United States Patent [19]

Keith et al.

[11] **Patent Number: 4,785,349**

[45] **Date of Patent: Nov. 15, 1988**

[54] **DIGITAL VIDEO DECOMPRESSION SYSTEM**

[75] Inventors: **John M. Keith**, Washington Crossing, Pa.; **Stuart J. Golin**, East Windsor, N.J.; **Allen H. Simon**, Belle Mead, N.J.; **Brian Astle**, Cranbury, N.J.

[73] Assignee: **Technology Inc. 64**, Princeton, N.J.

[21] Appl. No.: **104,131**

[22] Filed: **Oct. 5, 1987**

[51] **Int. Cl.**[4] **H04N 7/13**

[52] **U.S. Cl.** **358/136;** 358/11; 358/13; 358/133; 358/135

[58] **Field of Search** 358/133, 136, 135, 105, 358/13, 11; 375/27, 122

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,225,885	9/1980	Lux et al.	340/146.3
4,334,246	6/1982	Saran	358/261
4,386,366	5/1983	Mori	358/135
4,430,670	2/1984	Netrravali	358/135
4,468,708	8/1984	Coleman	358/310
4,520,401	5/1985	Takahashi et al.	358/310
4,546,342	10/1985	Weaver et al.	340/347
4,667,233	5/1987	Furukawa	358/136
4,691,329	9/1987	Juri	358/136

OTHER PUBLICATIONS

Leonardi & Kunt, "Adaptive Split and Merge for Image Analysis and Coding", Image Coding, Cannes, France, Dec. 1985 vol. 594 pp. 2–9, 1986.

Leonardi & Kunt, "Adaptive Split for Image Coding", International Symposium On Applied Signal Proc. and Digital Filtering, Paris 1985.

Milford et al., "Quad Encoded Display", IEEE Proceedings, vol. 131, Pt. E, No. 3, May 1984, pp. 70–75.

Baker & Gray "Image Compression Using Non-Adaptive Spatial Vector Quantization", IEEE Cat. #CH1809-3183/0000/0055 Conf. Paper in 16th Asilomar Conf. On Circuits, Sys. and Computers, Ca. 11/82 IEEE.

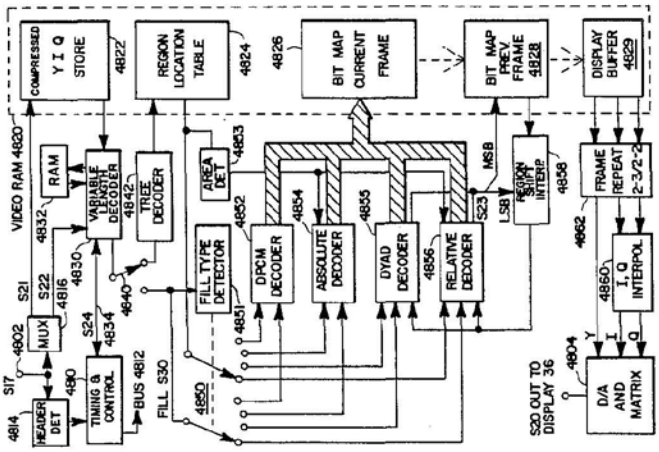
Primary Examiner—Howard W. Britton

Attorney, Agent, or Firm—Paul J. Rasmussen; Eric P. Herrmann; Kenneth N. Nigon

[57] **ABSTRACT**

A full motion color digital video signal is compressed, formatted for transmission, recorded on compact disc media and decoded at conventional video frame rates. During compression, regions of a frame are individually analyzed to select optimum fill coding methods specific to each region. Region decoding time estimates are made to optimize compression thresholds. Region descriptive codes conveying the size and locations of the regions are grouped together in a first segment of a data stream. Region fill codes conveying pixel amplitude indications for the regions are grouped together according to fill code type and placed in other segments of the data stream. The data stream segments are individually variable length coded according to their respective statistical distributions and formatted to form data frames. The number of bytes per frame is dithered by the addition of auxiliary data determined by a reverse frame sequence analysis to provide an average number selected to minimize pauses of the compact disc during playback thereby avoiding unpredictable seek mode latency periods characteristic of compact discs. A decoder includes a variable length decoder responsive to statistical information in the code stream for separately variable length decoding individual segments of the data stream. Region location data is derived from region descriptive data and applied with region fill codes to a plurality of region specific decoders selected by detection of the fill code type (e.g., relative, absolute, dyad and DPCM) and decoded region pixels are stored in a bit map for subsequent display.

15 Claims, 32 Drawing Sheets



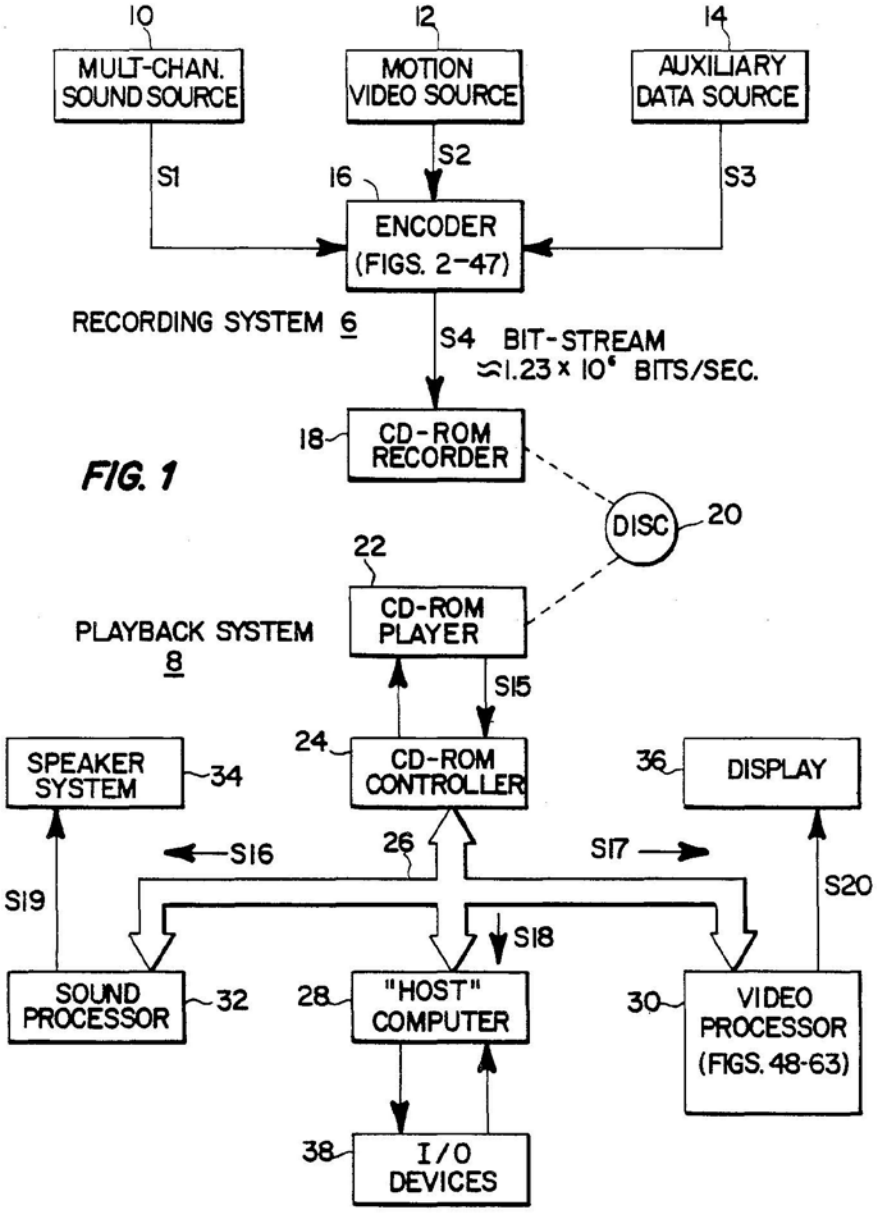

10
MULT-CHAN. SOUND SOURCE
12
MOTION VIDEO SOURCE
14
AUXILIARY DATA SOURCE
S1
S2
S3
16
ENCODER (FIGS. 2-47)
RECORDING SYSTEM 6
S4 BIT-STREAM ≃1.23 x 10⁶ BITS/SEC.
18
CD-ROM RECORDER
DISC
20
22
CD-ROM PLAYER
PLAYBACK SYSTEM 8
S15
24
CD-ROM CONTROLLER
SPEAKER SYSTEM
34
36
DISPLAY
S16
26
S17
S19
S20
S18
SOUND PROCESSOR
32
28
"HOST" COMPUTER
30
VIDEO PROCESSOR (FIGS. 48-63)
38
I/O DEVICES

FIG. 1

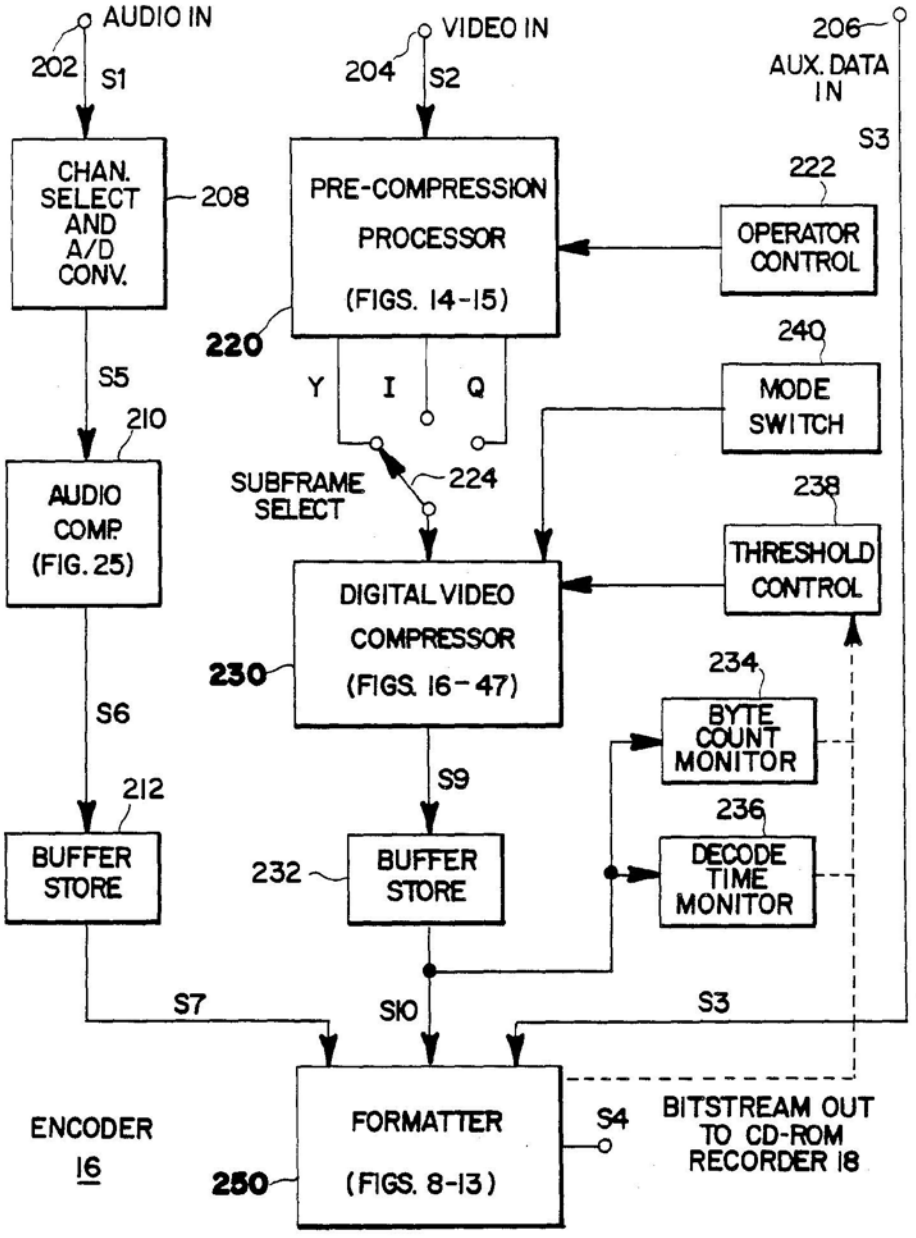

AUDIO IN
202
S1
VIDEO IN
204
S2
206
AUX. DATA IN
S3
CHAN. SELECT AND A/D CONV.
208
PRE-COMPRESSION PROCESSOR
(FIGS. 14-15)
220
222
OPERATOR CONTROL
240
MODE SWITCH
S5
210
AUDIO COMP. (FIG. 25)
Y
I
Q
SUBFRAME SELECT
224
238
THRESHOLD CONTROL
DIGITAL VIDEO COMPRESSOR
(FIGS. 16-47)
230
234
BYTE COUNT MONITOR
S6
S9
212
236
BUFFER STORE
232
BUFFER STORE
DECODE TIME MONITOR
S7
S10
S3
ENCODER
16
FORMATTER
(FIGS. 8-13)
250
S4
BITSTREAM OUT TO CD-ROM RECORDER 18

FIG. 2

FIG. 3

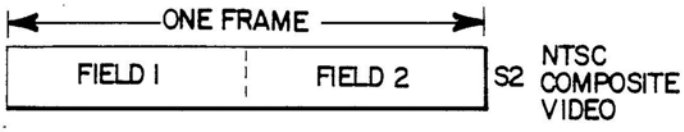


FIG. 4

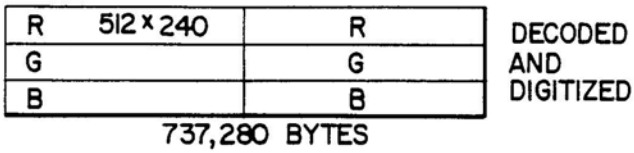


FIG. 5

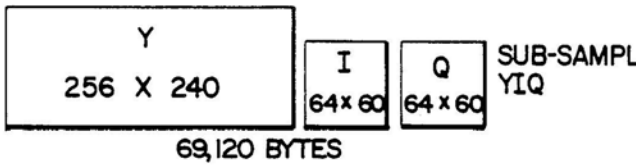


FIG. 6

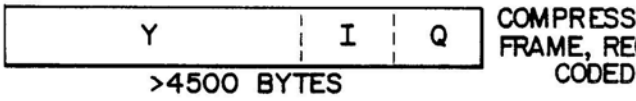


FIG. 7

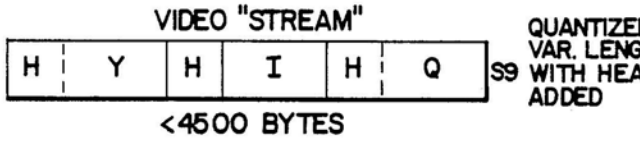


FIG. 8

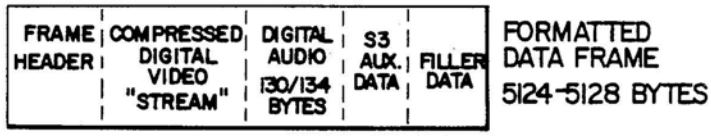


FIG. 9

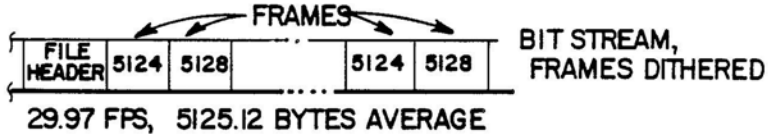


FIG. 10

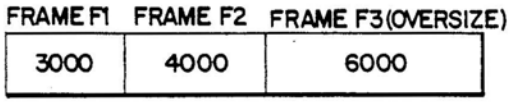


FIG. 11

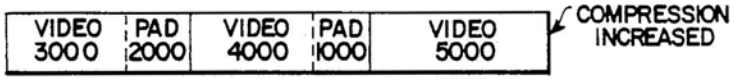
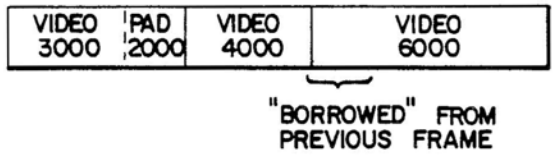


FIG. 12



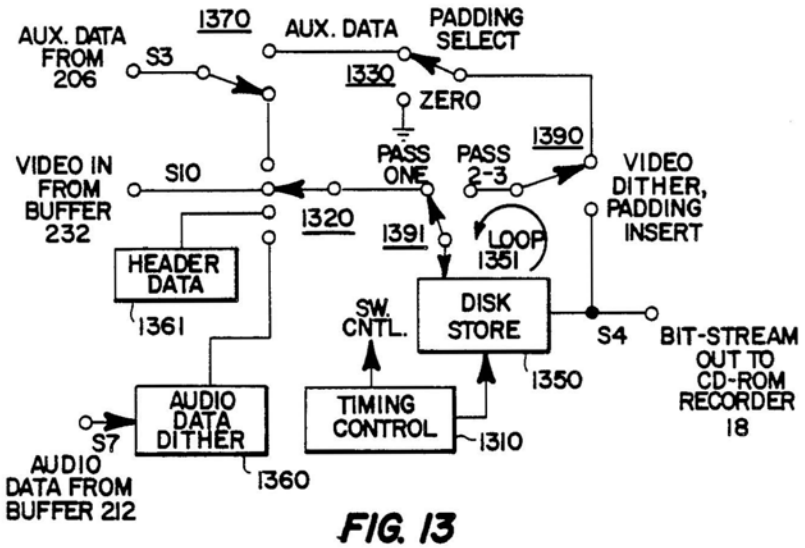


FIG. 13

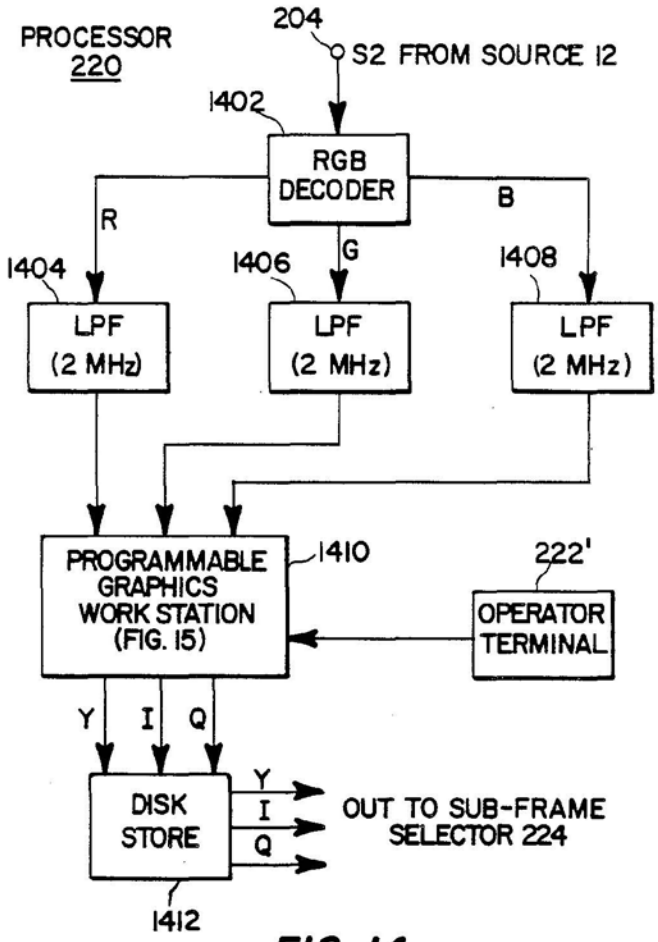

PROCESSOR
220
204
S2 FROM SOURCE 12
1402
RGB
DECODER
R
G
B
1404
1406
1408
LPF
(2 MHz)
LPF
(2 MHz)
LPF
(2 MHz)
PROGRAMMABLE
GRAPHICS
WORK STATION
(FIG. 15)
1410
222'
OPERATOR
TERMINAL
Y
I
Q
DISK
STORE
Y
I
Q
OUT TO SUB-FRAME
SELECTOR 224
1412

FIG. 14

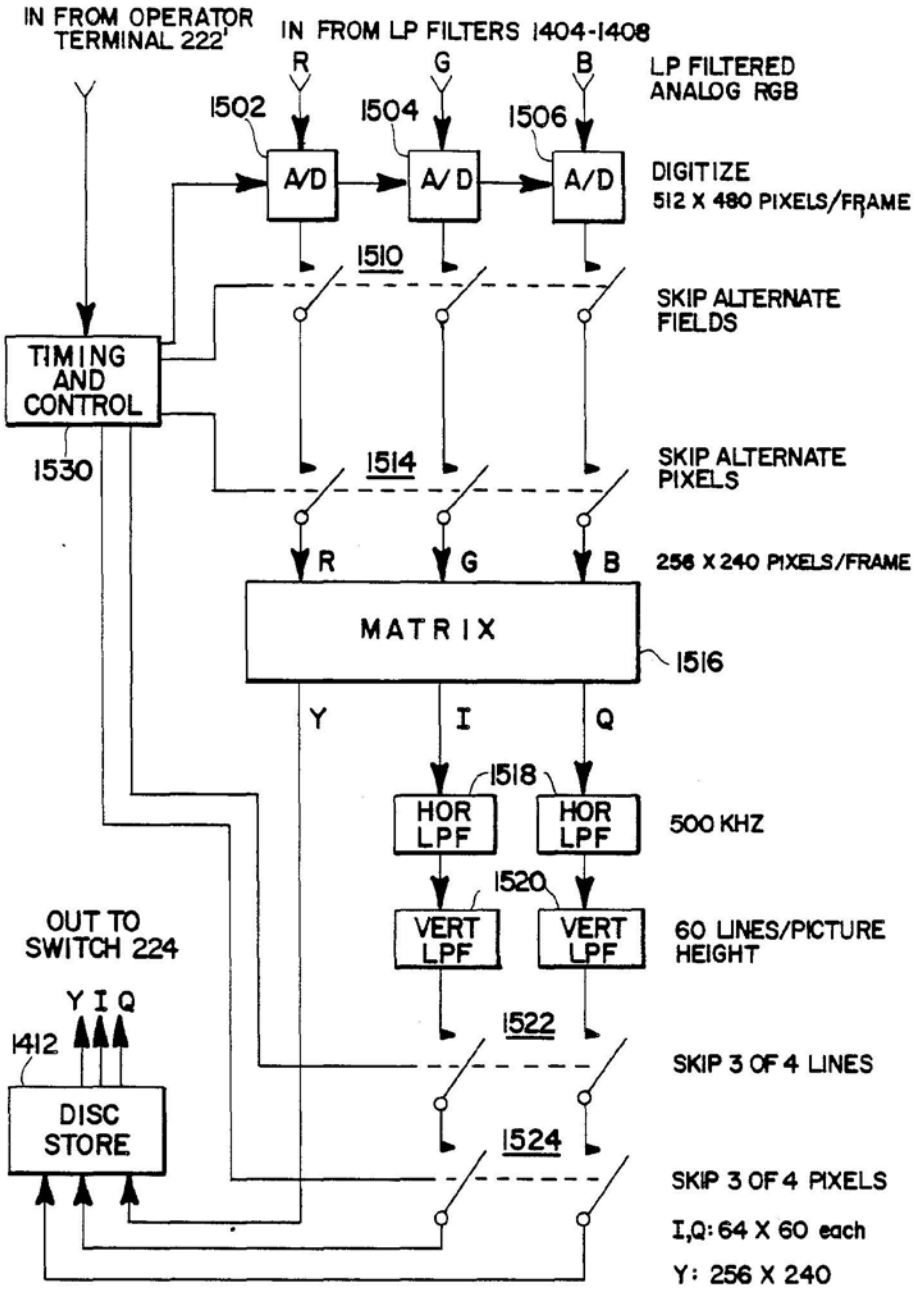

IN FROM OPERATOR TERMINAL 222'
IN FROM LP FILTERS 1404-1408
R
G
B
LP FILTERED ANALOG RGB
1502
1504
1506
A/D
A/D
A/D
DIGITIZE
512 X 480 PIXELS/FRAME
1510
SKIP ALTERNATE FIELDS
TIMING AND CONTROL
1530
1514
SKIP ALTERNATE PIXELS
R
G
B
256 X 240 PIXELS/FRAME
MATRIX
1516
Y
I
Q
1518
HOR LPF
HOR LPF
500 KHZ
1520
VERT LPF
VERT LPF
60 LINES/PICTURE HEIGHT
OUT TO SWITCH 224
Y I Q
1412
DISC STORE
1522
SKIP 3 OF 4 LINES
1524
SKIP 3 OF 4 PIXELS
I,Q: 64 X 60 each
Y: 256 X 240

FIG. 15

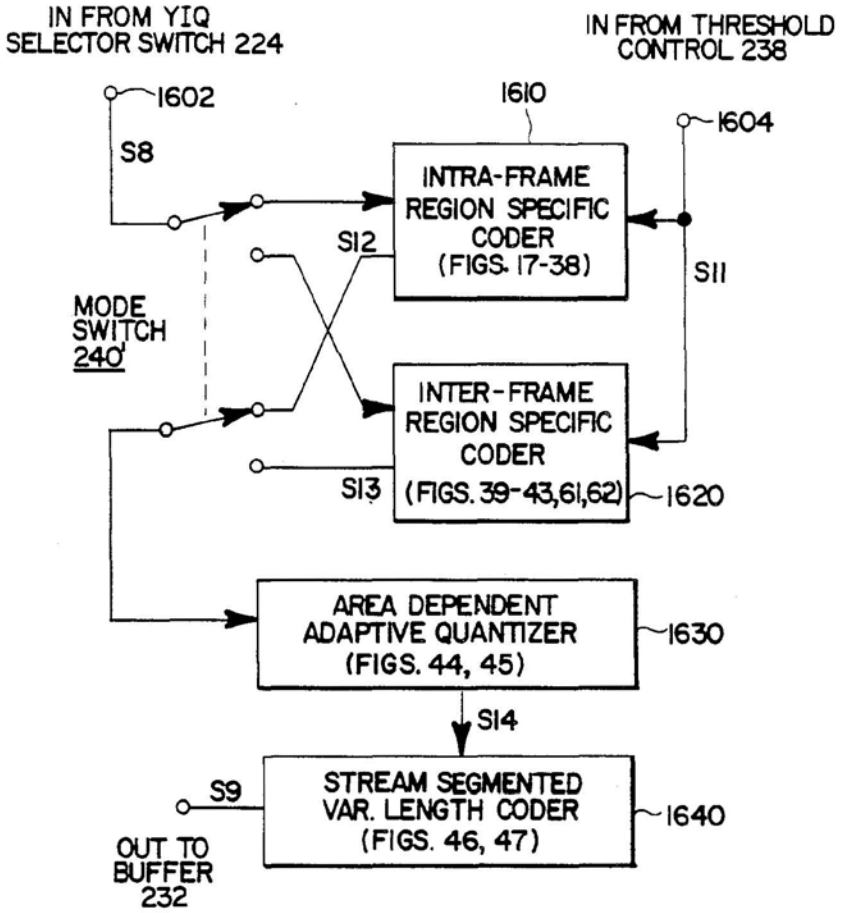

IN FROM YIQ SELECTOR SWITCH 224
IN FROM THRESHOLD CONTROL 238
1602
1610
1604
S8
INTRA-FRAME REGION SPECIFIC CODER (FIGS. 17-38)
S12
S11
MODE SWITCH 240'
INTER-FRAME REGION SPECIFIC CODER (FIGS. 39-43,61,62)
S13
1620
AREA DEPENDENT ADAPTIVE QUANTIZER (FIGS. 44, 45)
1630
S14
S9
STREAM SEGMENTED VAR. LENGTH CODER (FIGS. 46, 47)
1640
OUT TO BUFFER 232
DIGITAL VIDEO COMPRESSOR 230

FIG. 16

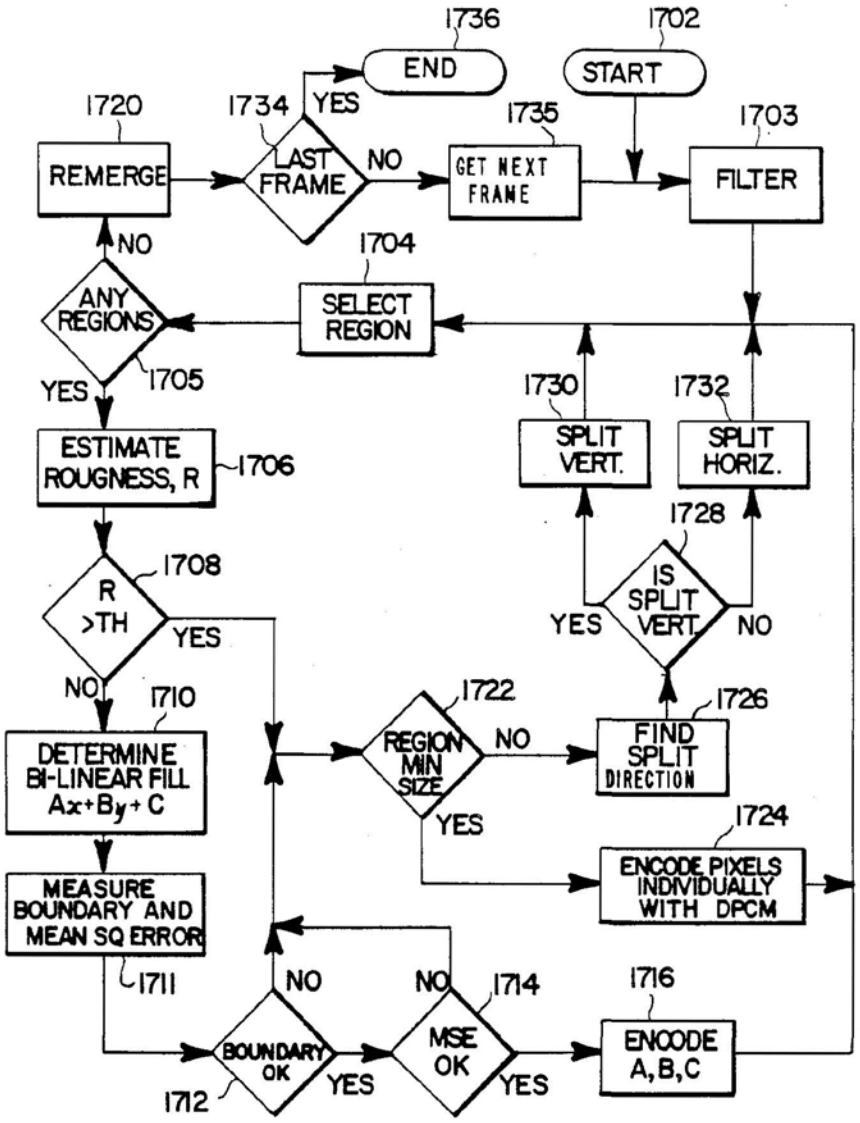

1736
END
1702
START
1720
REMERGE
1734
LAST FRAME
YES
NO
1735
GET NEXT FRAME
1703
FILTER
NO
1704
SELECT REGION
ANY REGIONS
1705
YES
1730
SPLIT VERT.
1732
SPLIT HORIZ.
ESTIMATE ROUGNESS, R
1706
1728
IS SPLIT VERT.
YES
NO
1708
R >TH
YES
NO
1710
DETERMINE BI-LINEAR FILL Ax+By+C
1722
REGION MIN SIZE
NO
YES
1726
FIND SPLIT DIRECTION
1724
ENCODE PIXELS INDIVIDUALLY WITH DPCM
MEASURE BOUNDARY AND MEAN SQ ERROR
1711
NO
NO
1714
1716
BOUNDARY OK
YES
1712
MSE OK
YES
ENCODE A,B,C

FIG. 17

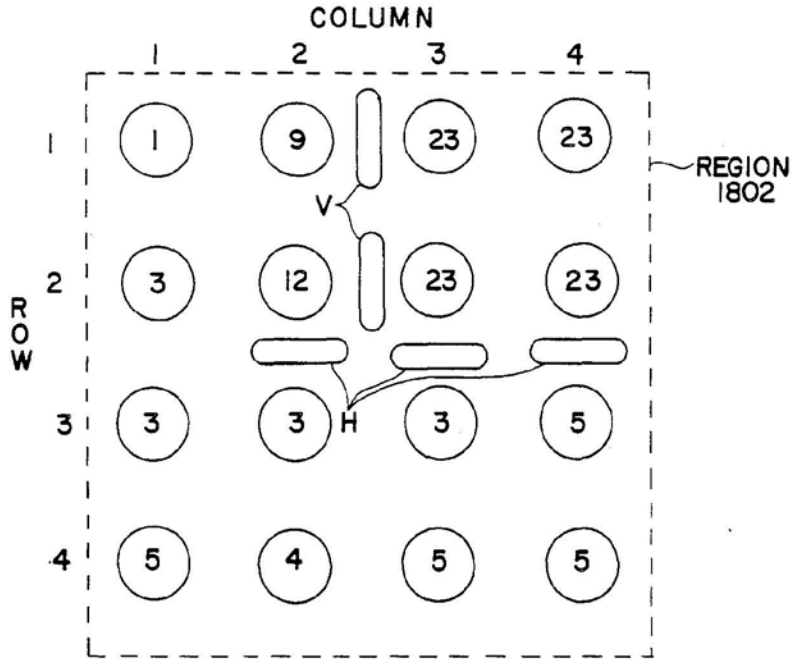

COLUMN
1
2
3
4
ROW
1
2
3
4
1
9
23
23
3
12
23
23
3
3
3
3
5
5
4
5
5
V
H
REGION
1802

FIG. 18

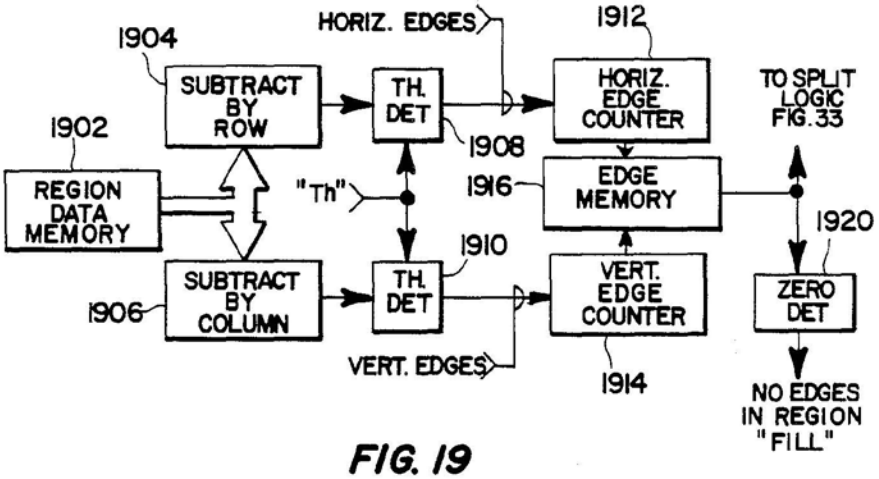

1904
SUBTRACT BY ROW
HORIZ. EDGES
TH. DET
1912
HORIZ. EDGE COUNTER
TO SPLIT LOGIC FIG. 33
1902
REGION DATA MEMORY
"Th"
1908
1916
EDGE MEMORY
1920
1910
SUBTRACT BY COLUMN
1906
TH. DET
VERT. EDGE COUNTER
ZERO DET
VERT. EDGES
1914
NO EDGES IN REGION "FILL"

FIG. 19

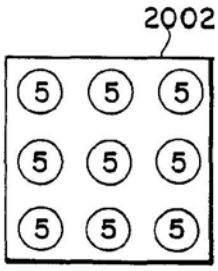


FIG. 20

NO "SLOPE" OR GRADIENT
FILL=Ax + By + C
A=0
B=0
C=5
CODE: ABS 0 0 5

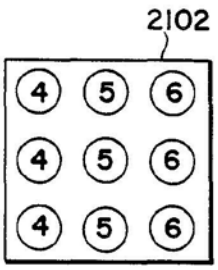


FIG. 21

UNIFORM HOR. SLOPE
FILL = Ax + By + C
A=1
B=0
C=4
CODE: ABS 1 0 4

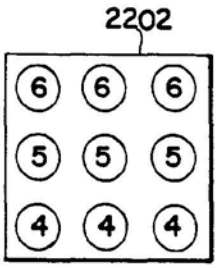


FIG. 22

UNIFORM VERT. SLOPE
FILL = Ax + By + C
A=0
B=-1
C=6
CODE: ABS 0-1 6

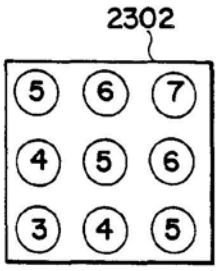


FIG. 23

VERT. AND HOR. SLOPE
FILL=Ax+By+C
A=1
B=-1
C=5
CODE: ABS 1-1 5

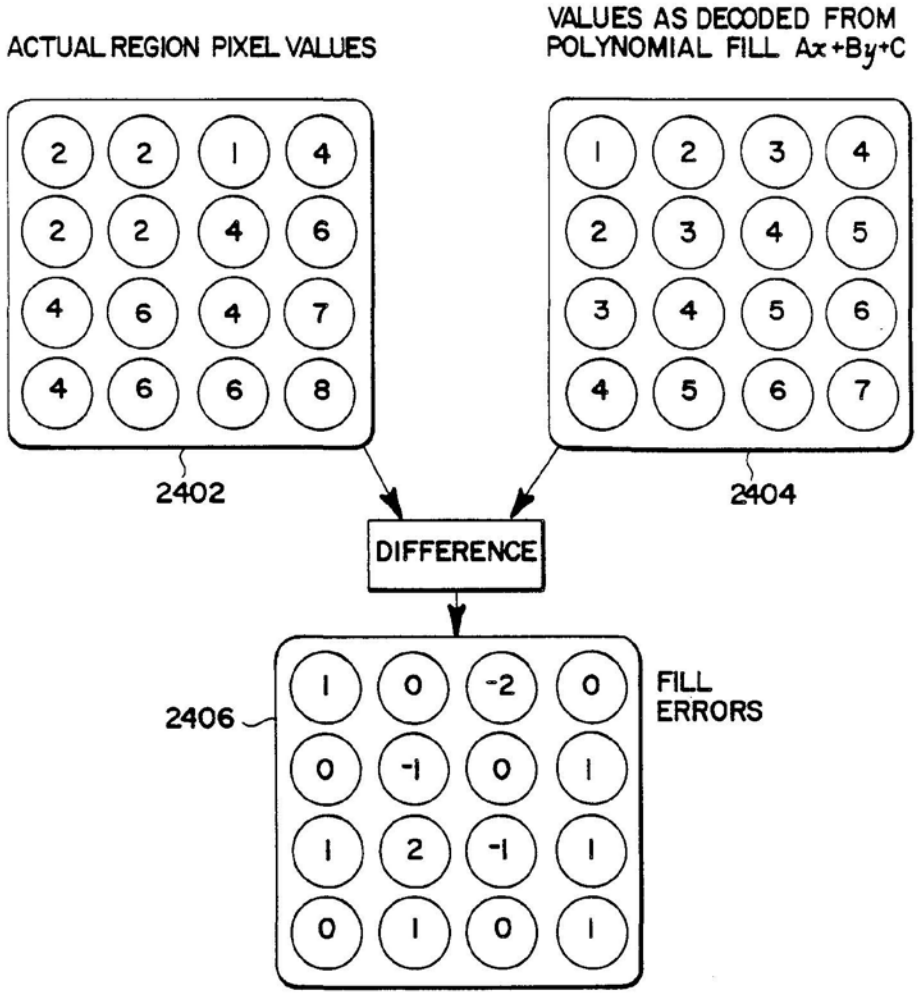

ACTUAL REGION PIXEL VALUES
VALUES AS DECODED FROM POLYNOMIAL FILL Ax+By+C
2 2 1 4
2 2 4 6
4 6 4 7
4 6 6 8
1 2 3 4
2 3 4 5
3 4 5 6
4 5 6 7
2402
2404
DIFFERENCE
2406
1 0 -2 0
0 -1 0 1
1 2 -1 1
0 1 0 1
FILL ERRORS

FIG. 24

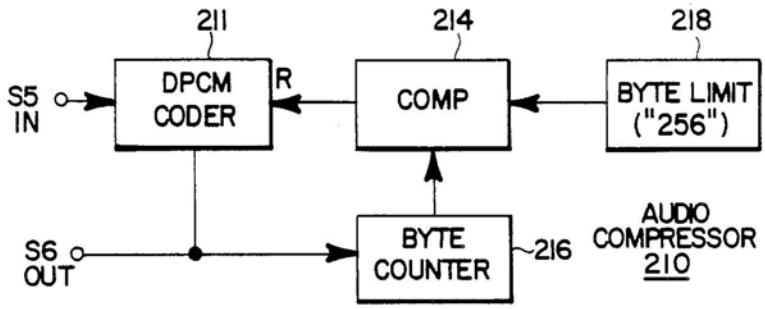
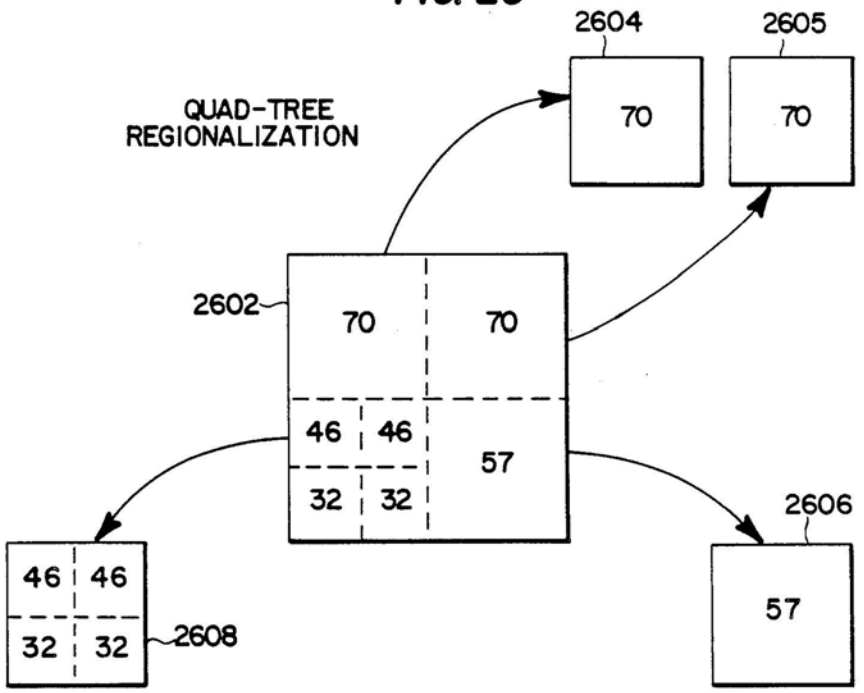

211
214
218
S5 IN
DPCM CODER
R
COMP
BYTE LIMIT ("256")
S6 OUT
BYTE COUNTER
216
AUDIO COMPRESSOR 210

FIG. 25

FIG. 26


2604
2605
QUAD-TREE REGIONALIZATION
70
70
2602
70
70
46
46
57
32
32
2606
46
46
32
32
2608
57

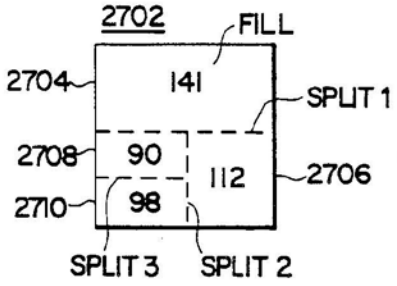


FIG. 27

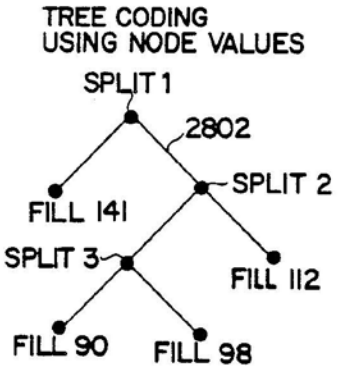


FIG. 28

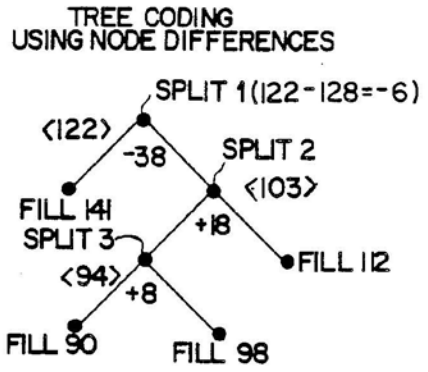


FIG. 29

FIG. 30

SPLIT/FILL
TREE CODE

HF 141 VHF 90 F 98 F 112

H: HORIZONTAL SPLIT ACTION
V: VERTICAL SPLIT ACTION
F: FILL ACTION

FIG. 31

SPLIT/FILL
TREE CODE

A. -6H-38 FV18 H8 FFF
B. 12H76 FV36 H 16 FFF
C. 12S 76 FS36 S 16 FFF

(S-Simple SPLIT
A-Alternate SPLIT)

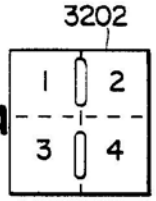


FIG. 32A

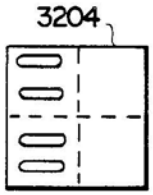


FIG. 32B

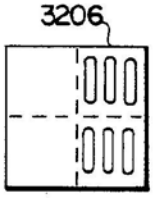


FIG. 32C

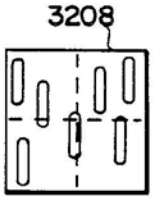


FIG. 32D

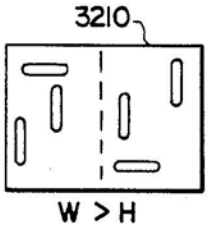


FIG. 32E

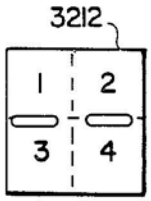


FIG. 32F

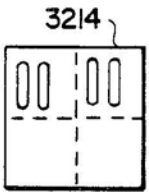


FIG. 32G

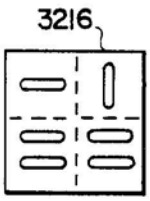


FIG. 32H

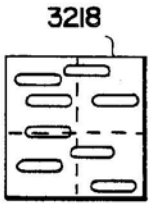


FIG. 32I

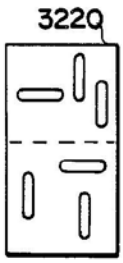


FIG. 32J

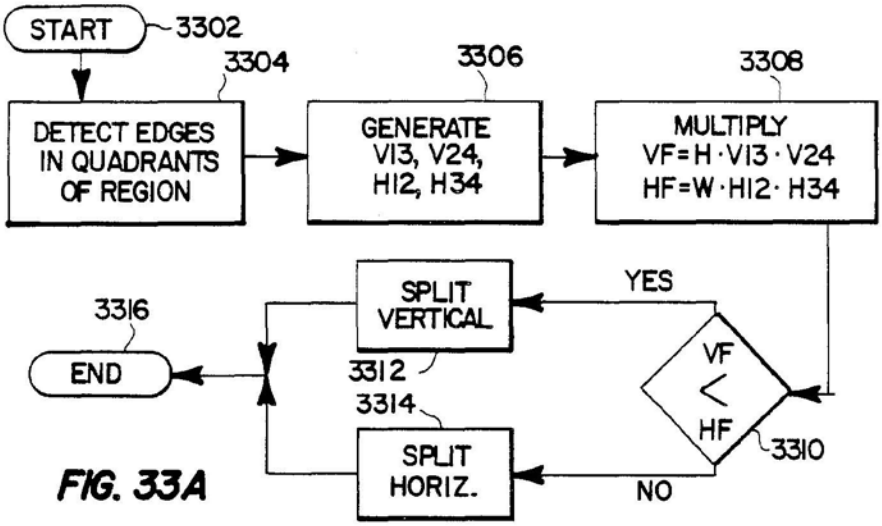


FIG. 33A

LEFT HALF (V13)	RIGHT HALF (V24)	TOP HALF (H12)	BOTTOM HALF (H34)
$(V1-V3)^2$ $+(H1-H3)^2$ $+H0^2+H1$ $+H3+1$	$(V2-V4)^2$ $+(H2-H4)^2$ $+H0^2+H2$ $+H4+1$	$(H1-H2)^2$ $+(V1-V2)^2$ $+V0^2+V1$ $+V2+1$	$(H3-H4)^2$ $+(V3-V4)^2$ $+V0^2+V3$ $+V4+1$

FIG. 33B

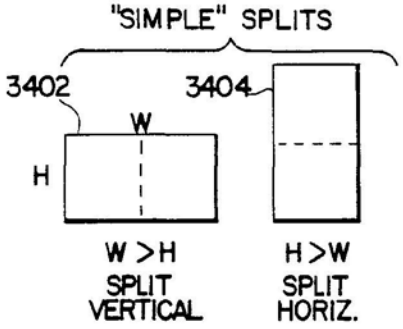
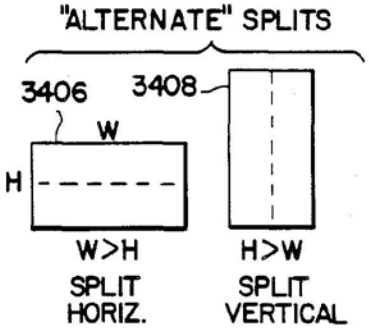


FIG. 34A

FIG. 34B



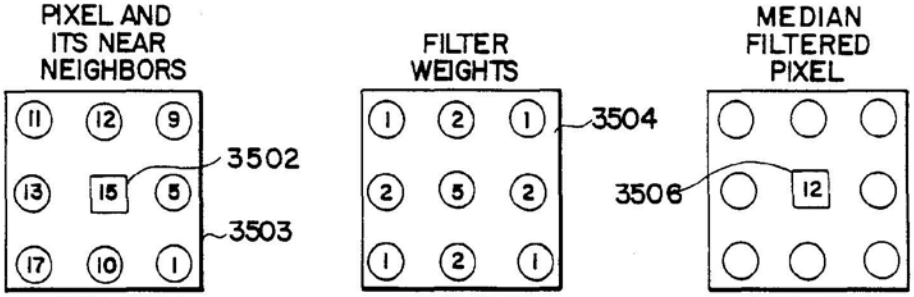


FIG. 35A FIG. 35B FIG. 35C

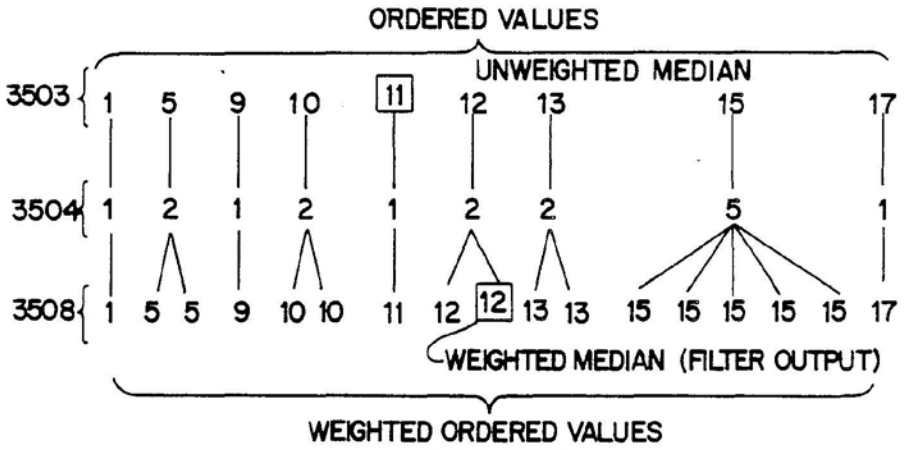
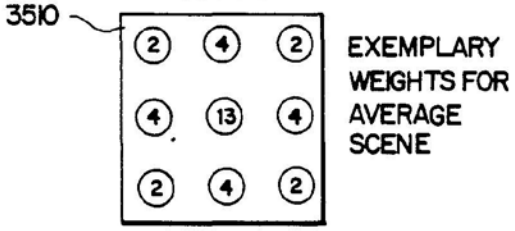


FIG. 35D

FIG. 35E



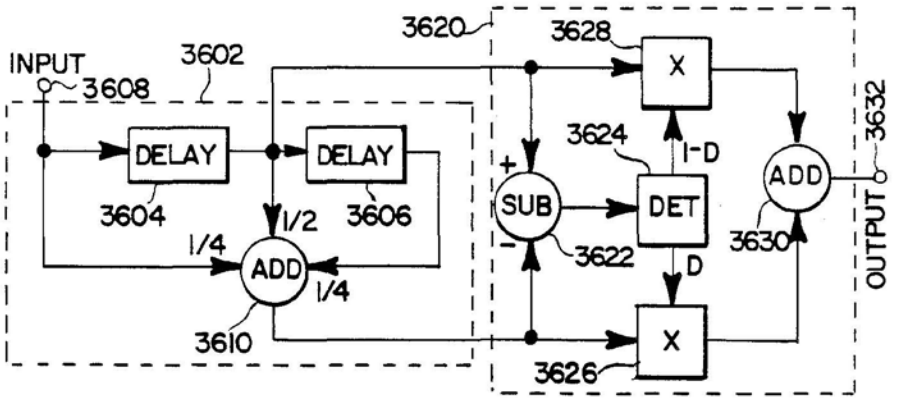
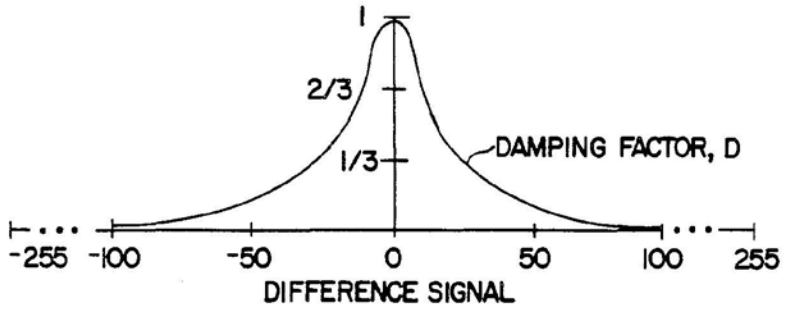


FIG. 36A

FIG. 36B



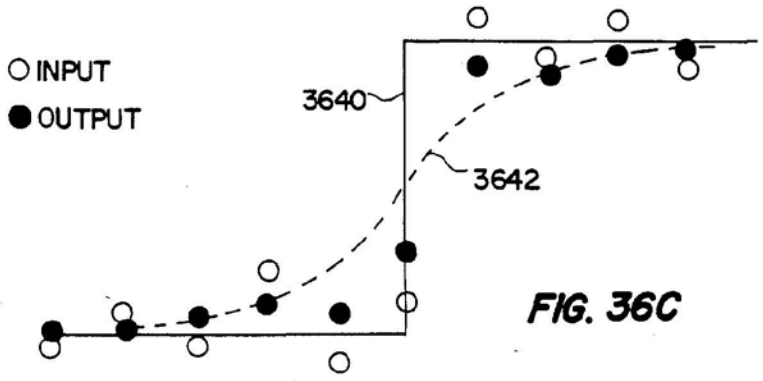


FIG. 36C

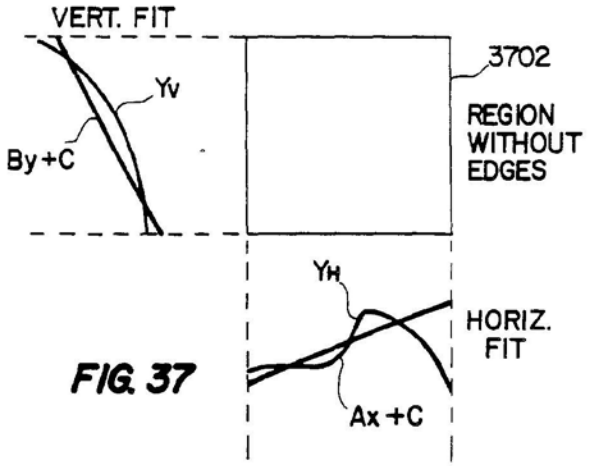

VERT. FIT
Yv
By+C
3702
REGION WITHOUT EDGES
YH
HORIZ. FIT
Ax+C

FIG. 37

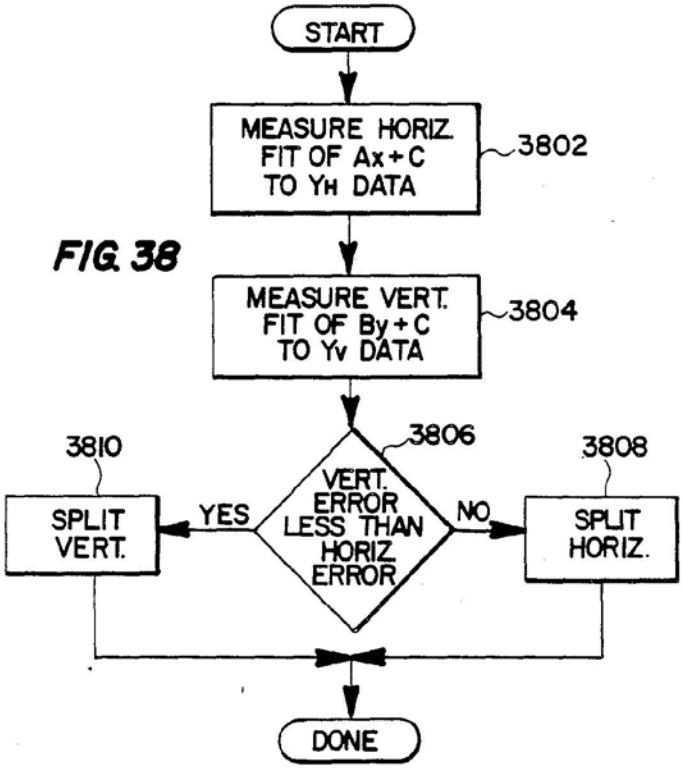

START
MEASURE HORIZ. FIT OF Ax+C TO YH DATA
3802
MEASURE VERT. FIT OF By+C TO Yv DATA
3804
3806
VERT. ERROR LESS THAN HORIZ. ERROR
YES
NO
3810
SPLIT VERT.
3808
SPLIT HORIZ.
DONE

FIG. 38

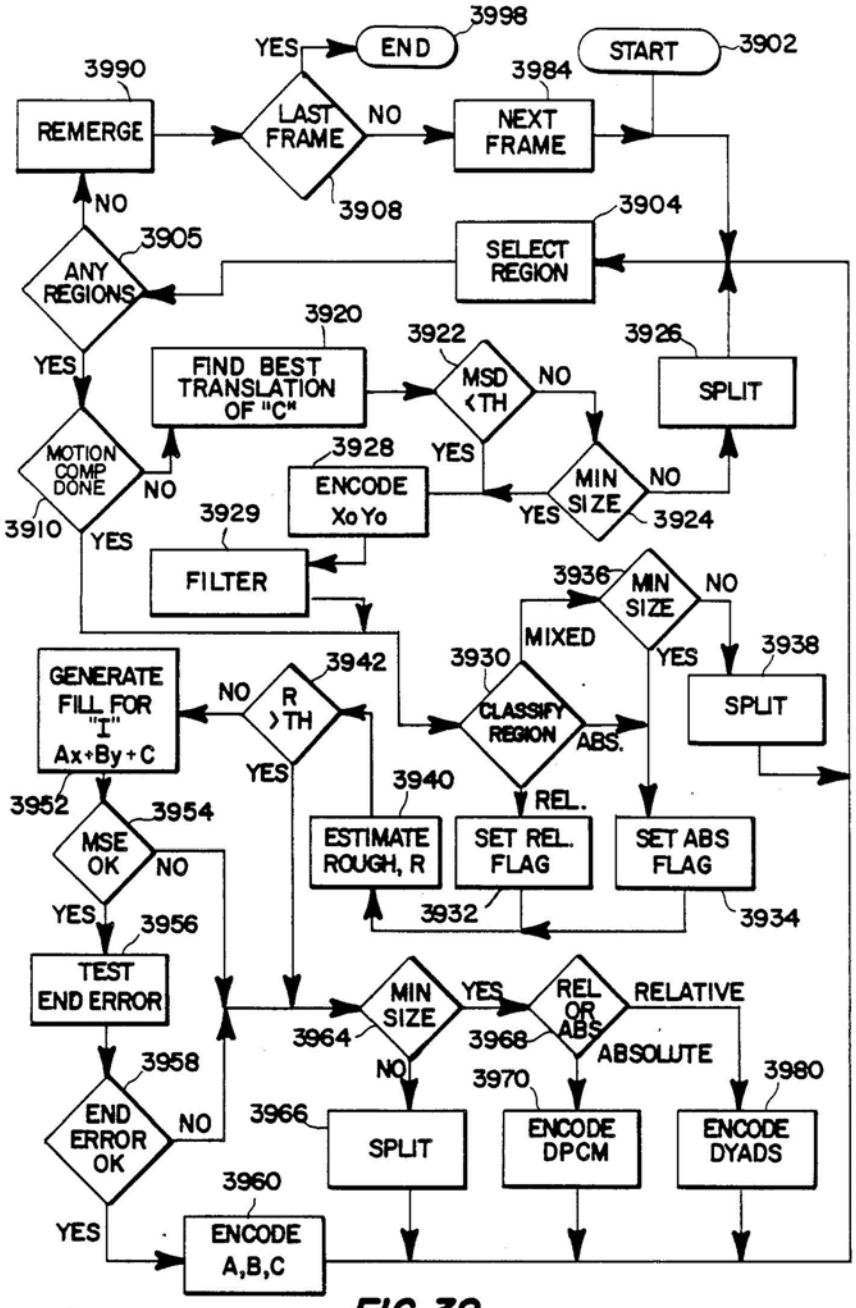

3998
END
YES
START
3902
3990
REMERGE
LAST FRAME
NO
3984
NEXT FRAME
3908
NO
3904
SELECT REGION
3905
ANY REGIONS
YES
3920
FIND BEST TRANSLATION OF "C"
3922
MSD <TH
NO
3926
SPLIT
MOTION COMP DONE
NO
3928
ENCODE Xo Yo
YES
MIN SIZE
NO
3910
YES
3929
FILTER
YES
3924
3936
MIN SIZE
NO
MIXED
YES
3938
SPLIT
GENERATE FILL FOR "I" Ax+By+C
NO
3942
R >TH
3930
CLASSIFY REGION
ABS.
YES
3940
REL.
3952
3954
MSE OK
NO
ESTIMATE ROUGH, R
SET REL. FLAG
SET ABS FLAG
YES
3956
3932
3934
TEST END ERROR
MIN SIZE
YES
REL OR ABS
RELATIVE
3964
3968
ABSOLUTE
3958
NO
3970
3980
END ERROR OK
NO
3966
SPLIT
ENCODE DPCM
ENCODE DYADS
3960
YES
ENCODE A,B,C

FIG. 39

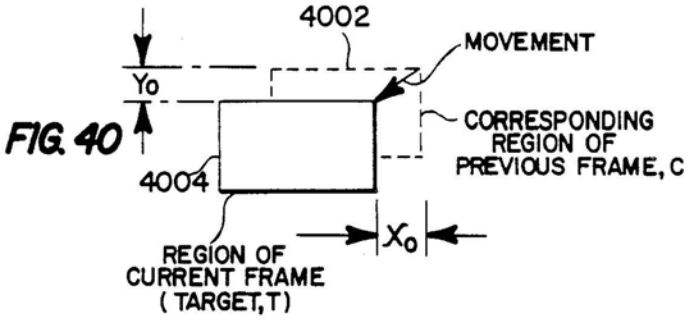


FIG. 40

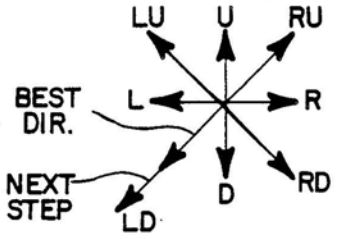


FIG. 41

FIG. 42

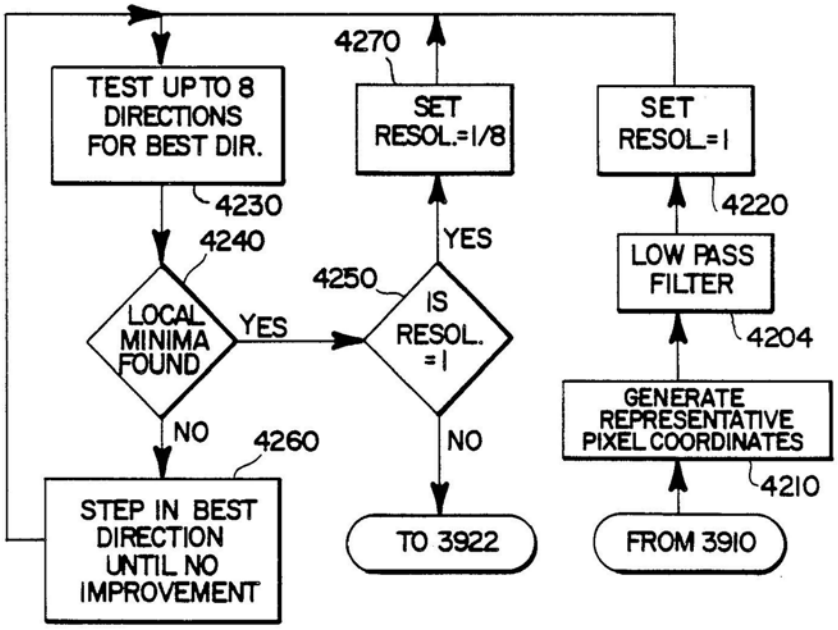
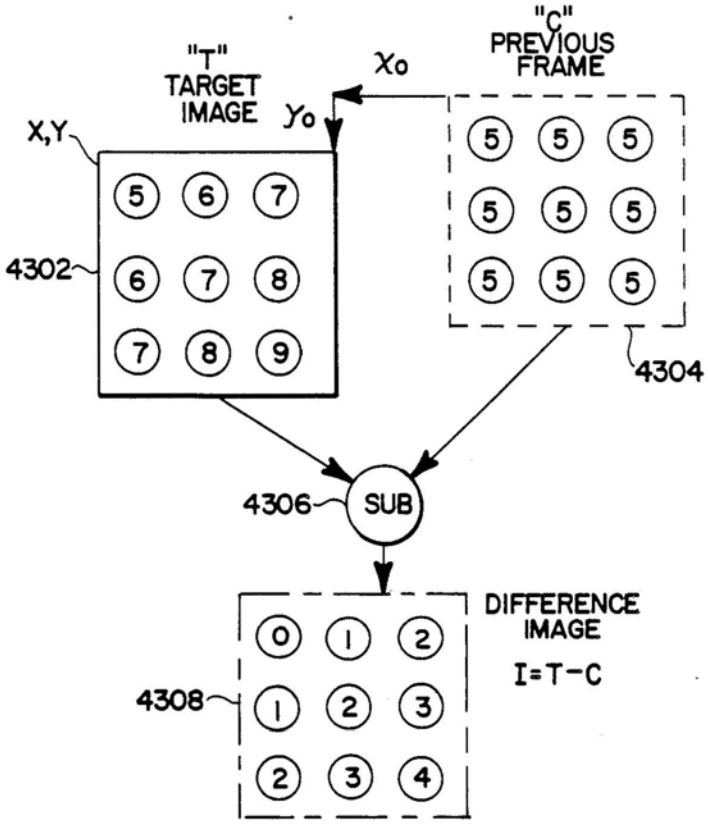


FIG. 43



RELATIVE FILL CODE: A + B + C + Xo + Yo

EXAMPLE: 1, -1, 0, Xo, Yo

FIG. 44

REGION AREA (PIXELS)	QUANTIZATION (BITS)
1	3
2-3	4
4-7	5
8-15	6
16-31	7
32-N	8

FIG. 45

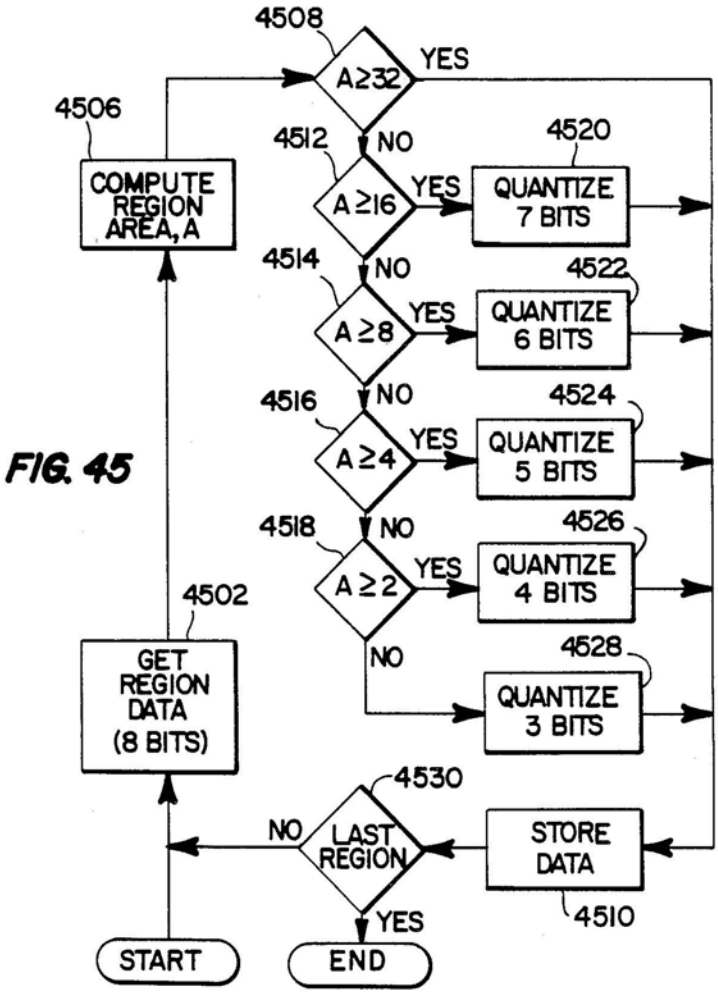


FIG. 46

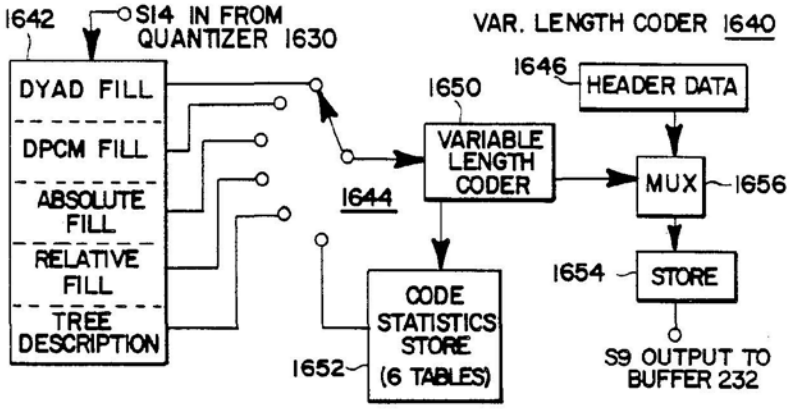


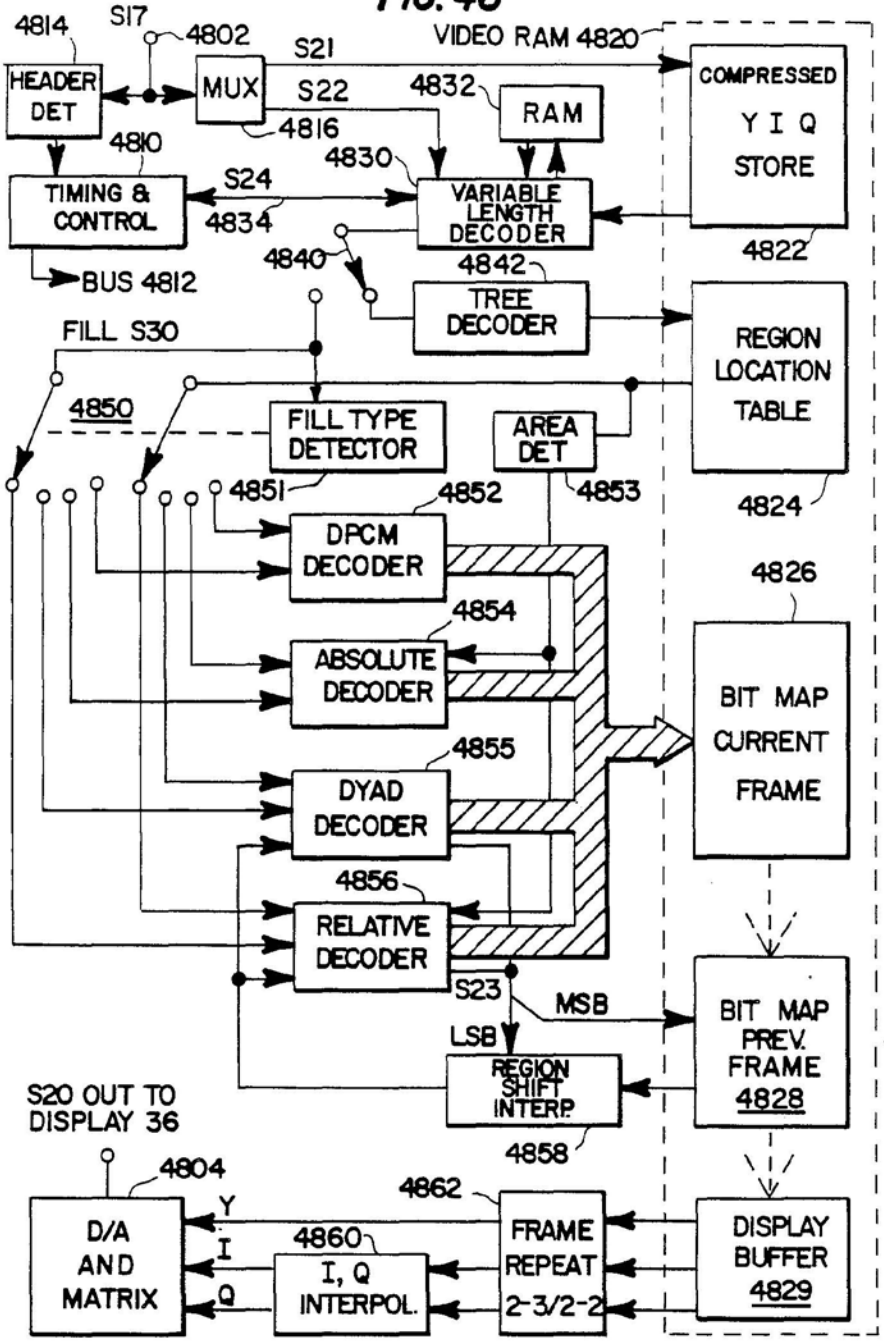
FIG. 47

FIXED LENGTH CODED
VAR. LENGTH CODED WITH IMPLICIT TABLE
VAR. LENGTH CODED WITH COL. 2 TABLES

HEADER	CODE TABLES	TREE DESCR.	RELATIVE DATA	ABSOLUTE DATA	DPCM DATA	DYAD DATA
TYPE (Y, I or Q) SIZE CHECK-SUM DPCM TABLE DYAD TABLE	1. TREE ACTIONS 2. XoYo AND C 3. COEF. A, B, C 4. ABS. SLOPES A, B 5. DPCM 6. DYADS	SPLIT FILL SHIFT VALUES XoYo ABSOL. CONST. "C"	POLY COEF. A B C	ABS. COEF. A B	DPCM VALUES ONE PER PIXEL	DYAD VALUES ONE PER TWO PIXELS

STREAM FORMAT FOR EACH OF Y, I AND Q SUBFRAMES

FIG. 48


4814
SI7
4802
S21
VIDEO RAM 4820
HEADER DET
MUX
S22
4832
RAM
COMPRESSED Y I Q STORE
4816
4810
4830
TIMING & CONTROL
S24
4834
VARIABLE LENGTH DECODER
4822
4840
4842
BUS 4812
TREE DECODER
FILL S30
REGION LOCATION TABLE
4850
FILL TYPE DETECTOR
AREA DET
4851
4852
4853
4824
DPCM DECODER
4826
4854
ABSOLUTE DECODER
BIT MAP CURRENT FRAME
4855
DYAD DECODER
4856
RELATIVE DECODER
S23
MSB
LSB
BIT MAP PREV. FRAME 4828
REGION SHIFT INTERP.
S20 OUT TO DISPLAY 36
4858
4804
4862
Y
D/A AND MATRIX
I
Q
4860
I, Q INTERPOL.
FRAME REPEAT 2-3/2-2
DISPLAY BUFFER 4829

REGION TYPE	X Y	H W	OTHER VALUES
ABSOLUTE RELATIVE DPCM OR DYAD	REGION LOCATION (UPPER LEFT CORNER)	DIMENSION (HEIGHT, WIDTH)	"C" (ABS. REGIONS) Xo Yo (REL. OR DYAD REGIONS)

(ONE ENTRY PER REGION) FIG. 49

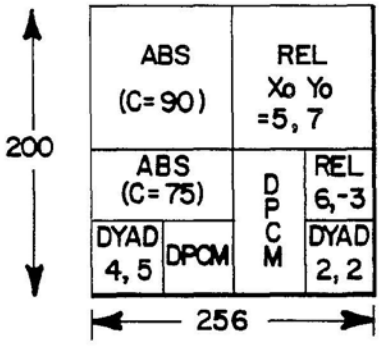
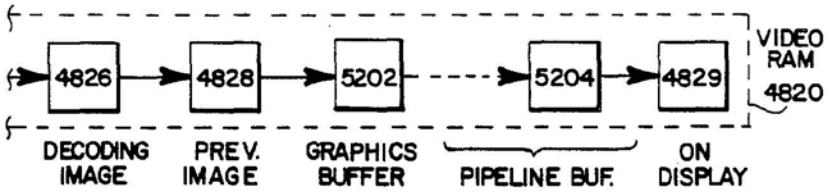


FIG. 50

FIG. 51

TYPE	X	Y	H	W	OTHER
ABS	0,	0	100	128	90
REL	128	0	100	128	5, 7
ABS	0	100	50	128	75
DYAD	0	150	50	64	4, 5
DPCM	64	150	50	64	
DPCM	128	100	100	64	
REL	192	100	50	64	6,-3
DYAD	192	150	50	64	2, 2

FIG. 52



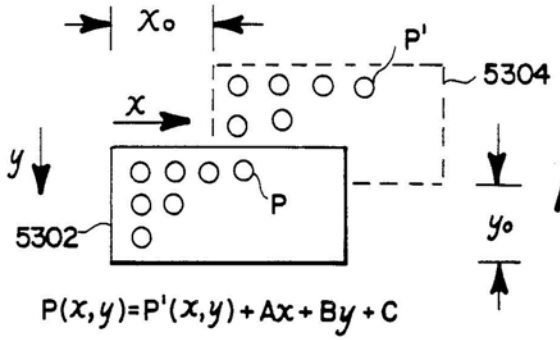

x_o
P'
5304
x
y
P
5302
y_o
$P(x,y)=P'(x,y)+Ax+By+C$

FIG. 53

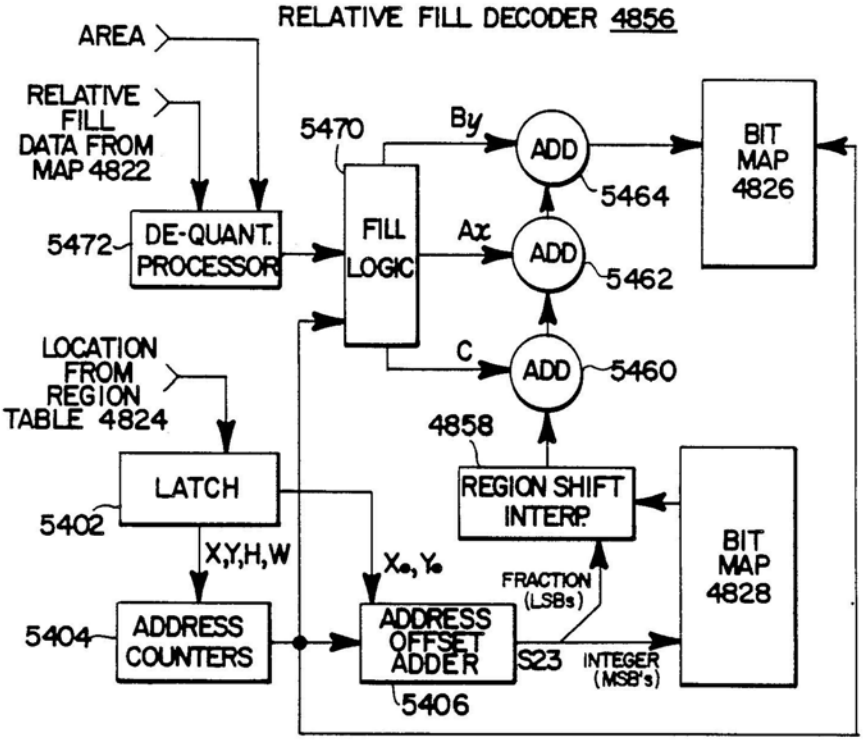

RELATIVE FILL DECODER 4856
AREA
RELATIVE FILL DATA FROM MAP 4822
5470
By
ADD
5464
BIT MAP 4826
5472
DE-QUANT. PROCESSOR
FILL LOGIC
Ax
ADD
5462
C
ADD
5460
LOCATION FROM REGION TABLE 4824
4858
LATCH
5402
REGION SHIFT INTERP.
BIT MAP 4828
X,Y,H,W
X_o, Y_o
FRACTION (LSBs)
5404
ADDRESS COUNTERS
ADDRESS OFFSET ADDER
S23
INTEGER (MSB's)
5406

FIG. 54

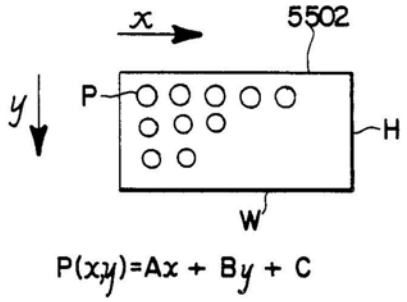

x
5502
y
P
H
W
P(x,y)=Ax + By + C

FIG. 55

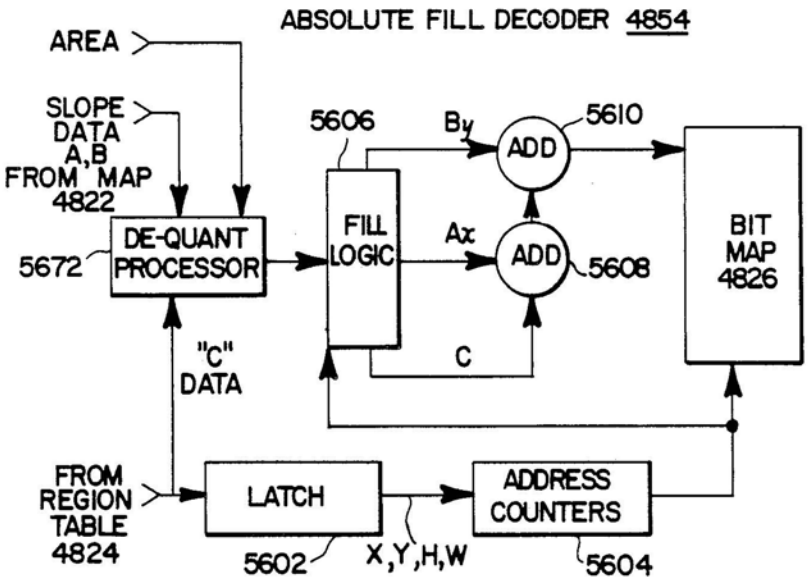

ABSOLUTE FILL DECODER 4854
AREA
SLOPE DATA A,B FROM MAP 4822
5672
DE-QUANT PROCESSOR
"C" DATA
5606
By
ADD
5610
FILL LOGIC
Ax
ADD
5608
C
BIT MAP 4826
FROM REGION TABLE 4824
LATCH
5602
X,Y,H,W
ADDRESS COUNTERS
5604

FIG. 56

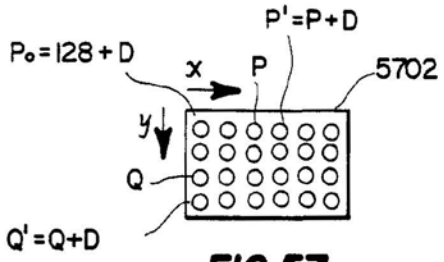
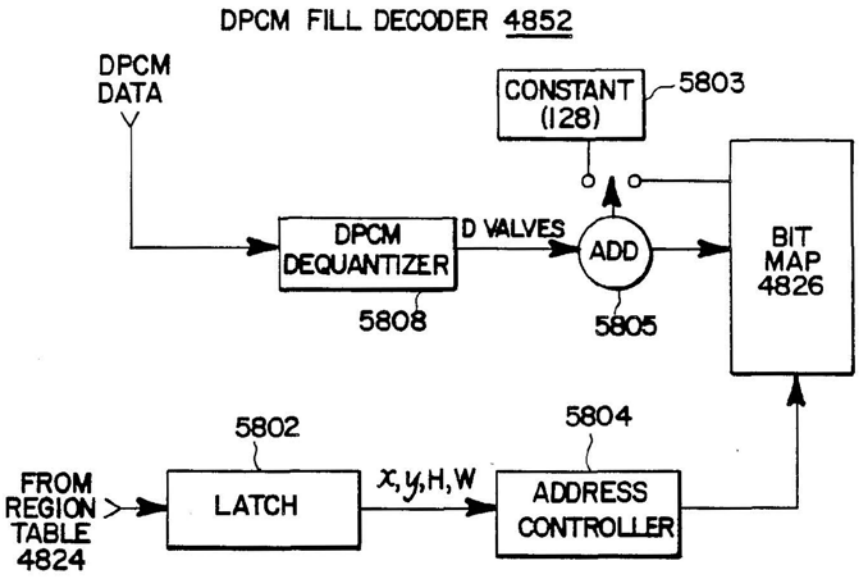

P'=P+D
Po=128+D
x
P
5702
y
Q
Q'=Q+D

FIG. 57

FIG. 58


DPCM FILL DECODER 4852
DPCM DATA
CONSTANT (128)
5803
DPCM DEQUANTIZER
D VALVES
ADD
BIT MAP 4826
5808
5805
5802
5804
FROM REGION TABLE 4824
LATCH
x,y,H,W
ADDRESS CONTROLLER

REGION AREA	QUANT. (BITS)	NUMBER OF LEVELS	SHIFT VALUE
1	3	8	5
2-3	4	16	4
4-7	5	32	3
8-15	6	64	2
16-31	7	128	1
>32	8	256	0

FIG. 59

FIG. 60

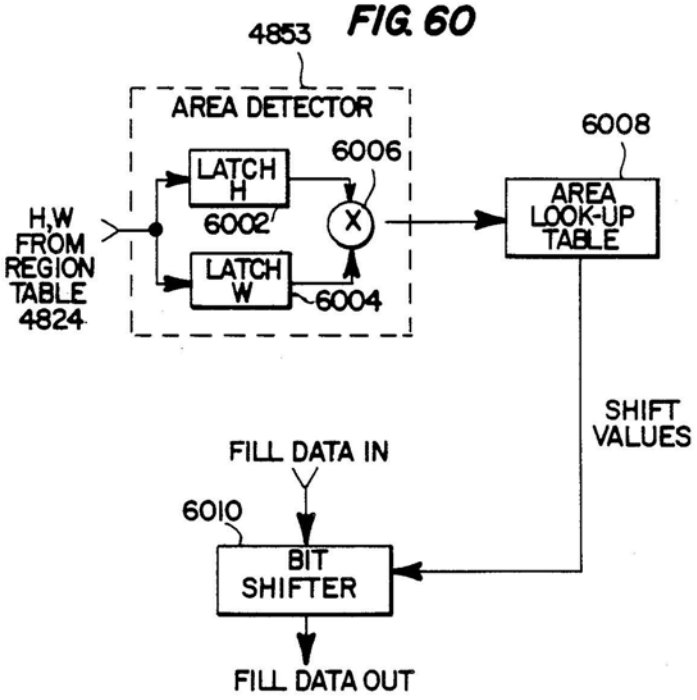


FIG. 61

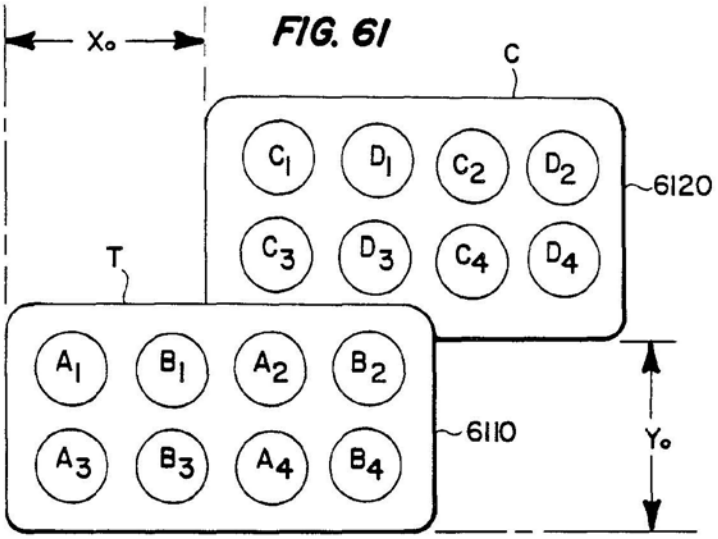
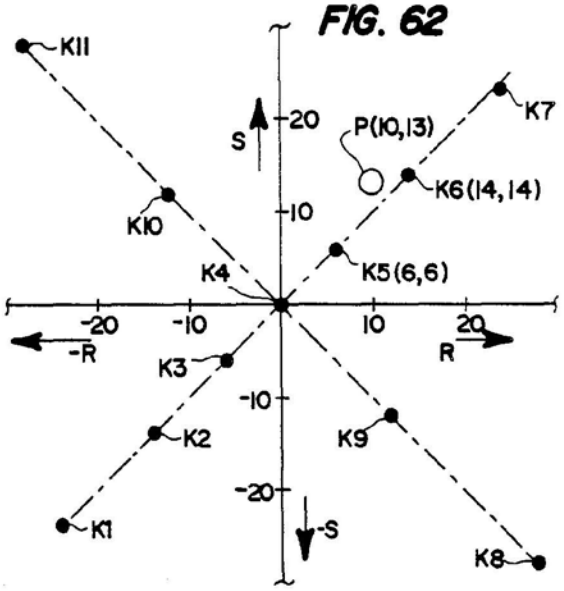


TABLE LOOK-UP (CODE $_i$) = R_i , S_i

$A_i = C_i + R_i$ $B_i = D_i + S_i$

FIG. 62



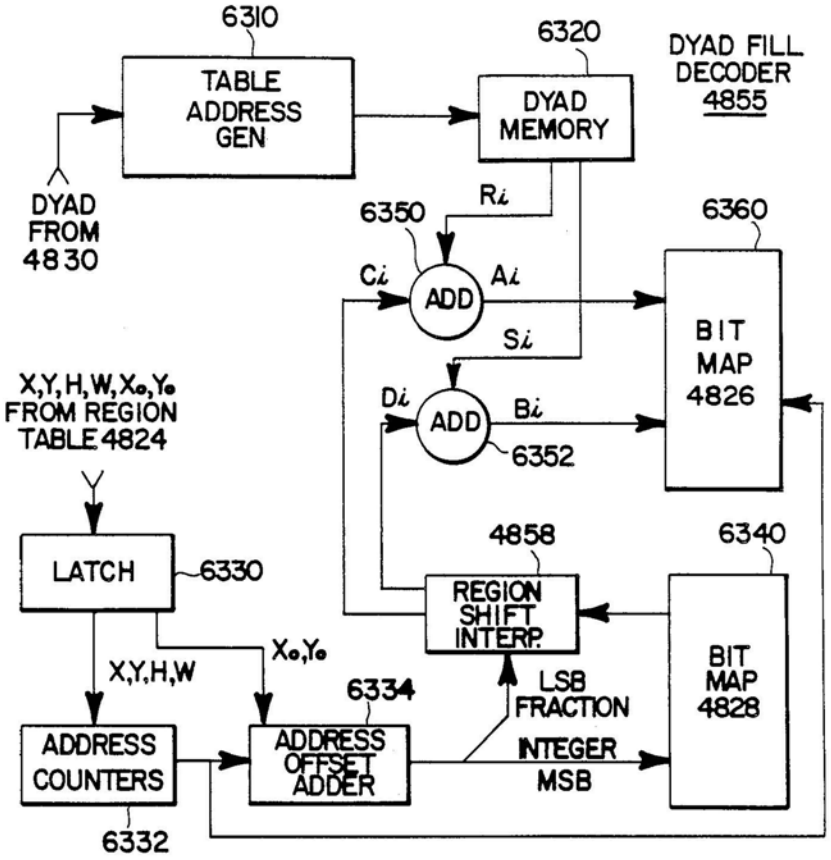

6310
TABLE ADDRESS GEN
6320
DYAD MEMORY
DYAD FILL DECODER 4855
DYAD FROM 4830
Ri
6350
Ci
ADD
Ai
6360
BIT MAP 4826
Si
Di
ADD
Bi
6352
X,Y,H,W,Xo,Yo FROM REGION TABLE 4824
LATCH
6330
4858
REGION SHIFT INTERP.
6340
BIT MAP 4828
X,Y,H,W
Xo,Yo
6334
LSB FRACTION
ADDRESS COUNTERS
ADDRESS OFFSET ADDER
INTEGER MSB
6332

FIG. 63

DIGITAL VIDEO DECOMPRESSION SYSTEM

FIELD OF THE INVENTION

This invention relates generally to systems for reducing the amount of digital data required to represent a digital video signal and particularly to systems for decoding a compressed digital video signal representative of a full motion color video signal.

BACKGROUND OF THE INVENTION

The need for compression to facilitate recording of a digital video signal on relatively narrow-band media, such as a compact disc (CD), has been recognized. In a system proposed by Takahashi et al. in U.S. Pat. No. 4,520,401, a digital video signal is encoded using differential pulse code modulation (DPCM) for recording on a digital audio disc. In the known system, luminance (Y) and chrominance (R−Y, B−Y) components of a video frame are separately compressed using DPCM. A circuit divides the components into picture element data groups of a specific number of rows or columns which are adjacent on a screen. A header signal is provided having a synchronizing signal, a picture mode identification signal and a picture information quantity identification code. The header signal is added to the beginning position of each of the divided picture element data groups to produce a digital video output signal having a signal format in which the digital luminance, the two kinds of digital color difference signal and the header signal are time sequentially multiplexed and recorded.

In an example of the Takahashi et al. system still frames of digital video are recorded and updated at a rate of about four seconds per frame. The division of the compressed data into groups of lines with each group containing complete color information provides a psuedo-motion effect in that the line groups may be sequentially updated while displaying the previous frame thereby providing a partially moving picture.

SUMMARY OF THE INVENTION

A system for compressing and formatting a full motion color digital video signal is described herein. The present invention is directed to satisfying the need for apparatus for decoding a compressed signal of the type herein described to enable display of full motion video images at normal video frame rates.

Decoding apparatus in accordance with an aspect of the invention includes means for providing a compressed digital video signal of a type in which each frame thereof comprises a plurality of coded regions, each region being encoded by a region desrriptive code and a region fill code, said region descriptive codes conveying data representative of the size and location of said regions within said frames, said region fill codes conveying pixel amplitude indications for said regions. A first decoding means, coupled to said input means, decodes said region descriptive code to provide region address data. A second decoding means, coupled to said input means, decodes said region fill codes to provide pixel values. A memory means, coupled to said first and second decoding means, stores said pixel values for said regions at addresses provided by said address data to provide a decoded bit map containing one frame of pixels.

In accordance with another aspect of the invention, said region descriptive and fill codes are conveyed by separate segments of said video input signal and further comprising control means for causing said first decoding means to decode said region descriptive codes prior to the decoding of said region fill codes by said second decoder means.

In accordance with a further aspect of the invention, said region descriptive and fill codes are conveyed by separate segments of said video input signal, said segments being individually variable length coded in accordance with their respective statistical distributions and said decoding apparatus includes a multimode variable length decoder for variable length decoding said region descriptive codes in accordance with a first statistical code and for variable length decoding said region fill codes in accordance with a second statistical code, said decoding apparatus further including circuit means coupled to said input means for deriving said statistical codes from data included in a third segment of said video input signal.

BRIEF DESCRIPTION OF THE DRAWING

The foregoing and further features of the invention are shown in the accompanying drawing in which like elements are denoted by like reference designators and in which:

FIG. **1** is a block diagram of a digital video interactive system embodying the invention providing recording and reproduction of full-motion video, multi-channel digital audio and auxiliary (e.g., interactive) data using a compact disc read-only memory (CD-ROM) as the recording media;

FIG. **2** is a block diagram of a digital video encoder used in a recording portion of the system of FIG. **1**;

FIGS. **3–9** are diagrams illustrating digital video signal formats at various stages of processing in the encoder of FIG. **2**;

FIGS. **10–12** are diagrams illustrating two methods of processing "oversized" frames in the encoder of FIG. **2**;

FIG. **13** is a block diagram of a formatter providing padding and dithering for use in the encoder of FIG. **2**;

FIG. **14** is a block diagram of a pre-compression processor used in the encoder of FIG. **2**;

FIG. **15** is a block diagram illustrating details of a portion of the processor of FIG. **14**;

FIG. **16** is a block diagram of a digital video compressor used in the encoder of FIG. **2** providing intra-frame and inter-frame region-specific coding, quantization by region area and frame-segmented variable length coding;

FIG. **17** is a flow chart illustrating operation of an intra-frame coder used in the compressor of FIG. **16** for compressing still video frames and the first frame of a motion video sequence;

FIG. **18** is a region diagram illustrating image edge analysis used in the compressor of FIG. **16**;

FIG. **19** is a block diagram of a roughness estimator providing split/fill decisions for use in the compressor of FIG. **16**;

FIGS. **20–23** are region diagrams illustrating bi-linear absolute fill coding used in the compressor of FIG. **16**;

FIG. **24** is a region diagram illustrating measurement of boundary errors;

FIG. **25** is a block diagram of an audio compressor used in the encoder of FIG. **2**;

FIG. **26** is a diagram illustrating quad-tree regionalization;

FIG. **27** is a diagram illustrating binary tree regionalization of an image in the compressor of FIG. **16**;

FIGS. 28 and 29 are examples of split/fill coding diagrams for the regionalized image of FIG. 27;

FIGS. 30 and 31 are examples of "tree" codes for the coding diagrams of FIGS. 28 and 29, respectively;

FIGS. 32A–J are region diagrams illustrating edge distribution analysis for determining a most favorable region split direction;

FIG. 33A is a flow chart for computer apparatus for determining a most favorable split direction in the compressor of FIG. 16 by analysis of the distribution of horizontal and vertical edges in a region;

FIG. 33B is a table listing of parameters for the apparatus of FIG. 33A;

FIGS. 34A and 34B are diagrams illustrating two forms of region splitting in the compressor of FIG. 16;

FIGS. 35A–35E are diagrams illustrating weighted median filtering in the compressor of FIG. 16;

FIGS. 36A–36C are diagrams illustrating non-linear low-pass filtering for use in the encoder of FIG. 16;

FIG. 37 is a diagram illustrating finding a most favorable split direction by polynomial fit comparisons;

FIG. 38 is a flow chart for computer apparatus implementing the split direction method of FIG. 37;

FIG. 39 is a flow chart illustrating operation of an inter-frame coder used in the compressor of FIG. 16 for coding the second frame and all subsequent frames of a motion video sequence;

FIG. 40 is a diagram illustrating region translation in the inter-frame coder of FIG. 39;

FIGS. 41 and 42 are vector and flow chart diagrams, respectively, illustrating selection of a best region search direction in the inter-frame coder of FIG. 39;

FIG. 43 is a diagram illustrating region translation and relative coding used in the inter-frame coder of FIG. 39;

FIG. 44 is a table illustrating region area dependent adaptive quantization used in the compressor of FIG. 16;

FIG. 45 is a flow chart illustrating operation of the apparatus in FIG. 16 providing area dependent quantization of FIG. 44;

FIG. 46 is a block diagram of a stream segmented variable length coder for use in the compressor of FIG. 16;

FIG. 47 is a diagram illustrating the format of data "streams" provided by the compressor of FIG. 16;

FIG. 48 is block diagram of a compressed digital video signal decoder used in the playback system 8 of FIG. 1;

FIGS. 49, 50 and 51 are examples of table listings of data stored in a region location memory of the decoder of FIG. 48 for absolute, relative, dyad and DPCM coded regions of FIG. 48;

FIG. 52 is a block diagram illustrating a memory organization for use in the decoder of FIG. 48;

FIG. 53 is a diagram illustrating relative region decoding of an inter-frame coded region by the decoder of FIG. 48;

FIG. 54 is a block diagram of apparatus providing the relative decoding of FIG. 53;

FIG. 55 is a diagram illustrating absolute region decoding in the decoder of FIG. 48 of an intra-frame coded region;

FIG. 56 is block diagram of apparatus providing the absolute decoding of FIG. 55;

FIG. 57 is a diagram illustrating DPCM decoding of a region in the decoder of FIG. 48;

FIG. 58 is block diagram of apparatus providing the region DPCM decoding of FIG. 57;

FIG. 59 is a table listing of area dependent adaptive quantization values for "dequantizing" pixel data in the decoder of FIG. 48;

FIG. 60 is a block diagram of apparatus for providing area dependent dequantization in the decoder of FIG. 48;

FIGS. 61 and 62 are diagrams illustrating dyad decoding in the decoder of FIG. 48; and

FIG. 63 is a block diagram of a dyad decoder for use in the decoder of FIG. 48.

DETAILED DESCRIPTION

The digital video interactive system of FIG. 1 comprises a recording system 6 and a playback system 8. The recording system includes sources 10, 12 and 14 which provide, respectively, a multi-channel sound signal S1, a color motion video signal S2 and an auxiliary data signal S3. An encoder 16 encodes and combines signals S1, S2 and S3 to form a digital recording signal S4 (hereinafter, "bit-stream") that is recorded on a compact disc read-only memory (CD-ROM) disc 20 by means of a CD-ROM recorder 18. Auxiliary data signal S3 may comprise interactive data associated with the video or audio signals or some other type of digital data which may be independent of the audio or video data.

The average data rate of the bit-stream S4 is controlled by a selection of encoding parameters to equal the standard CD-ROM record/playback bit-rate of about 1.2 mega-bits per second. The parameters are selected, as will be explained, so as to enable recording of up to one hour of full-motion digitally encoded color video, multi-channel digital audio and auxiliary data on CD-ROM disc 20.

The encoding of the digital full-motion color video portion of the recording signal to meet the relatively low channel capacity of the CD-ROM disc player requires very substantial data reduction. In a specific example to be described, this data reduction is on the order of about 150:1 for an exemplary video frame rate of 30 FPS (frames per second). To meet this critical requirement, while avoiding visible "artifacts" associated with conventional video compression techniques, encoder 16 converts the video signal S2 to a color frame sequential component form and separately subjects each frame of each component to a number of specially adapted processes as will be described. Briefly listed, these include variable sub-sampling, variable inter-frame and intra-frame compression employing what will herein be termed "region-specific" encoding, area dependent adaptive quantization, "segmented" variable length coding, reverse frame sequence reformatting, padding and frame dithering.

The selection of the individual processes, the selection of the share of data reduction provided by each process and the selection of variable compression parameters (e.g., thresholds, operating modes and, particularly, when to quit compressing) represents critical choices in meeting the objective of encoding full motion color video for storage on CD-ROM digital audio tape (DAT) or other bandwidth limited media. Such choices depend on more than merely the channel capacity of the CD-ROM media. They depend as well on variables such as the video frame rate, the desired spatial resolution, certain specific characteristics of the video image content and on parameters of the decoder that is ulti-

mately used for reconstituting the image. As will be explained, each individual video frame is converted to a component form and each component is divided to form a number of blocks (hereafter "regions") of picture elements ("pixels"). Each region is then individually "custom" encoded. This process is hereafter referred to as "region-specific" coding. The coding for each region is selected from a group of codes to enable the video decoder in playback system **8** to meet the strict requirement of completing all decoding tasks assigned to it in "real time", that is, within one video frame interval (a variable).

The foregoing and other aspects of recording system **6** are discussed in detail with reference to FIGS. **2–47** and **61, 62**. Details of playback system **8** are discussed later with reference to FIGS. **48–63**.

Encoder **16**, in FIG. **2**, includes input terminals **202, 204** and **206** for receiving audio signal **S1** from source **10**, video signal **S2** from source **12** and auxiliary data signal **S3** from source **14**, respectively. As an overview of the audio processing, signal **S1** is subjected to channel selection and analog-to-digital (A/D) conversion, compressed with provisions for preventing frame-to-frame propagation of errors and stored for later recovery as blocks of audio data to be included in each video frame of bit stream **S4** thereby providing audio/video synchronization.

In detail, audio signal **S1** is applied to a channel selector and analog-to-digital A/D converter unit **208** which includes operator controls (not shown) for selecting the number of channels to be encoded and the channel sampling rate. One channel is selected for monophonic recording, two for stereo, four for stereo/bilingual, etc. The sampling rate currently used for high quality audio recording is 31.25 KHz which supports a 15 KHz audio bandwidth. The rate may be halved for standard quality or quartered for voice grade audio applications.

The data rate of the digitized audio signal **S5** is reduced for recording by means of an adaptive differential pulse code modulation (ADPCM) encoder **210** which encodes the sample-to-sample differences of signal **S5** to form a compressed digital audio signal **S6**. Since successive audio samples are often highly correlated, fewer bits are required to encode the sample differences. The term "adaptive" means that the encoder is of a type that changes the bit significance of encoded differences as a function of the previous encoded difference so as to provide fine resolution over a wide dynamic range.

Encoder **210** may be of conventional design but it is highly desirable for purposes of overall audio/video coding that provision be made either to bypass or reset it on a periodic basis so as to periodically encode an audio sample with full resolution. Illustratively, encoder **210** (FIG. **25**) is reset once every 256 bytes. Recall that the audio signal is ultimately organized in a block form with one block of audio data included with each block of video data in bit stream **S4**. The formation of audio data "blocks" is supported via buffer store **212** which stores signal **S6**. Later the formatter **250** recovers the stored signal (**S7**) periodically on a frame-by-frame basis when the audio and video data are combined as will be explained. Typical audio block sizes currently used are 130 and 134 bytes for a video frame rate of **30** FPS and voice grade audio. The audio block size depends on the sampling rate, the number of audio channels to be recorded, and audio dithering within the formatter **250**.

One reason for periodically resetting or bypassing DPCM encoder **210** is to prevent audio errors, which may occur in the CD-ROM transmission system, from propagating from frame-to-frame. This feature also facilitates subsequent editing of sequences to enable any frame to be chosen as an edit point. This feature is implemented as, shown in FIG. **25** by means of a comparator **214** which supplies a reset signal to reset input R of audio A DPCM encoder **211** when the byte count of the compressed audio signal **S6** (produced by a byte counter **216**) exceeds the byte limit set by a byte limit source **218**.

Video Coding Overview

The principal elements providing video encoding in FIG. **2** comprise a pre-compression processor **220**, a digital video compressor **230** and an output signal formatter **250** which are described herein in detail with refeence to FIGS. **3–47**. As an overview, processor **220** provides conversion of video signal **S2** to a non-standard format that provides a variable amount of data reduction, facilitates subsequent compression and contributes to certain features of the system relating to variable frame-rate processing for controlling spatial-temporal resolution. Some images are converted at one frame rate for subsequent display at an entirely different rate.

Compressor **230** employs, broadly speaking, four types of processing for reducing the quantity of digital data to encode a frame to a specific "optimum" value. This value is related to the CD-ROM channel capacity but varies as a function of several variables including the frame rate, the desired spatial-temporal resolution, and other factors relating to error propagation and visual appearance. The processing "types" include intra-frame region-specific coding for still frames and for the first frame of a motion video sequence. Inter-frame region-specific coding is used for the second and subsequent frames of a motion video sequence. Encoded frames are subjected to further data reduction by two processes in compressor **230** which will be referred to herein as "area dependent adaptive quantization" and "segmented stream variable length coding". These processes are applied to each video frame to reach the desired "optimum" value noted above. Some sequences of frames may be repeatedly compressed with a change of compression thresholds to reach the optimum compression value.

From time to time, an "impossible" frame may be encountered which is hopelessly oversized and can not be reduced to the desired byte count by altering compression parameters without introducing noticeable visual artifacts. Such oversized frames receive special treatment in formatter **250** which combines the audio, video, auxiliary (e.g., interactive) and other data to create the recording bit-stream signal **S4**. Specifically, formatter **250** analyzes frames backwards from the last frame to the first and "borrows" space from short frames to hold the extra data of the oversized frames. Other functions provided by formatter **250** include adding "padding" data to undersized frames and dithering the number of bytes of data per frame to arrive at a specific average frame rate selected to keep the CD-ROM system operating at its maximum channel capacity and to avoid pauses during playback. Pauses are avoided because the recovery time (the "seek mode latency") of a CD-ROM player can be lengthy and unpredictable.

Details of video processing are discussed in the following five sections entitled "Video Pre-Compression-Processing", "Video Compression Processing", "Post-Compression Processing", "Playback System", and "Video Decoding".

Video Pre-Compression Processing

Pre-compression processor **220** is coupled to input terminal **204** (in FIG. 2) for converting the standard video signal S2 to a non-standard form specially adapted for the particular types of compression and formatting functions subsequently employed in encoder **16**. Specifically, each frame of the video signal S2 is converted in the "pre-compression" processor **220** to form three separate component frames comprising one luminance sub-frame and a pair of color-difference signal sub-frames. Each of the color-difference sub-frames is sub-sampled by a predetermined amount with respect to the luminance sub-frame which, itself, may or may not be sub-sampled with respect to the original video frame. The original video signal may be analog or digital and may be of component form, composite form or of another suitable form such as multiplexed analog component (MAC) form.

FIGS. **3**, **4** and **5** illustrate the pre-compression processing of one frame of video signal S2 for the case where signal S2 is assumed to be an NTSC standard composite video signal, one frame of which is shown in FIG. **3**. FIG. **4** illustrates an intermediate stage of pre-compression processing in which the composite signal has been decoded to RGB component form, stripped of synchronizing and blanking intervals and digitized to form RGB picture element (pixel) arrays representing the "active" video portion of each RGB field. The array dimensions, as illustrated, are 512 pixels horizontally by 240 pixels vertically for each RGB component.

FIG. **5** illustrates the final stage of pre-compression processing in which the digital RGB arrays of FIG. **4** have been converted to form a single luminance signal sub-frame (Y) measuring 256×240 pixels and two color difference signal subframes (I and Q) each measuring 64×60 pixels. The three sub-frames are stored in a memory (to be described) for subsequent individualized "custom" compression. Comparing FIGS. **3**, **4** and **5** it is seen that one frame of signal S2 (FIG. **3**) which requires 737,280 bytes in digital RGB form (FIG. **4**) is reduced to 69120 bytes after sub-sampling, conversion and formatting (FIG. **5**) thus providing an effective data reduction for the frame of a factor of about 11:1 for the assumed rate of 30 FPS.

An operator control unit **222** is provided in FIG. 2 for varying the sizes of the sub-frames of FIG. **5** as a function of the frame rate to facilitate varying the temporal and spatial resolution of encoded frames. This feature of the system relates to subsequent compression of the signals in the following way. The CD-ROM recording system can support a bit rate of about 1.2 mega-bits per second as previously noted. For 30 FPS (frame per second) video this channel capacity corresponds to a video byte count (8-bits/byte) of 5125.12 bytes per frame. Of this, typically about 4500 bytes per frame are available for video with the remainder being used for audio and other data. The video compressor (to be described) meets this requirement by compressing the formatted YIQ sub-frames by another factor of about 15:1 from 69120 to under 4500 bytes per frame for the assumed rate of 30 FPS. If the playback frame rate is halved then twice as much time (1/15th second) is available for decoding each frame and 9,000 bytes are available for encoding each frame. This increased decoding time and quantity of image data can be used in a variety of ways to provide improved image quality. One may for example, increase the number of pixels in the encoded frame or may more accurately enoode the same number of pixels as at the higher frame rate (30 FPS).

FIG. **14** shows a specific implementation of pre-compression processor **220** for providing the variable sub-sampling and format conversion functions previously described. Processor **220** comprises an RGB decoder **1402** which converts the composite video signal to RGB component form. The RGB components are applied via anti-aliasing (2 MHz) low-pass filters (**1404**, **1406** and **1408**) to inputs of a programmable graphics workstation **1410**. A suitable workstation is the "Adage 3000 Color Raster Display System". Operator control unit **222** of FIG. **2** comprises a terminal unit **222'** (in FIG. **14**) which supplies a "skip list" of fields, lines and pixels to workstation **1410** as well as anti-alias filter coefficients and sample rate control data. Data reduced sub-frames of Y, I and Q samples are produced by the work station and stored in a disc store **1412**.

FIG. **15** is a block diagram illustrating the specific programmed configuration of workstation **1410** for use in processing video signal S2 to create the non-standard sub-frame signal format of FIG. **5**. The anti-alias filtered analog RGB signals provided by filters **1404–1408** are applied to respective analog-to-digital converters **1502–1506** which digitize the signals at a rate selected to provide 512 pixels per active line interval as controlled by terminal **222'** coupled to the workstation timing and control unit **1530**. The digitized RGB signals (FIG. **4**) are sub-sample by two banks of switches **1510** and **1514**. Switches **1510** are timed by unit **1530** to skip alternate fields of the RGB signals. Switches **1514** skip alternate pixels, so that the resultant digitized and sub-sampled RGB signals each comprise arrays of 256×240 pixels per frame.

A matrix **1516** converts the sub-sampled RGB signals to YIQ form. The I and Q color difference signals are each sub-sampled 4:1 both vertically and horizontally with respect to the luminance signal Y. This is provided by horizontal anti-alias low-pass filters **1518** (500 KHz), vertical anti-alias low-pass filters **1520** (60 lines/picture height), switches **1522** which skip 3 of 4 lines and switches **1524** which skip 3 of 4 pixels. The formatted Y, I and Q sub-frame signals (FIG. **5**) are then stored in respective sub-frame locations in the disc store (e.g., a hard disc drive) **1412** for subsequent recovery and compression.

As previously explained, the filtering and sub-sampling parameters are variables which depend on the frame rate. For the specific examples of FIGS. **14** and **15** the frame rate is assumed to be 30 FPS. At different frame rates the operator inputs appropriate anti-alias filter coefficients, skip lists and conversion frequencies to timing and control unit **1530** via terminal **222'**. At any frame rate or resolution, however, it is important that the original signal, of whatever form (analog or digital, component, composite or MAC), be converted as shown in FIG. **5** to a form comprising a luminance component Y and a pair of color-difference components that are filtered and sub-sampled both vertically and horizontally with respect to the luminance component. Color difference components I and Q are used as examples herein. Alternatively, the color components may be of other forms, such as R-Y and B-Y or U and V.

Pre-compression processor **220** of FIG. **14** may be modified for processing a video input signal of MAC format by replacing RGB decoder **1408** with a MAC decoder providing YUV line sequential to YUV line parallel outputs, deleting the RGB/YIQ matrix in FIG. **15** and changing the sub-sampling parameters as needed to arrive at the individual (separated) sub-frames of luminance and color-difference components of FIG. **5**. It will be appreciated that other variations are possible. For example, the source may be decoded to YIQ or YUV component form prior to filtering. Sampling may be done on either RGB or YIQ.

Video Compression Processing

After pre-compression processing the Y, I and Q video sub-frames are recovered one at a time from disc store **1412** for independent compression. The sequential recovery of sub-frames is indicated symbolically in FIG. **2** by sub-frame selector switch **224**. In the position shown, switch **224** applies all Y sub-frames of a motion video sequence to compressor **230** which compresses and stores the complete sub-frames in a buffer store **232**. Switch **224** is then advanced and the compression process is repeated for all of the I sub-frames of the sequence. Finally, compression is applied to all of the Q sub-frames of the sequence thereby completing an initial stage of compression of a sequence of color frames. Alternatively, switch **224** may be advanced to select the Y, I and Q subframes of one complete frame of the sequence for compression before advancing to the next frame of a sequence.

The compressed signal **S9**, as shown in FIG. **7**, includes the three individually compressed sub-frames, each of which consists of a bitstream hader (H) followed by the compressed data for the sub-frame (Y, I, or Q). The header identifies which sub-frame the data corresponds to, the size (number of pixels horizontally and vertically) of the sub-frame, a checksum for diagnostic purposes, and various tables used by the decoder. Further details of the format of signal **S9** are discussed later with reference to FIGS, **46** and **47**. The compressed data of FIG. **7** will hereafter be referred to as a video data "stream".

The feature of compressor **230** of individually compressing the YIQ sub-frames to form the compressed digital video "stream" **S9** greatly enhances the compression efficiency. One reason is that even though the sub-frames represent the same image, they can differ from one-another dramatically because they represent different color measures of the image. Some images, for example, may contain no flesh tones. Others may contain no blue-green tones. Others may contain no color at all. A further reason for individual sub-frame compression relates to the statistical distribution of codes representing the image. Variable length coding is employed as one compression step. Variable length codes are selected in accordance with the frequency distribution or statistics of data to be coded. Since the statistics of Y, I and Q encoded sub-frames differ, individual variable length codes are employed that are optimized for each sub-frame. There are, in fact a number of separate statistical codes for each sub-frame as will be discussed.

After compression, the compressed video streams (**S10**) are recovered from buffer store **232** and applied to a byte count monitor **234** and to a decode time monitor **236** which identify, respectively, the number of data bytes and the decoding time for each individual frame of a video sequence. Since audio and auxiliary data will be added to each frame, the average byte count should be less than the total number of bytes allowed per frame in the bit stream **S4**. For encoding a video signal for playback at 30 FPS from a CD-ROM, the average number of bytes available per frame is 5125.12. This is determined by dividing the CD-ROM channel capacity by the video frame rate. Monitor **234** provides an accumulated average byte count over a sequence of video frames (alternatively monitor **234** may be arranged to count bytes on a frame-by-frame basis). This count is used for setting compression thresholds in a compression threshold control unit **238** to maintain the average byte count of signal **S10** below 4500 bytes per frame. This allows room in the frame for audio and other data that is later added. Dashed lines are used to signify this closed loop procedure which is presently performed manually in a current implementation of encoder **16**.

As previously noted, oversized video frames that can not be reduced to 4500 bytes are accounted for during reformatting by borrowing space from an earlier frame. The mechanics of this are discussed later, in the section on video post compression. Decode time monitor **236** measures the time it takes to decompress each sub-frame of the compressed digital video signal **S10**. This measurement may be accomplished by applying the signal **S10** to a decoder such as processor **30** of the playback system **8** and measuring the processor decode time. For an exemplary playback rate of 30 FPS, the decode tim of a frame should be no more than 1/30th of a second. When this monitor detects a larger decode time, thresholds in the threshold control **238** are adjusted to reduce the decode time of the "oversized" frame.

Alternatively, threshold **238** can be adjusted to merely keep the running average of the decode time below 1/30th of a second. With such a strategy, there is no need to repeat a compression, even if it exceeds the allowed decode time. In other words, the average can still be acceptable even if individual frames are not. As will be described subsequently, the playback system can cope with such temporary excesses in the decode time, without any effect on the playback rate, by using a technique of borrowing decode time from "short" frames (i.e., those frames that require less than 1/30th of a second to decode). This alternative technique of coding "oversized" frames applies where the average decode time is less than 1/30th of a second, and the playback system has adequate buffer storage. The amount of buffer storage needed by the playback system is monitored within the formatter **250** (FIG. **2**), and if it is excessive, the threshold control is adjusted to reduce the decode time further. This alternative strategy for using the decode time monitor is desirable, because it permits a more accurate encoding of those frames that need a long decode time.

The decode time monitor may alternatively comprise an estimator, based on the known decoding time characteristics of the video processor **30**. A careful examination of the decode process will reveal that it consists of a fixed number of well defined operations (say "A", "B", etc.) each of which requires a maximum length of time to complete. The encoder has available to it the precise bit stream that will be processed by the decoder. Hence the encoder can determine precisely how many times each of these operations will be performed for each sub-frame. The decode time estimate, T, is simply the sum of products:

$$T = \sum_{i=1}^{N} A_i K_i \quad (1)$$

In the summation, each term "A_i" represents the total number of times a particular decoding action is performed. The term K_i represents the maximum decoding time of the action. Examples of such actions include relative, absolute, DPCM and dyad decoding. Moreover, each decoding action may comprise several actions depending on where the pixel is in the region being decoded. To facilitate the use of such an estimator, the digital video compressor **230** stores the A_i counts associated with each sub-frame in the buffer-store **232**. They are retrieved by means of a connection (not shown) from monitors **234** and **236** to store **232**. As an example of the use of equation 1 for estimating decoding time, the products that may be summed are (1) the number of regions described by respective fill data codes times respective first constants, (2) the number of pixels included in each type of region times respective second constants and (3) the number of rows of pixels included in respective types of regions times respective third constants. A constant term may be added to the sum of products to account for decoding steps common to all regions to be decoded.

FIG. **16** is a simplified block diagram of digital video compressor **230** which includes an input terminal **1602** for receiving the YIQ selected sub-frame signal **S8** from switch **224** and another input **1604** for receiving the threshold control signal **S11** from control **238**. Mode switch **240** of FIG. **2** is indicated symbolically as switch **240'** in FIG. **16**. In the position shown (UP), mode switch **240'** applies the video sub-frame signal **S8** to an intra-frame region-specific coder **1610** which produces a region-specific coded signal **S12** that is applied via mode switch **240'** to an area dependent adaptive quantizer **1630**. The quantized region coded signal **S14** is applied to a stream-segmented variable length coder **1640** as the final compression step in producing the compressed signal **S9** for storage in buffer **232** (FIG. **2**). Reversing the position of switch **240'** applies the video input signal **S8** to an inter-frame region-specific coder **1620** and selects the inter-frame coded signal **S13** for quantization. Both encoders **1610** and **1620** are coupled to receive the threshold control signal **S11**.

In operation, mode switch **240'** is placed in the UP position for encoding still frames and the first frame of a motion video sequence using intra-frame coder **1610**. Briefly stated, coder **1610** splits the frame into a number of small groups of similar pixels referred to herein as "regions". For each region a code is produced for representing the values of all pixels of the region. This technique provides very substantial data reduction (compression) because very few bytes of code are needed to specify where a region is, how big it is and what "fill" values are to be used to represent the region pixels. Further, the specific coding method used for each region is optimally chosen based on detailed characteristics of each region. This technique (herein, "region-specific" coding) of tailoring the encoding strategy, not just to individual images, but actually to individual regions within an image, greatly increases the amount of compression possible. Details of (1) how to find the regions, (2) how to code or "fill" the region (3) how to identify "good" and "bad" fill values and (4) what to do about "bad" fills are shown and described with reference to FIGS. **17–38**.

Switch **240'** is placed in the down position for encoding the second frame and all subsequent frames of a motion video sequence using inter-frame coder **1620**. This different coding mode is used because once the first frame is encoded by coder **1610**, the second and later frames can be coded on a "relative" basis using differences of the regions from frame-to-frame. One advantage of this "relative" coding of region differences is that smaller numbers are produced and smaller numbers can be represented using fewer bits by means of variable length coding in which shorter codes are assigned to smaller numbers. Details of (1) how to find the best direction to look for corresponding regions in a previous frame, (2) how to encode the region if found and (3) what to do if a corresponding region does not exist are discussed with reference to FIGS. **39–43** and **61, 62**.

The region-specific coded signals **S12** and **S13** are subjected to what is termed herein as "area dependent" adpative quantization in quantizer **1630** which provides further data reduction. Recall that frames are coded as regions of pixels. The size of each region varies with details of the overall image. For example, in areas of high detail there will be many small regions of a few pixels each. Conversely, in areas of low detail there will be a smaller number of regions but these regions will contain tens or even hundreds of pixels each. Quantizer **1630** achieves data reduction by variably quantizing region data as a function of the region area (i.e., the number of pixels in the region) such that smaller regions are more coarsely quantized (and thus require fewer bits) than larger regions. This process, and the psychovisual effect that makes the quantization essentially invisible, will be discussed with reference to FIGS. **44** and **45**.

The quantized region-specific coded signal **S14** receives additional data reduction (compression) in variable length coder **1640**. Briefly, the data describing an image is rather complex. It includes data describing how the regions were split and filled, how regions were shifted, parameters describing the fill values in terms of bi-linear polynomial coefficients and further data in DPCM and dyad coded form. The point is that each video stream includes many types of data. These different types of data are formatted to occur in separate "segments" of each video stream. Coder **1640** determines the statistical occurrence of data for each individual segment of a video stream and assigns the shortest code to the most frequently occurring data within each segment. This is done independently for each one of the Y, I and Q sub-frames comprising a stream. In a preferred application, the different forms of region-specific codes are biased, so to speak, towards zero so that small numbers have a higher frequency of occurrence than larger numbers and thus are assigned shorter variable length codes by coder **1640**. Details of the foregoing "stream segmented" variable length coding are described with reference to FIGS. **46** and **47**.

Compressor **230** of FIG. **16** has been implemented by programming a digital computer as described with reference to FIGS. **17–47**. For the computer, a model VAX 11/785 manufactured by Digital Equipment Corporation was selected. Compression speeds of a few minutes per frame have been achieved for typical motion video sequences. The principal goal of compressor **230** is not speed but rather is high quality for the images

that are ultimately displayed. This goal is achieved in large part through the use of what is herein termed "region-specific" coding as will now be described.

Region specific coding comprises two actions, namely, (1) dividing the image into seeral regions ("regionalization"), and (2) selecting "optimal" fill parameters for each region. These two actions are performed concurrently, as will be described with reference to FIG. **17**.

FIGS. **27–31** provide an overview of the regionalization process called binary tree decomposition. In this simplified example, the region **2702** consists of four subregions (**2704, 2706, 2708, 2710**) in which the pixels are assumed to have uniform gray levels (e.g., 141, 112, 90 and 98 out of a possible range of 256 gray levels). The pixel value distribution of this sub-frame is atypical, and is only intended to illustrate how binary tree regionalization is applied, and how the resulting decomposition can be efficiently encoded. In the more general case, the "fill" (i.e., the code representing the region pixel values) is described by the linear expression $Ax+By+C$, where the coefficient "A" represents the slope or brightness gradient in the horizontal (X) direction, "B" represents the gradient in the vertical (Y) direction and "C" represents a constant or uniform level of brightness over the region. In the example of FIG. **27**, the terms A and B of the fill polynomial $Ax+By+C$ are both zero.

Binary tree decomposition is performed by splitting a region in half, and then possibly splitting each of the resulting sub-regions in half, until the resulting subregions can be efficiently encoded. Later, in the discussion of FIG. **17**, a number of strategies are described for deciding when a sub-region should be split, and in which direction it should be split, horizontal or vertical. For FIG. **27**, these decisions are easy. The first split, labeled split **1** in FIG. **27** splits the region horizontally into two equal halves. The top half **2704** can be efficiently encoded by the single value **141**, while the bottom half needs further decomposition. A further vertical split, split **2** divides the remaining area in half. The right half (**2706**) can be efficiently encoded by the value **112** and hence is not split any further. The left half, however, requires a further horizontal split, into two subregions **2708** and **2710** which can be efficiently encoded by the values **90** and **98**.

Other regionalization strategies are possible. For example in quad-tree decomposition, instead of picking a single split direction, both split directions are used together. This leads to a regionalization as shown in FIG. **26** where region **2602** is split to form four more regions **2604–2608** one of which (**2608**) is further split to form four reigons. Binary tree regionalization is the preferred mode because it has been found to normally result in fewer regions and hence fewer bits and less decode time.

FIGS. **28** and **30** illustrate the encoding of the absolute fill values and region locations of the example of FIG. **27**. The term "absolute" as used herein signifies fill values obtained solely from the region data of the region being coded. The term "relative" as used herein signifies region fill values based upon frame-to-frame region differences. The inverted tree-like structure of the coding diagram **2802** in FIG. **28** represents successive divisions of region **2702** and is called a "binary tree" because each branch is split to form two branches. The top node of the tree represents the whole image. Each time a region is split, two new node values are formed. Terminal nodes of the tree are encoded with the region fill values.

The code (FIG. **30**) to describe the complete tree consists, therefore, of two types of data: "values," which are the fill values, and "actions", which are the split or fill commands. The "actions" and "values" are encoded using the same code "space". That is, they each comprise variable-length-encoded non-negative numbers. It is always possible, however, to distinguish between an action and a value based on context, that is, the position of the action or value in the code sequence. For instance, in the example of FIG. **28**, when a "fill" action is encountered, the next number must be a value. The next item after this value must be another action, etc.

In more detail, the tree description data is ordered using the following rule. For each node that is split, all the data pertaining to the "top" node (if a horizontal split) or the "left" node (if a vertical split) is listed, followed by all the data for the other node. This is an inherently recursive procedure that begins with the root node of the tree and operates successively on nodes of the tree until all terminal nodes of the tree are reached. For the example tree in FIG. **28** this yields the tree code shown in FIG. **30**. This short code, together with the dimensions of the original image, gives all the information one needs to specify the size and location of every region and the value of every pixel in the image **2702**. The "H" and "V" symbols signify horizontal and vertical splits. The "F" symbol signifies a fill action.

FIG. **29** illustrates an alternative and preferred format for encoding the binary tree data for the regions of FIG. **27**. It differs from the method of FIG. **28** in that the fill data is encoded as node differences rather than as the actual values of the end nodes. This requires calculation in the decoder to recover the actual fill values but has an advantage in that the encoded values are numerically smaller. Compare, for example, the values 141, 90, 98, 112 of FIG. **30** with the values -7, -37, -18, and 8 of FIG. **31**. Since the values are encoded using a variable-length code, this produces greater coding efficiency, since this weights the statistics of the values more heavily towards small numbers.

The coding procedure which results in the binary tree illustrated in FIG. **29** is performed as follows. First, the encoding process which develops the binary tree of FIG. **28** is performed. Next, pairs of fill values at terminal nodes from the same branch point are differenced and averaged. The difference value is assigned to the branch point and is the value which will subsequently be encoded in the tree description. The average value is also assigned to the branch point, but only for the purpose of determining other nodal or branch values working backwards up the tree. That is, the average values are averaged and differenced with absolute or average values from a corresponding node on a parallel branch. The difference value is assigned to the branch point as the value to be encoded, and the new average value is used to determine the next difference and average value working hierarchically up the tree. Differences are determined by subtracting the left nodal or branch value from the right nodal or branch value.

In the example illustrated the terminal nodal value **90** is subtracted from the terminal value **98** to produce the difference value $+8$ which is assigned to the branch point designated "split 3". The average of the nodal values $(90+98)/2=94$ is also applied to the branch point and shown in angle brackets. The average value