are sign-magnitude subtracters that yield the magnitude of the difference. The output of each accumulator 85 provides the selection error for the cell received at the input of quantizer 81, based on the level of quantization of the associated quantizer 81.

The quantized signal of quantizer 81, which forms another output of the path, is also applied to rate calculator 86 which develops a measure of the number of bits that would be required to describe the signal of the incoming cell, if quantizer 81 were selected as the best quantizer to use. To summarize, each element 80 of FIG. 9 receives cell information and delivers a quantized signal, selection error signal and rate information for each of the possible quantization levels of the system.

The fourth path—which is not shown in FIG. 9—is the one that uses a "0 quantizer". It offers no quantized signal at all, a rate for this quantized signal that is 0, and an error signal that is equal to its input signal. The target distortion is subtracted from this and the absolute value accumulated as with the other quantizers.

FIG. 10 presents an embodiment of QVS 38. The input from DCT transfer element 37 (FIG. 2) is first applied to buffer 178 to create the superblock vectors (arrangement of cells as depicted at the right side of FIG. 8). The 64 cells of the superblock vector are each applied to a different one of the 64 blocks 80. As described in connection with FIG. 9, each block 80 develops a number of output triplets, with each triplet providing a quantized signal, a measure of the selection error of the cell in response to a given level of quantization, and the resulting number of bits for encoding the cell with that given level of quantization. The outputs of the 64 blocks 80 are each applied to 64 selector blocks 79.

The 64 selector blocks 79 are responsive to blocks 80 and also to codebook vector block 87. Block 87 contains the set of codebook vectors, which are the quantization selection patterns described above. In operation, block 87 cycles through its codebook vectors and delivers each of them, in parallel, to the 64 selector blocks. More particularly, each of the 64 elements of a vector codebook is fed to a different one of the 64 blocks 79. Under control of the applied codebook vector signals, each selection block 79 selects the appropriate one of the output triplets developed by its associated block 80 and applies the selected triplet to combiner circuit 88. The output of combiner circuit 88 is a sequence of quantized superblock signals, and the associated overall error and rate signals for the different codebook vectors that are

sequentially delivered by codebook block 87. These overall triplet signals are applied to threshold circuit 89.

Threshold circuit 89 is also responsive to codebook block 87. It maintains information pertaining to the identity of codebook vector that produced the lowest overall selection error level, the number of bits that this codebook vector requires to quantize the superblock vector and, of course, the quantized superblock vector itself. As indicated previously, in cases where two codebook vectors yield very similar overall selection error levels, the vector that is selected is the one that requires a fewer number of bits to quantize the superblock vector.

The process of threshold circuit 89 can be easily carried out within two hardware sections. In the first section, a comparison is made between the current lowest overall selection error and the overall selection error that is applied to threshold circuit 89. The overall selection error that is higher is arbitrarily caused to assume some maximum rate. The setting to a maximum rate is disabled when the applied overall selection error is equal or almost equal to the current lowest overall selection error. In the second section, a comparison is made between rates, and the overall selection error with the lower rate is selected as the new current lowest overall selection error. After all of the codebook vectors are considered, threshold circuit 89 outputs the identity of one (optimum) codebook vector and its values for the superblock vector delivered by buffer 178. The process then repeats for the next superblock.

From threshold circuit 89 the codebook vector identification and the quantized values of the superblock vector cells are applied to degradation decision circuit 115. As mentioned earlier, in a digital HDTV system that aims to communicate via terrestrial transmission, it is important to provide for graceful degradation; particularly since existing analog TV transmission in effect offers such graceful degradation.

Graceful degradation is achieved by sorting the information developed and designating different information for different treatment. The number of different treatments that one may wish to have is a designer's choice. In the system as depicted in FIG. 2, that number is two (encoders 46 and 47). The criterion for differentiating may be related to spatial resolution, to temporal resolution, or, for example, to dynamic range.

With a strictly temporal resolution approach, for example, the task of block 115 may be quite simple. The idea may be to simply allow every odd frame to have all, or a large portion, of its information designated for superior treatment, and every even frame to have most, if not all, of its information designated for poorer treatment. In accordance with such an approach, degradation decision circuit 115 need only know whether the frame is even or odd and need be able to assign proportions. During odd frames most of the signals are routed to variable length encoder 47 for superior treatment, and during even frames most of the signals are routed to variable length encoder 46 for poorer treatment.

For a spatial resolution approach to different treatment, what is desired is to give preferential treatment to the low frequency components in the image over the high frequency components in the image. This can be easily accomplished in the system depicted in FIG. 2 in one of two ways. Experimentally it has been determined that the codebook vectors of block 87 form a collection of vectors that can be ordered by the number of higher frequency components that are included in the vector. Based on that, degradation decision circuit 115 need only know which codebook vector is being sent (that information is available from threshold circuit 89) and direct its input information to either encoder 47 or 46, accordingly. Alternatively, the low frequency cells of all the vectors may be designated for preferential treatment and sent to encoder 47.

It may be noted in passing that the above discussion regarding what information is sent to encoder 47 and what information is sent to encoder 46 relates to the quantized signals of the superblock vectors, and not to the identity of the codebook vectors themselves. At least where the low frequency coefficients of all of superblock the vectors are sent to encoder 47, the identity of the codebook vectors must all be sent to encoder 47.

VARIABLE LENGTH ENCODERS 46 AND 47

Variable length encoders 46 and 47 may be conventional encoders, such as Huffman encoders. The reason two are used is because the information directed to encoder 47 needs to be encoded in a manner that will guarantee a better chance of an error-free reception of the encoded signal. Therefore, the encoding within encoder 47 may be different from the encoding within encoder 46.

Of course, the encoding per se may not be deemed enough to enhance the chances of error-free reception of the signals encoded in encoder 47. Additional encoding, therefore, may be carried out in BFF block 56, or even beyond block 56.

INVERSE QUANTIZER 39

Inverse quantizer 39 is responsive to the scale factors S_{ji}, but with respect to the quantized superblock signals, it is responsive only to that output of QVS 38 which is applied to variable length encoder 47. In other words, inverse quantizer 39 operates as if it has a very poor reception, with none of the signals from encoder 46 having reached it. A block diagram of block 39 is depicted in FIG. 11. As shown in FIG. 11, inverse quantizer 39 includes a vector codebook 102 that is identical to codebook 87 and a quantization decoder 103. With the direct aid of codebook 102, decoder 103 reverses the quantization effect of QVS 38. It is understood, of course, that the quantization noise introduced by the quantization encoder of QVS 38 cannot be reversed, but at least the general level of the initial signal is recovered. For example, say that the quantization steps within QVS 38 converted all levels from 0 to 8 to "0", all level between 8 and 16 to "1", all levels from 16 to 24 to "2", and all level from 24 to 32 to "3". For such an arrangement, inverse quantizer 39 may convert a "0" to 4, a "1" to 12, a "2" to 20 and a "3" to 28. An input of 22 would be quantized to "3" by QVS 38, and the quantized "3" would be inverse quantized in circuit 39 to 20. The quantization error level in this case would be 2.

Following the inverse quantization step within block 39, a correction step must be undertaken. Remembering that the input signals were scaled within QVS 38 prior to quantization, the opposite operation must be performed in block 39 and, accordingly, the output of decoder 103 is applied to multiplier 104 which is responsive to the scale factors S_{ij}. The output of multiplier 104 forms the output of inverse quantizer 39.

PERCEPTUAL CODER 49

Before proceeding to describe the details of perceptual coder 49, it should be pointed out that the choice of what functions are included in the forward estimation block of FIG. 1 or in BFF block 56 and what functions are included in the perceptual coder is somewhat arbitrary. As will be evident from the forward estimation below, some of the functions could easily be included in the forward estimation block or in BFF block 56.

The primary job of perceptual coder 49 is to assist QVS 38 to take an input image frame and represent it as well as possible using the number of bits allocated per frame. The process which produces the transformed displaced frame difference at the output of element 37 (FIG. 2) is designed to produce a representation which has less transform domain statistical redundancy. At that point, all of the information required to reconstruct the original input image still exits. It has only been transformed into representation that requires fewer bits to represent than the original. If there were enough bits available, it would be possible to produce a "coded" image that is bit for bit equivalent to the original.

Except for the most trivial of inputs, there are not enough bits available to represent that signal with that degree of fidelity. Therefore, the problem that must be solved is to produce a coded image that is as close to the original as possible, using the available bits, subject to the constraint that the output will be viewed by a human observer. That constraint is what perceptual coder 49 introduces via the target distortion signal that it sends to QVS 38. In other words, two constraints are imposed: 1) the result must be an essentially constant bit rate at the output of BFF 56, and 2) the error must be minimized, as perceived by a person.

One way of performing the bit assignment to assume a linear system and minimize the mean square error between the original image and coded image. This is the approach taken in many image coders. The problem with this approach is that the human visual system is not a linear system and it does not utilize a mean-square-error metric. The purpose of perceptual coder 49, therefore, is to provide perceptual thresholds for performing bit allocation based on the properties of the human visual system. Together, they form the target distortion level of FIGS. 9 and 10.

Perceptual thresholds provide a means of performing bit allocation so that the distortion present in the coded image appears, to a human observer, to be uniformly distributed. That is, though the use of perceptual thresholds the coded artifacts that are present will all have about the same visibility; and, if the bit rate of the system is reduced, the perceived quality of the coded image will generally drop without the creation of any obvious (i.e., localized) coding errors.

FIG. 12 presents a block diagram of perceptual coder 49. It includes perceptual threshold generator 93 responsive to the $\bar{I}_T(t\text{-}4)$ signal from the forward estimation block of FIG. 1, a rate processor 91 responsive to buffer fullness signal from the output buffer within BFF block 56, and a multiplier 92 for developing the target distortion signals in response to output signals of generator 93 and processor 91.

The output of the perceptual threshold generator 93 is a set of thresholds, one for each frequency domain element received by element 38 (FIG. 2), which give an estimate of the relative visibility of coding distortion at that spatial location and for that frequency band. As described in more detail below, these thresholds depend on the scene content of the original image and hence, the bit allocations adapt to variations in the input.

One image characteristic accounted for in generator 93 is frequency sensitivity. (In the context of this disclosure, we deal with transform domain coefficients, which are related to rapidity of changes in the image, and not with what is normally thought of as "frequency". In the next few paragraphs, however, it is believed helpful to describe what happens in terms of "frequency".) Frequency sensitivity exploits the fact that the visual systems modulation transfer function (MTF) is not flat. Relative visibility as a function of frequency starts at a reasonably good level, increases with frequency up to a peak as some frequency, and thereafter drops with frequency to below the relative visibility at low frequencies.

The MTF function means that more quantization error can be inserted at high frequencies than at low frequencies. Therefore, the perceptual thresholds for the high frequency subbands will be higher than those for the lower frequency subbands. The absolute frequency at which the peak visibility occurs depends on the size of the screen and the viewing distance. Approximately, however, the peak visibility occurs near the upper end of the second lowest frequency elements applied to QVS 38. Also, since the HVS is sensitive to low frequency flicker, the DC threshold is set to a value substantially less than that strictly required by the MTF.

Extending perceptual thresholds to textured input requires a definition of texture. "Texture" may be viewed as the amount of AC energy at a given location, weighted by the visibility of that energy. Actually, however, the HVS is very sensitive to distortion along edges, but much less sensitive to distortion across edges. This is accounted for by introducing the concept of directionality. Instead of computing a single texture estimate over all frequencies, separate estimates are made for horizontal, vertical, and diagonal components (RawHoriz, RawDiag and RawVert components) from which horizontal texture (HorizTex), diagonal texture (DiagTex) and vertical texture (VertTex) signals are developed in accordance with equations 10, 11 and 12.

$$\text{HorizTex} = \text{RawHoriz} + 0.50 \times \text{RawDiag} \tag{10}$$

$$\text{DiagTex} = 0.25 \times \text{RawHoriz} + \text{RawDiag} + 0.25 \times \text{RawVert} \tag{11}$$

$$\text{VerTex} = 0.5 \times \text{RawDiag} + \text{RawVert} \tag{12}$$

where

$$\text{RawHoriz} = \sum_{\substack{\text{over} \\ 8\times 8\text{window}}} MFT(0,j) \cdot \bar{T}_T^2(0,j), \tag{13}$$

which is a summation only over the top row of the 8×8 window;

$$\text{RawVert} = \sum_{\substack{\text{over} \\ 8\times 8\text{window}}} MFT(i,0) \cdot \bar{T}_T^2(i,0), \tag{14}$$

which is a summation only over the left column of the 8×8 window; and

$$\text{RawDiag} = \sum_{\substack{\text{over} \\ 8\times 8\text{window}}} MFT(i,j)\cdot \bar{I}_T^2(i,j),$$

where $i \neq 0$ and $j \neq 0$

which is a summation over the remaining frequency elements in the window. Actually, the summations may be restricted to the lowest frequency quadrant, since these coefficients contain over 90 percent of the energy in typical 8×8 windows.

The final component in generator **93** accounts for temporal masking. When, at a fixed location in the scene, there is a large change in image content between two frames, the HVS is less sensitive to high frequency details at that location in the current frame. By detecting the occurrence of large temporal differences, the perceptual thresholds at these locations can be increased for the current frame. This allows the bits that would have been allocated to the high frequency detail at these locations to be utilized for other portions of the image.

In FIG. **13**, the transformed image information (with the mean removed), $\bar{I}_T(t\text{-}4)$ is applied to adder **101**, to buffer **94**, to subtracter **95**, to current texture processor **96**, and to base threshold look-up table **111**. Adder **101** combines the (0,0) coefficients of $\bar{I}_T(T\text{-}4)$ with its mean signal, M(t-4) to produce a local brightness estimate and sends the results to brightness correction truncation circuit **97**. Thus, circuit **97** converts the (0,0) transform coefficients to local brightness correction control signals that are applied to brightness correction look-up table **110**.

The other 63 subbands are split into two data streams. One goes to frame buffer **94** and to subtracter **95**, where a temporal frame difference is developed. This difference is used to compute the temporal masking correction by applying the temporal frame difference to texture processor **98**. The other path is applied to texture processor **96** directly. Texture processors **96** and **98** each implement equations 10 to 15.

As depicted in FIG. **14**, a texture processor comprises a look-up table (LUT) **114** which receives the image signals $\bar{I}_T(x,y)$ of frame t-4 and develops the factors $MTF(x,y)\cdot\bar{I}_T^2(i,j)$ of the frame for equations 13, 14, and 15. Selector **105** routes each factor to either RawHoriz accumulator **106**, RawVert accumulator **107** or RawDiag accumulator **108**. The developed raw texture estimates are appropriately added in combiner **109** according to equations 10, 11 and 12 to result in the projected directional texture estimates.

Continuing with FIG. **13**, the two sets of projected directional texture estimates developed in processors **96** and **98** are passed though combiner **99** which develops a CombinedTexture signal in accordance with the following equation:

$$\text{CombinedTexture} = T_0\text{(Outputs of 96)} + T_1\text{(Outputs of 98)}. \quad (16)$$

where T_0 and T_1 are fixed weighting constants; typically, 0.5. The combined texture estimates are applied to a Mapping LUT **100**, which develops texture correction factors in accordance with:

$$\text{TextureCorrection} = 1 + K_1\log(1 + K_2\text{CombinedTexture}) \quad (17)$$

where **K1** is a constant, typically between 0.5 and 2, and **K2** is a constant typically between 1/1000 and 1/10000. This LUT converts the directional power domain texture estimates to amplitude domain threshold corrections.

The output of LUT **100** is applied to brightness correction look-up table **110**, which is also responsive to brightness correction signal of block **97**. This table multiplies the amplitude domain texture threshold corrections by the appropriate brightness correction factor. The output of look-up table **110** is finally applied to base threshold look-up table **111**, which modulates the image signal with the output of table **110** and thereby develops a perceptual threshold for each frequency element of each block of the frame signal $\bar{I}_T(t\text{-}4)$. The base threshold LUT takes the appropriate directional texture/brightness correction factor and applies a frequency dependent base threshold. That is, it develops signals $PT_{ij}(t\text{-}4)$, which are the 64 signals for each frame, one for each of the frequency coefficients supplied by DCT circuit **37** to QVS **38**.

As stated above, one of the goals of perceptual coder **49** is to insure that the rate of bits delivered by QVS **38** (through BFF **56**) to subsequent circuitry is essentially constant. This is accomplished by making sure that the fullness of the buffer within BFF **56** is controlled properly.

The buffer control within the encoder of FIG. **2** is based upon modifying a frame-wide target distortion within QVS **38**. If the buffer fills up to a point higher than some reference level, a larger target distortion is set to allow the buffer to lower its occupancy level. On the other hand, if the buffer fullness is lower than the reference level, then a lower target distortion is set.

Given a certain buffer fullness B_t the desired buffer fullness for the next frame can be formulated as

$$B_{p+1} = B_{ref} + (B_p - B_{ref}) \times k_0. \quad (13)$$

where B_{ref} is the desired level of buffer fullness, B_p is the buffer fullness at frame p, and k_0 is a buffer control parameter which is constant,

$$0 < k_0 < 1. \quad (14)$$

But,

$$B_{p+1} = B_p + R_{p+1} - R_{CH}. \quad (15)$$

where R_{p+1} is the number of bits coming into the buffer at frame p+1, and R_{CH} is the number of bits (channel capacity) that leave the buffer with each frame.

We have determined experimentally that one good target rate for a chosen distortion level D, $R_T(D)$, can be calculated (using the t-4 frame reference in FIG. **2**) and where T stands for "target", in accordance with equation 16.

$$R_T(D_T) = a + b\log\left(\frac{D_T}{\sigma_{t-4}}\right) \quad (16)$$

where the standard deviation σ is computed in processor **53** (FIG. **1**) and parameters a and b are computed from the two previous frames by

$$b = \min\left(b_{max} \cdot \frac{R_{t-5} - R_{t-6}}{\log\left(\frac{D_{t-5}}{\sigma_{t-5}}\right) - \log\left(\frac{D_{t-6}}{\sigma_{t-6}}\right)}\right) \quad (17)$$

and

$$a = R_{t-5} - b \cdot \log\left(\frac{D_{t-5}}{\sigma_{t-5}}\right). \quad (18)$$

The minimization operation in equation 17 is included merely to avoid instabilities for small values of the denominator. Using equations 16, 17 and 18, the target distortion is

$$D_T = \sigma_{t-4} e^{\frac{(RT-a)}{b}}. \quad (19)$$

Replacing the rate, R_T in equation 19 with the buffer fullness measures (with the aid of equation 15), yields

$$D_T = \sigma_{t-4}\left(\frac{D_{t-5}}{\sigma_{t-5}}\right) e^{\frac{\{(B_{t-5} - B_{ref}) \cdot k_0 + (B_{t-5} - B_{t-6})\}}{-b}} \quad (20)$$

The computation of equation 20 is performed in processor **91**. It requires the constant k_o, the constant B_{ref}, the current frame's σ value (σ_{t-4}), the previous frame's D value, σ value and B value (D(t-5), σ(t-5), and B(t-5)), and the B value before that (i.e., B(t-6)). The sigma values come from processor **53** (FIG. **1**) and the B values come from BFF block **56**. Of course, the various delayed images of B and σ are obtained with appropriate registers within processor **91**. The exponentiation and other computations that are called for by equation 20 can be either computed or derived from a look-up table.

The D value developed by processor **91** is applied to multiplier **92** to alter the perceptual thresholds developed in block **93**. The altered perceptual threshold signals are applied, as described above, to selectors **79** in FIG. **10**.

BUFFER FULLNESS AND FORMATTER 56

As indicated above, buffer fullness circuit **56** needs to supply perceptual coder **49** with the information on buffer fullness. Of course, that implies that block **56** includes a buffer which is filled. That indeed is the case. BFF block **56** accumulates the various segments of data that must be transmitted and forwards that data to modulation circuitry, power amplification circuitry, and the transmitting antenna.

To recap block **56** accepts the following signals:

1. The coded motion vectors CODED MV(t-4). These are a collection of Hoffman coded packets, where each packet describes the motion vectors of a slice, as described in detail in connection with FIGS. **5** and **6**.
2. Leak factor signals L(t-4).
3. Scaling factors S_{ij}.
4. Coded information from encoder **47**, which is the identity of the codebook vectors selected from codebook **87**, and the quantized superblock vectors.
5. Coded information from encoder **46** (much like information from encoder **47**).

As indicated above, the encoded information of encoder **47** is considered more important than the encoded information of encoder **46** and accordingly, only after the encoded information of encoder **47** is accepted and there is room left in the buffer of block **56**, will the information of encoder **46** be considered for inclusion. However, even with respect to the information of encoder **47**, the buffer of block **56** is filled with the more important information first. Buffer underflow in BFF **56** is handled by delivering dummy data to the following circuitry, in order to maintain a constant bit rate into the channel. This highly unlikely event is easily handled by simply retransmitting the data at the buffer's 0 address.

Buffer overflow is handled by simply not transmitting data that doesn't fit into the buffer, in which case it is advisable to drop the least significant data first. By "drop" we mean not transmit some of the data that is in the buffer and empty the buffer for the next frame's data, and not load the buffer with new data that is of low importance. Of course, the buffer fullness measurement in combination with the perceptual thresholds are combined in perceptual block **49** to form a global target distortion level that will allow the output buffer of block **56** to be populated with all of the data that is generated; including the data of encoder **46**. The primary consequence of the encoding within encoder is to allow more sophisticated encoding in the data of encoder **47** which, in turn, enhances the receivability of those signals.

The information received from sources other than encoder **46** need to be transmitted in a manner that is likely to be received correctly. That means that the formatter section of block **56** must encode the information is a manner that will ensure that. This can be accomplished with conventional coder means that incorporates error correcting codes. The signals derived from other than encoder **46** are encoded with powerful error correcting codes, while the signals received from encoder **46** are encoded with less powerful error correcting codes (or perhaps no error correcting codes.

In a copending application, Ser. No. 07/611,225 a system is disclosed for encoding signals using the concepts of code constellations. Code constellations can be formed with some codes in the constellations having a large Hamming distance from all other codes, while other codes in the constellations have a smaller Hamming distance from other codes. The principles of such coding can be advantageously incorporated within the formatter section of block **56**, or in circuitry beyond block **56** to achieve the goals of graceful degradation.

HDTV RECEIVER'S DECODER

FIG. **15** present's a block diagram of an HDTV receiver that conforms to the HDTV transmitter encoder described above. It receives the signal, e.g., from an antenna, and decodes i in block **211** to yield the signals loaded into BFF block **56** within the transmitter. These signals are the codebook vector identifiers, the quantized superblock vector signals, the leak factors, the scaling factors, the frame means, and the motion vectors. The receptions of these signals, their separation from the combined received signal, the error code verifications, and the decoding of the variable length codes are all performed in block **211**.

The processing in the decoder begins with the codebook vector identifiers applied to a codebook vector

201, and the quantized vector signals and the scale factors applied to quantization decoder 202. Blocks 201 and 202 correspond to blocks 102 and 103, respectively, of FIG. 11, and together they form an inverse quantization element akin to element 39 of FIG. 2. As in FIG. 2, the output of the inverse quantization element is applied to an inverse DCT transform circuit (in FIG. 15, this is circuit 203); and that output is combined in adder 204 with signals already stored in the decoder.

Since the quantized vector signals of a frame were created from image signals with the frame mean deleted, the output of adder 204 is missing the frame mean. This is remedied in adder 205, which adds the frame mean. The output signals of adder 205 form the frame output of the HDTV receiver's decoder. This output is applied to amplifier-and-display circuit 212 and to frame buffer circuit 206, where one frame's worth of information is stored. For each frame that is stored in buffer circuit 206, buffer circuit 206 outputs the previous frame. The previous frame signal is augmented in motion compensation block 207 which, in response to the applied motion signals, forms an estimate of the current frame. Motion compensation block 207 is identical to motion compensation block 43 in FIG. 2. The frame mean is substracted from the output of motion compensation block 207 by substracting therefrom the previous frame's mean in substracter 209. The previous frame's mean is obtained from buffer 208, into which the current frame's mean is inserted. Finally, the output of subtracter 209 is applied to multiplier 210, which multiplies that signal by the leak factor signal. The output of multiplier 210 is the signal that is employed in adder 204 as described above.

We claim:

1. An encoder including a coder for developing encoder output signals from frame difference signals, prediction means responsive to said encoder output signals for predicting a next frame's signals, and means for developing said frame difference signals from applied next frame signals of an image frame and from output signals of said prediction means, the improvement comprising:

said coder including controllable quantizer means that quantizes said difference signals in accordance with a quantization schema that varies with the dictates of a control signal; and

said coder including means, responsive to said applied next frame signals, to develop said control signal, which control signal varies throughout said applied next frame with changes in at least one selected characteristic of said applied next frame signals.

2. The encoder of claim 1 wherein said control signal is further responsive to a selected bit rate target level for said encoder output signals.

3. The encoder of claim 2 wherein said control signal forms a selected quantization error target level for said encoder output signals.

4. The encoder of claim 1 wherein said selected characteristic is a measure of texture in said applied next frame signals.

5. The encoder of claim 4 wherein said measure of texture is a combination of texture measured horizontally, texture measured vertically, and texture measured over a selected area of said image frame.

6. The encoder of claim 4 wherein said measure of texture is a combination of a texture measure of said applied next frame signals and of previously applied next frame signals.

7. The encoder of claim 1 wherein said selected characteristic is a measure of brightness in said applied nex frame signals.

8. The encoder of claim 1 further comprising an output buffer for receiving said encoder output signals, and said coder comprising means for receiving signals from said output buffer that indicate the level of buffer fullness of said output buffer.

9. The encoder of claim 8 wherein selected characteristic is a measure of buffer fullness of said output buffer.

10. The encoder of claim 8 wherein selected characteristic is a combination of said buffer fullness of said output buffer, brightness of said applied next frame signals and texture of said applied next frame signals.

11. The encoder of claim 8 wherein said characteristic is related to the buffer fullness of said output buffer in the previous frame.

12. The encoder of claim 8 wherein said characteristic is related to the buffer fullness of said output buffer in the previous frame and in the frame previous to the previous frame.

13. The encoder of claim 1 wherein said coder is responsive to applied scale factor signals to control the effective range of said quantizer means.

14. The encoder of claim 1 further comprising:

forward estimation means including means for processing said next frame signals to develop said control signal within a processing interval; and

means for delaying said next frame signals by said processing interval.

15. The encoder of claim 14 wherein said forward estimation means further includes means for developing said measure of brightness within said processing interval.

16. The encoder of claim 14 wherein said forward estimation means further includes means for developing said measure of texture within said processing interval.

17. The encoder of claim 1 wherein said next frame signals comprise a sequence of signal sections, each of which is related to a transform of at least one block of said frame difference signals, and each of which including a collection of N transform element signals, and said control signal comprising N control signal cells, where N in a constant, and each control signal cell controls the quantization schema for a different one of said N transform element signals.

18. The encoder of claim 17 wherein said control signal having N control signal cells is a control signal vector that is selected from a look-up table that contains a collection of control signal vectors.

19. The encoder of claim 18 wherein the selection from said look-up table is based on an error level that is obtained by coding said frame difference signal under control of each vector and said selected characteristic, and by developing a quantization error signal that is subtracted from a quantization error target level.

20. The encoder of claim 19 wherein said selected characteristic is scale factors that are related to the power at various subbands in said next frame's signals.

21. The encoder of claim 19 wherein said quantization error target level is related to signal characteristics of said next frame's signals.

22. The encoder of claim 19 wherein said quantization error target level is further related to buffer fullness of a buffer that stores said encoder output signals.

23. The encoder of claim 19 wherein said selection is further based on the number of bits created by said coder quantizer means.

24. The encoder of claim 18 wherein the selection from said look-up table is based on an error level that is obtained by scaling said frame difference signal, coding the scaled frame difference signal under control of each vector and said selected characteristic and by developing a quantization error signal that is subtracted in an absolute sense from a quantization error target level.

25. The encoder of claim 24 wherein said selected characteristic is related to texture of said next frame signal.

26. An encoder comprising:

prediction means responsive to output signals of said encoder, for developing frame prediction signals;

means for developing frame difference signals in response to said frame prediction means and applied frame signals;

coder means, responsive to said frame difference signals and to a control signal, for encoding frame difference signals under direction of said control signal, where said coder means codes different portions of said frame difference signals with different coding schemas, where different coding schemas yield different numbers of bits when coding any given signal, said coder means thereby generates a number of bits when encoding said applied frame signals; and

control means for developing said control signal in response to said encoder output signals, to control the number of bits generated by said coder means while encoding said applied frame signals.

27. The encoder of claim 26 wherein said control means develops a control signal that controls said coder means to develop a number of bits for each frame of said applied frame signals that approaches a preselect number.

28. The encoder of claim 26 wherein said control means develops a control signal that controls said coder means to develop a number of bits for each frame of said applied frame signals that on average approaches a preselect number.

29. The encoder of claim 26 where said control means is further responsive to said applied frame signals and modifies its developed control signal based on said applied frame signals to control coding error signals created in said coder in the course of coding of said frame difference signals.

30. The encoder of claim 27 where said control means modifies said control signal in response to said applied frame signal to equalize coding error signals throughout a frame of said applied frame signals.

31. The encoder of claim 27 where said control means develops said control signal in accordance with a visual perception model for humans, to cause the creation of encoder outputs signals that, when said encoder output signals are decoded and a frame image is created and displayed, the coding error signals found in said created and displayed frame image are perceived, in accordance with said visual perception model, to be essentially equally visible throughout said frame.

* * * * *

JS 44 (Rev. 12/07)

CIVIL COVER SHEET

The JS 44 civil cover sheet and the information contained herein neither replace nor supplement the filing and service of pleadings or other papers as required by law, except as provided by local rules of court. This form, approved by the Judicial Conference of the United States in September 1974, is required for the use of the Clerk of Court for the purpose of initiating the civil docket sheet. (SEE INSTRUCTIONS ON THE REVERSE OF THE FORM.)

2009 JUN 25 PM 4:06

CLERK US DISTRICT COURT
SOUTHERN DISTRICT OF CALIFORNIA
BY DEPUTY

I. (a) PLAINTIFFS
Multimedia Patent Trust
991 Route 22 West, 3rd Floor, Bridgewater, NJ 08807

(b) County of Residence of First Listed Plaintiff Somerset County
(EXCEPT IN U.S. PLAINTIFF CASES)

(c) Attorney's (Firm Name, Address, and Telephone Number)
John Kyle, Cooley Godward Kronish LLP, 4401 Eastgate Mall
San Diego, California (858)550-6000

DEFENDANTS
Tandberg, Inc.

County of Residence of First Listed Defendant Fairfax County
(IN U.S. PLAINTIFF CASES ONLY)
NOTE: IN LAND CONDEMNATION CASES, USE THE LOCATION OF THE LAND INVOLVED.

'09 CV 1377 JM WMc

II. BASIS OF JURISDICTION (Place an "X" in One Box Only)

- ☐ 1 U.S. Government Plaintiff
- ☒ 3 Federal Question (U.S. Government Not a Party)
- ☐ 2 U.S. Government Defendant
- ☐ 4 Diversity (Indicate Citizenship of Parties in Item III)

III. CITIZENSHIP OF PRINCIPAL PARTIES (Place an "X" in One Box for Plaintiff and One Box for Defendant)
(For Diversity Cases Only)

	PTF	DEF		PTF	DEF
Citizen of This State	☐ 1	☐ 1	Incorporated *or* Principal Place of Business In This State	☐ 4	☐ 4
Citizen of Another State	☐ 2	☐ 2	Incorporated *and* Principal Place of Business In Another State	☐ 5	☐ 5
Citizen or Subject of a Foreign Country	☐ 3	☐ 3	Foreign Nation	☐ 6	☐ 6

IV. NATURE OF SUIT (Place an "X" in One Box Only)

☐ 110 Insurance	PERSONAL INJURY	PERSONAL INJURY	☐ 610 Agriculture	☐ 422 Appeal 28 USC 158	☐ 400 State Reapportionment
☐ 120 Marine	☐ 310 Airplane	☐ 362 Personal Injury - Med. Malpractice	☐ 620 Other Food & Drug	☐ 423 Withdrawal 28 USC 157	☐ 410 Antitrust
☐ 130 Miller Act	☐ 315 Airplane Product Liability	☐ 365 Personal Injury - Product Liability	☐ 625 Drug Related Seizure of Property 21 USC 881		☐ 430 Banks and Banking
☐ 140 Negotiable Instrument	☐ 320 Assault, Libel & Slander	☐ 368 Asbestos Personal Injury Product Liability	☐ 630 Liquor Laws	☐ 820 Copyrights	☐ 450 Commerce
☐ 150 Recovery of Overpayment & Enforcement of Judgment	☐ 330 Federal Employers' Liability	PERSONAL PROPERTY	☐ 640 R.R. & Truck	☒ 830 Patent	☐ 460 Deportation
☐ 151 Medicare Act	☐ 340 Marine	☐ 370 Other Fraud	☐ 650 Airline Regs.	☐ 840 Trademark	☐ 470 Racketeer Influenced and Corrupt Organizations
☐ 152 Recovery of Defaulted Student Loans (Excl. Veterans)	☐ 345 Marine Product Liability	☐ 371 Truth in Lending	☐ 660 Occupational Safety/Health		☐ 480 Consumer Credit
☐ 153 Recovery of Overpayment of Veteran's Benefits	☐ 350 Motor Vehicle	☐ 380 Other Personal Property Damage	☐ 690 Other		☐ 490 Cable/Sat TV
☐ 160 Stockholders' Suits	☐ 355 Motor Vehicle Product Liability	☐ 385 Property Damage Product Liability	☐ 710 Fair Labor Standards Act	☐ 861 HIA (1395ff)	☐ 810 Selective Service
☐ 190 Other Contract	☐ 360 Other Personal Injury		☐ 720 Labor/Mgmt. Relations	☐ 862 Black Lung (923)	☐ 850 Securities/Commodities/ Exchange
☐ 195 Contract Product Liability			☐ 730 Labor/Mgmt.Reporting & Disclosure Act	☐ 863 DIWC/DIWW (405(g))	☐ 875 Customer Challenge 12 USC 3410
☐ 196 Franchise			☐ 740 Railway Labor Act	☐ 864 SSID Title XVI	☐ 890 Other Statutory Actions
			☐ 790 Other Labor Litigation	☐ 865 RSI (405(g))	☐ 891 Agricultural Acts
☐ 210 Land Condemnation	☐ 441 Voting	☐ 510 Motions to Vacate Sentence	☐ 791 Empl. Ret. Inc. Security Act		☐ 892 Economic Stabilization Act
☐ 220 Foreclosure	☐ 442 Employment	**Habeas Corpus:**		☐ 870 Taxes (U.S. Plaintiff or Defendant)	☐ 893 Environmental Matters
☐ 230 Rent Lease & Ejectment	☐ 443 Housing/ Accommodations	☐ 530 General		☐ 871 IRS—Third Party 26 USC 7609	☐ 894 Energy Allocation Act
☐ 240 Torts to Land	☐ 444 Welfare	☐ 535 Death Penalty	☐ 462 Naturalization Application		☐ 895 Freedom of Information Act
☐ 245 Tort Product Liability	☐ 445 Amer. w/Disabilities - Employment	☐ 540 Mandamus & Other	☐ 463 Habeas Corpus - Alien Detainee		☐ 900Appeal of Fee Determination Under Equal Access to Justice
☐ 290 All Other Real Property	☐ 446 Amer. w/Disabilities - Other	☐ 550 Civil Rights	☐ 465 Other Immigration Actions		☐ 950 Constitutionality of State Statutes
	☐ 440 Other Civil Rights	☐ 555 Prison Condition			

V. ORIGIN (Place an "X" in One Box Only)

☒ 1 Original Proceeding ☐ 2 Removed from State Court ☐ 3 Remanded from Appellate Court ☐ 4 Reinstated or Reopened ☐ 5 Transferred from another district (specify) ☐ 6 Multidistrict Litigation ☐ 7 Appeal to District Judge from Magistrate Judgment

VI. CAUSE OF ACTION
Cite the U.S. Civil Statute under which you are filing (**Do not cite jurisdictional statutes unless diversity**):
35 U.S.C. section 1 et seq.
Brief description of cause:
Patent Infringement

VII. REQUESTED IN COMPLAINT:
☐ CHECK IF THIS IS A CLASS ACTION UNDER F.R.C.P. 23
DEMAND $
CHECK YES only if demanded in complaint:
JURY DEMAND: ☒ Yes ☐ No

VIII. RELATED CASE(S) IF ANY (See instructions): JUDGE Unassigned DOCKET NUMBER 1:09-cv-456 (D. Dela.) 3:09-cv-0278

DATE 6/25/2009

SIGNATURE OF ATTORNEY OF RECORD

FOR OFFICE USE ONLY

RECEIPT # 2355 AMOUNT 350.- APPLYING IFP JUDGE MAG. JUDGE

6/25/09

JS 44 Reverse (Rev. 12/07)

INSTRUCTIONS FOR ATTORNEYS COMPLETING CIVIL COVER SHEET FORM JS 44

Authority For Civil Cover Sheet

The JS 44 civil cover sheet and the information contained herein neither replaces nor supplements the filings and service of pleading or other papers as required by law, except as provided by local rules of court. This form, approved by the Judicial Conference of the United States in September 1974, is required for the use of the Clerk of Court for the purpose of initiating the civil docket sheet. Consequently, a civil cover sheet is submitted to the Clerk of Court for each civil complaint filed. The attorney filing a case should complete the form as follows:

I. (a) Plaintiffs-Defendants. Enter names (last, first, middle initial) of plaintiff and defendant. If the plaintiff or defendant is a government agency, use only the full name or standard abbreviations. If the plaintiff or defendant is an official within a government agency, identify first the agency and then the official, giving both name and title.

(b) County of Residence. For each civil case filed, except U.S. plaintiff cases, enter the name of the county where the first listed plaintiff resides at the time of filing. In U.S. plaintiff cases, enter the name of the county in which the first listed defendant resides at the time of filing. (NOTE: In land condemnation cases, the county of residence of the "defendant" is the location of the tract of land involved.)

(c) Attorneys. Enter the firm name, address, telephone number, and attorney of record. If there are several attorneys, list them on an attachment, noting in this section "(see attachment)".

II. Jurisdiction. The basis of jurisdiction is set forth under Rule 8(a), F.R.C.P., which requires that jurisdictions be shown in pleadings. Place an "X" in one of the boxes. If there is more than one basis of jurisdiction, precedence is given in the order shown below.

United States plaintiff. (1) Jurisdiction based on 28 U.S.C. 1345 and 1348. Suits by agencies and officers of the United States are included here.

United States defendant. (2) When the plaintiff is suing the United States, its officers or agencies, place an "X" in this box.

Federal question. (3) This refers to suits under 28 U.S.C. 1331, where jurisdiction arises under the Constitution of the United States, an amendment to the Constitution, an act of Congress or a treaty of the United States. In cases where the U.S. is a party, the U.S. plaintiff or defendant code takes precedence, and box 1 or 2 should be marked.

Diversity of citizenship. (4) This refers to suits under 28 U.S.C. 1332, where parties are citizens of different states. When Box 4 is checked, the citizenship of the different parties must be checked. (See Section III below; federal question actions take precedence over diversity cases.)

III. Residence (citizenship) of Principal Parties. This section of the JS 44 is to be completed if diversity of citizenship was indicated above. Mark this section for each principal party.

IV. Nature of Suit. Place an "X" in the appropriate box. If the nature of suit cannot be determined, be sure the cause of action, in Section VI below, is sufficient to enable the deputy clerk or the statistical clerks in the Administrative Office to determine the nature of suit. If the cause fits more than one nature of suit, select the most definitive.

V. Origin. Place an "X" in one of the seven boxes.

Original Proceedings. (1) Cases which originate in the United States district courts.

Removed from State Court. (2) Proceedings initiated in state courts may be removed to the district courts under Title 28 U.S.C., Section 1441. When the petition for removal is granted, check this box.

Remanded from Appellate Court. (3) Check this box for cases remanded to the district court for further action. Use the date of remand as the filing date.

Reinstated or Reopened. (4) Check this box for cases reinstated or reopened in the district court. Use the reopening date as the filing date.

Transferred from Another District. (5) For cases transferred under Title 28 U.S.C. Section 1404(a). Do not use this for within district transfers or multidistrict litigation transfers.

Multidistrict Litigation. (6) Check this box when a multidistrict case is transferred into the district under authority of Title 28 U.S.C. Section 1407. When this box is checked, do not check (5) above.

Appeal to District Judge from Magistrate Judgment. (7) Check this box for an appeal from a magistrate judge's decision.

VI. Cause of Action. Report the civil statute directly related to the cause of action and give a brief description of the cause. **Do not cite jurisdictional statutes unless diversity.** Example: U.S. Civil Statute: 47 USC 553
Brief Description: Unauthorized reception of cable service

VII. Requested in Complaint. Class Action. Place an "X" in this box if you are filing a class action under Rule 23, F.R.Cv.P.

Demand. In this space enter the dollar amount (in thousands of dollars) being demanded or indicate other demand such as a preliminary injunction.

Jury Demand. Check the appropriate box to indicate whether or not a jury is being demanded.

VIII. Related Cases. This section of the JS 44 is used to reference related pending cases if any. If there are related pending cases, insert the docket numbers and the corresponding judge names for such cases.

Date and Attorney Signature. Date and sign the civil cover sheet.

Court Name: USDC California Southern
Division: 3
Receipt Number: CAS002355
Cashier ID: sramirez
Transaction Date: 06/25/2009
Payer Name: AMERICAN MESSENGER SVCS

CIVIL FILING FEE
For: MULTIMEDIA PATENT V. TANDBERG
Case/Party: D-CAS-3-09-CV-001377-001
Amount: $350.00

CHECK
Check/Money Order Num: 1910
Amt Tendered: $350.00

Total Due: $350.00
Total Tendered: $350.00
Change Amt: $0.00

There will be a fee of $45.00
charged for any returned check.