US006683978B1

(12) **United States Patent**
Iourcha et al.

(10) **Patent No.: US 6,683,978 B1**
(45) **Date of Patent: Jan. 27, 2004**

(54) **FIXED-RATE BLOCK-BASED IMAGE COMPRESSION WITH INFERRED PIXEL VALUES**

(75) Inventors: **Konstantine I. Iourcha**, San Jose, CA (US); **Krishna S. Nayak**, Stanford, CA (US); **Zhou Hong**, San Jose, CA (US)

(73) Assignee: **S3 Graphics Co., Ltd.**, Grand Cayman (KY)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/442,114**

(22) Filed: **Nov. 17, 1999**

Related U.S. Application Data

(63) Continuation of application No. 09/351,930, filed on Jul. 12, 1999, which is a continuation of application No. 08/942,860, filed on Oct. 2, 1997, now Pat. No. 5,956,431.

(51) **Int. Cl.**[7] **G06K 9/00**

(52) **U.S. Cl.** **382/166**; 382/232; 725/146

(58) **Field of Search** 382/166, 239, 382/253, 232; 345/550; 725/146; 358/462, 1.15; 348/63

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,821,208 A	*	4/1989	Ryan et al.	345/550
4,887,151 A		12/1989	Wataya	358/539
5,734,744 A		3/1998	Wittenstein et al.	382/166
5,742,892 A	*	4/1998	Chaddha	725/146
5,748,904 A		5/1998	Huang et al.	345/544
5,787,192 A		7/1998	Takaichi et al.	382/166
5,822,465 A		10/1998	Normile et al.	382/253
5,956,425 A		9/1999	Yoshida	382/234
5,956,431 A		9/1999	Iourcha et al.	382/253
6,075,619 A		6/2000	Iizuka	382/166

FOREIGN PATENT DOCUMENTS

JP	405216993	8/1993	G06F/15/70

OTHER PUBLICATIONS

A. Schilling et al., "Texram: A Smart Memory for Texturing", IEEE Computer Graphics & Applications, May 1996, 16(3), pp. 9–19.

G. Knittel et al., "Hardware and Software for Superior Texture Performance"In 10, Eurographics Hardware Workshop, Maastricht, NL, Aug. 28–29 1995, pp. 1–8.

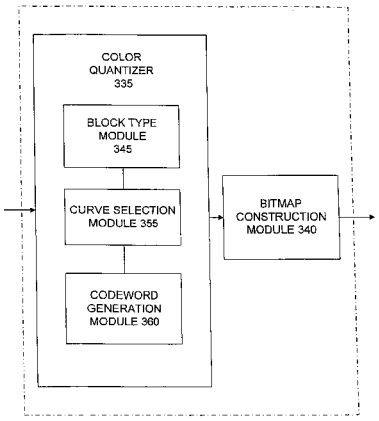
(List continued on next page.)

Primary Examiner—Anh Hong Do
(74) *Attorney, Agent, or Firm*—Carr & Ferrell LLP

(57) **ABSTRACT**

An image processing system includes an image encoder system and a image decoder system that are coupled together. The image encoder system includes a block decomposer and a block encoder that are coupled together. The block encoder includes a color quantizer and a bitmap construction module. The block decomposer breaks an original image into blocks. Each block is then processed by the block encoder. Specifically, the color quantizer selects some number of base points, or codewords, that serve as reference pixel values, such as colors, from which quantized pixel values are derived. The bitmap construction module then maps each pixel colors to one of the derived quantized colors. The codewords and bitmap are output as encoded image blocks. The decoder system includes a block decoder. The block decoder includes a block type detector, one or more decoder units, and an output selector. Using the codewords of the encoded data blocks, the comparator and the decoder units determine the quantized colors for the encoded image block and map each pixel to one of the quantized colors. The output selector outputs the appropriate color, which is ordered in an image composer with the other decoded blocks to output an image representative of the original image. A method for encoding an original image and for decoding the encoded image to generate a representation of the original image is also disclosed.

29 Claims, 16 Drawing Sheets



318

OTHER PUBLICATIONS

G. Campbell et al., "Two Bit/Pixel Full Color Encoding", Computer Graphics (Proc. SIGGRAPH '86), Aug. 18–22, 1986, vol. 20, No. 4, Dallas, TX, pp. 215–219.

Yang, Ching Yung et al., "Hybrid Adaptive Block Truncation Coding for Image Compression," Optical Engineering, Soc. of Photo–Optical Instrumentation Engineers, Apr. 1, 1997, p. 1021–1027, vol. 36, No. 4, Bellingham, WA, USA.

Kugler, A. "High–Performance Texture Decompression Hardware," Visual Computer, Springer–Verlag, 1997, p. 51–63, vol. 13, No. 2, Berlin, Germany.

Nasiopoulos, Panos, et al., "Adaptive Compression Coding," IEEE Transactions on Communications, IEEE Inc., Aug. 1, 1991, p. 1245–1254, vol. 39, No. 8, New York, USA.

Knittel, G. et al., "Hardware for Superior Texture Performance," Eurographics Workshop on Graphics Hardware, Jul. 28, 1995, p 33–40.

Campbell, G. et al., "Two Bit/Pixel Full Color Encoding," Computer Graphics, Aug. 1986, p. 215–219, vol. 20, No. 4, New York, New York, USA.

Delp E. J. et al., "Image Compression Using Block Truncation Coding," IEEE Transactions on Communcations, Sep. 1979, p. 1335–1342, vol. COM–27, No. 9, IEEE Inc., New York, USA.

Yang, Ching Yung et al., "Use of Radius Weighted Mean to Cluster Two–Class Data," Electronics Letters, May 12, 1994, p. 757–759, vol. 30, No. 10, IEE Stevenage, Great Britain.

Russ, J. C., "Optimal Grey Scale Images From Multiplane Color Images," Journal of Computer–Assisted Microscopy, Dec. 1995, p. 221–223, vol. 7, No. 4, Plenum, USA.

Knittel, G., et al., "Hardware and Software for Superior Texture Performance," Eurographics Hardware Wokshop '95, Aug. 28–29, 1995, pp. 1–8, Masstricht, NL.

Schilling, A., et al., "Texram: A Smart Memory for Texturing," IEEE Computer Graphics & Applications, May 1996, pp. 9–19, vol. 16, No. 3.

Feng et al., "A Dynamic Address Vector Quantization Algorithm Based on Inter–Block and Inter–Color Correction for Color Image Coding," IEEE International conference on Acoustics, Speech, and Signal Processing, May 1989, pp. 1755–1758, vol. 3.

* cited by examiner

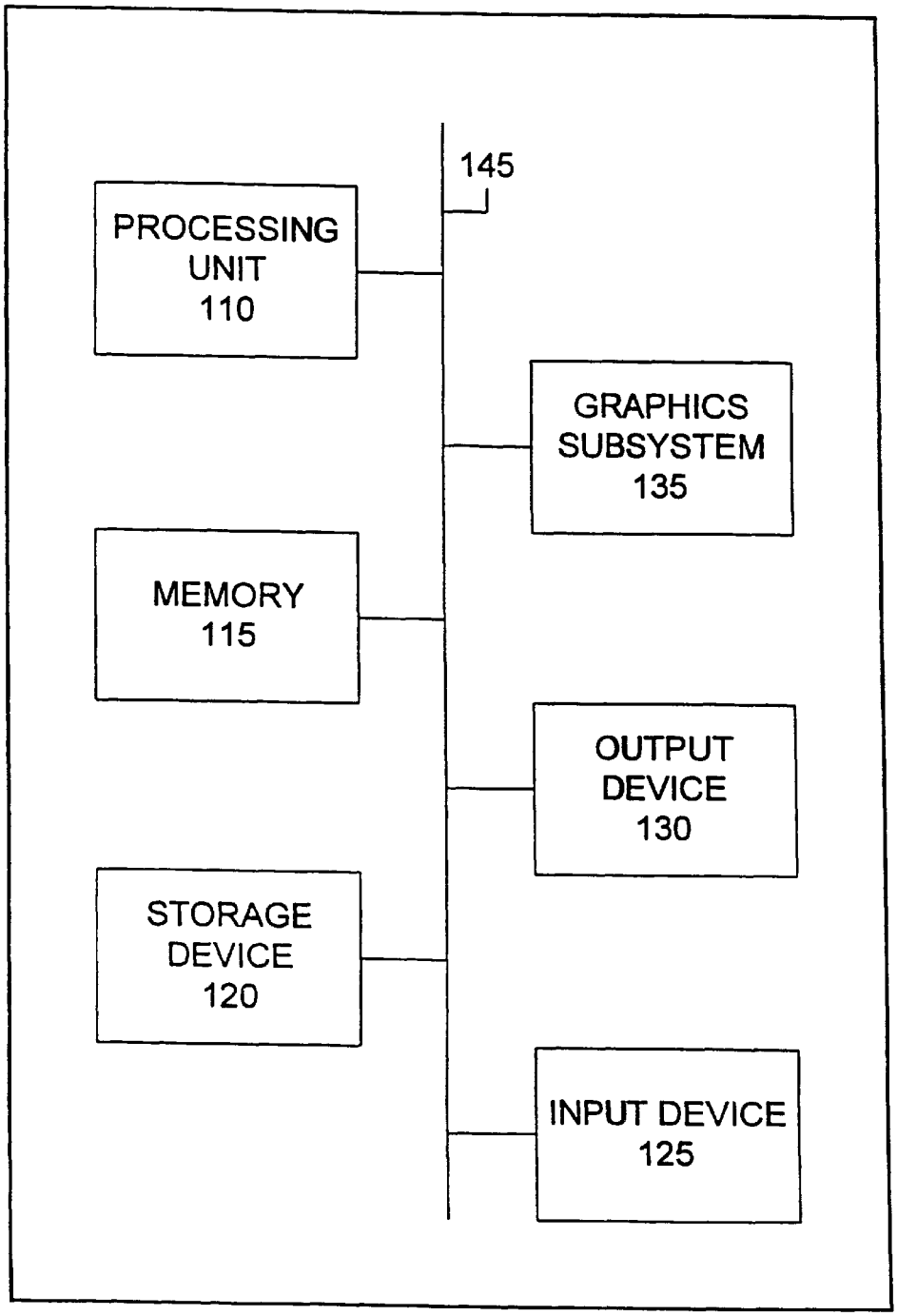

145
PROCESSING UNIT 110
GRAPHICS SUBSYSTEM 135
MEMORY 115
OUTPUT DEVICE 130
STORAGE DEVICE 120
INPUT DEVICE 125

FIG. 1 105

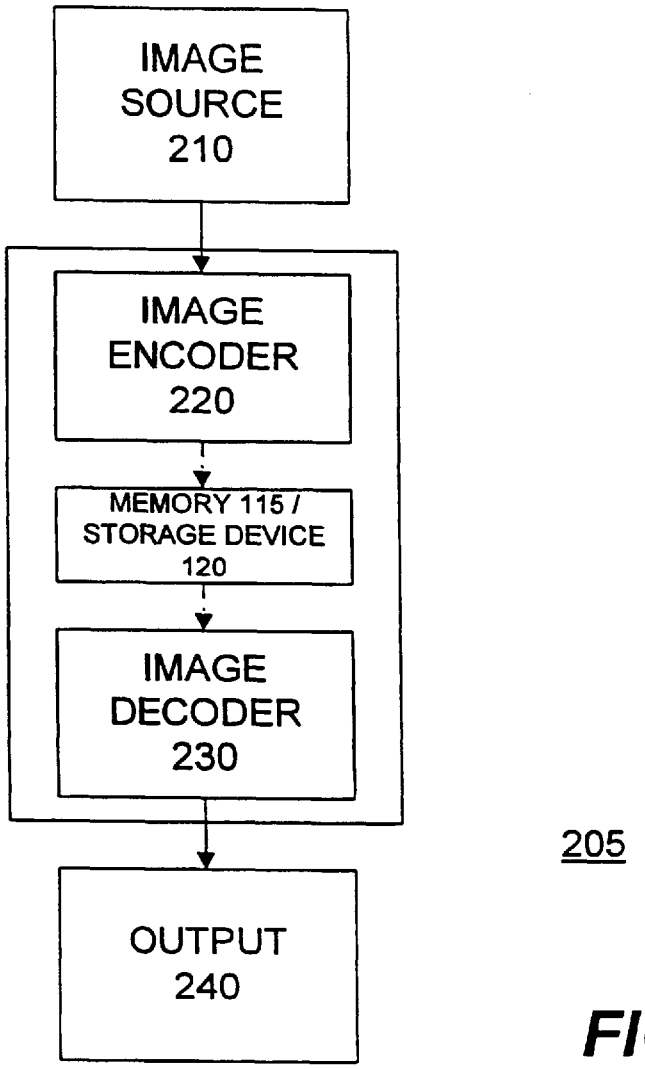


IMAGE SOURCE 210
IMAGE ENCODER 220
MEMORY 115 / STORAGE DEVICE 120
IMAGE DECODER 230
OUTPUT 240
205

FIG. 2A

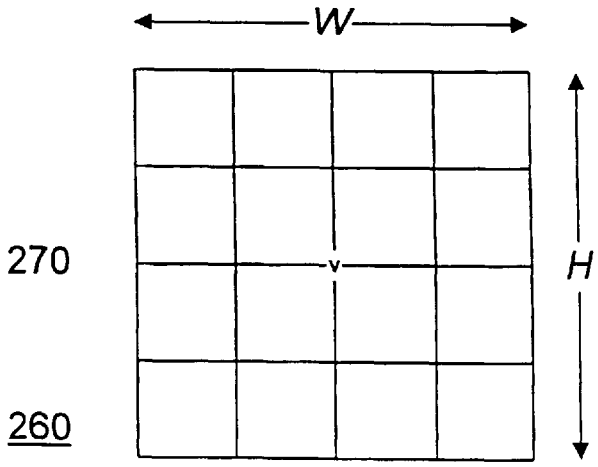

W
H
270
260

FIG. 2B

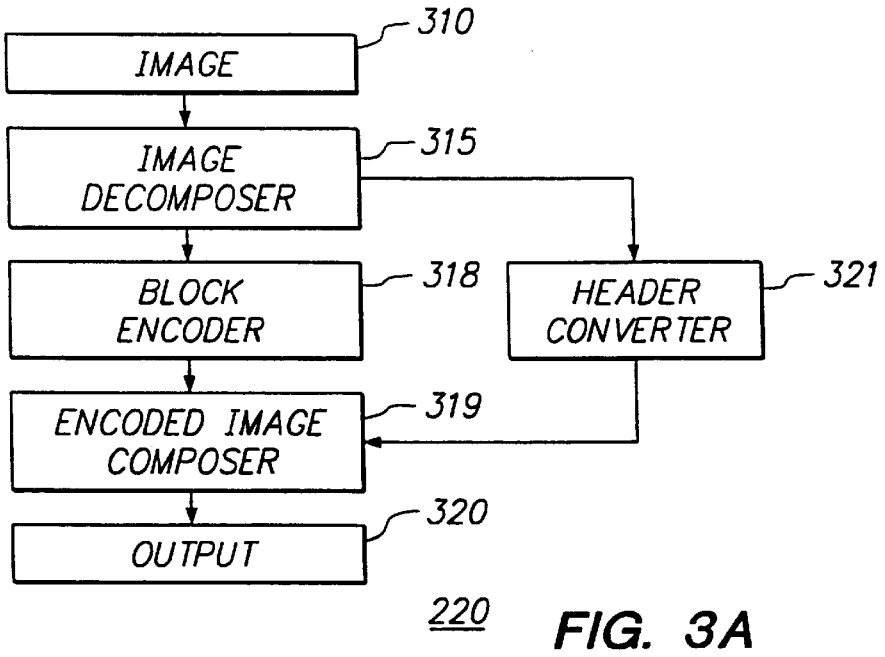


IMAGE
310
IMAGE DECOMPOSER
315
BLOCK ENCODER
318
HEADER CONVERTER
321
ENCODED IMAGE COMPOSER
319
OUTPUT
320
220

FIG. 3A

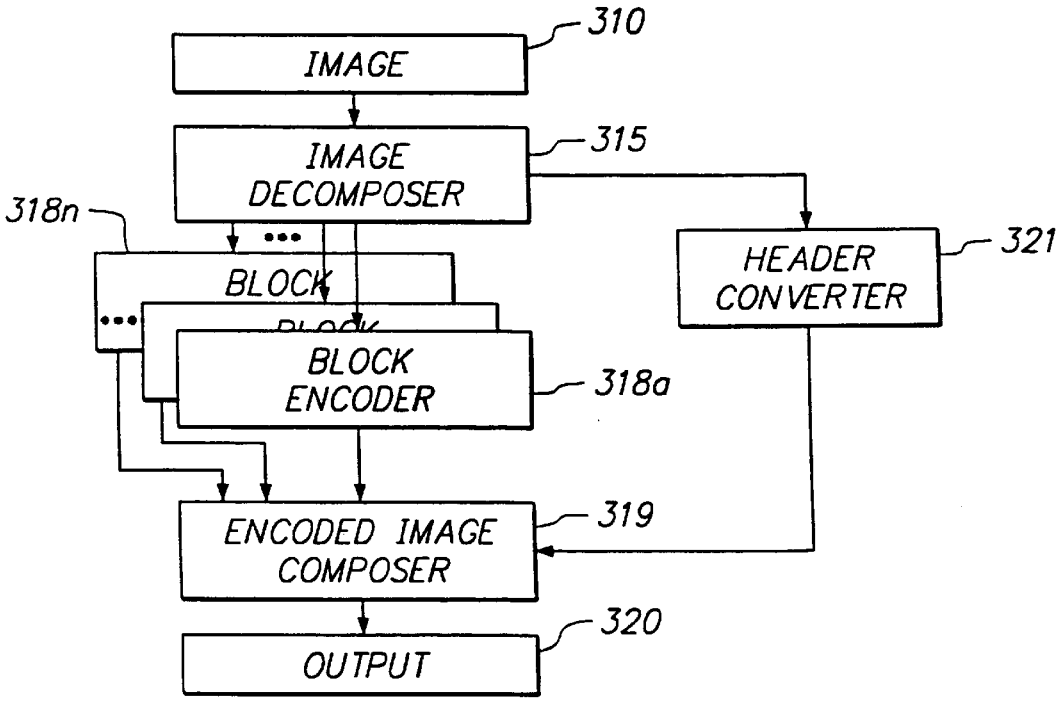


IMAGE
310
IMAGE DECOMPOSER
315
318n
BLOCK
BLOCK ENCODER
318a
HEADER CONVERTER
321
ENCODED IMAGE COMPOSER
319
OUTPUT
320
220

FIG. 3B

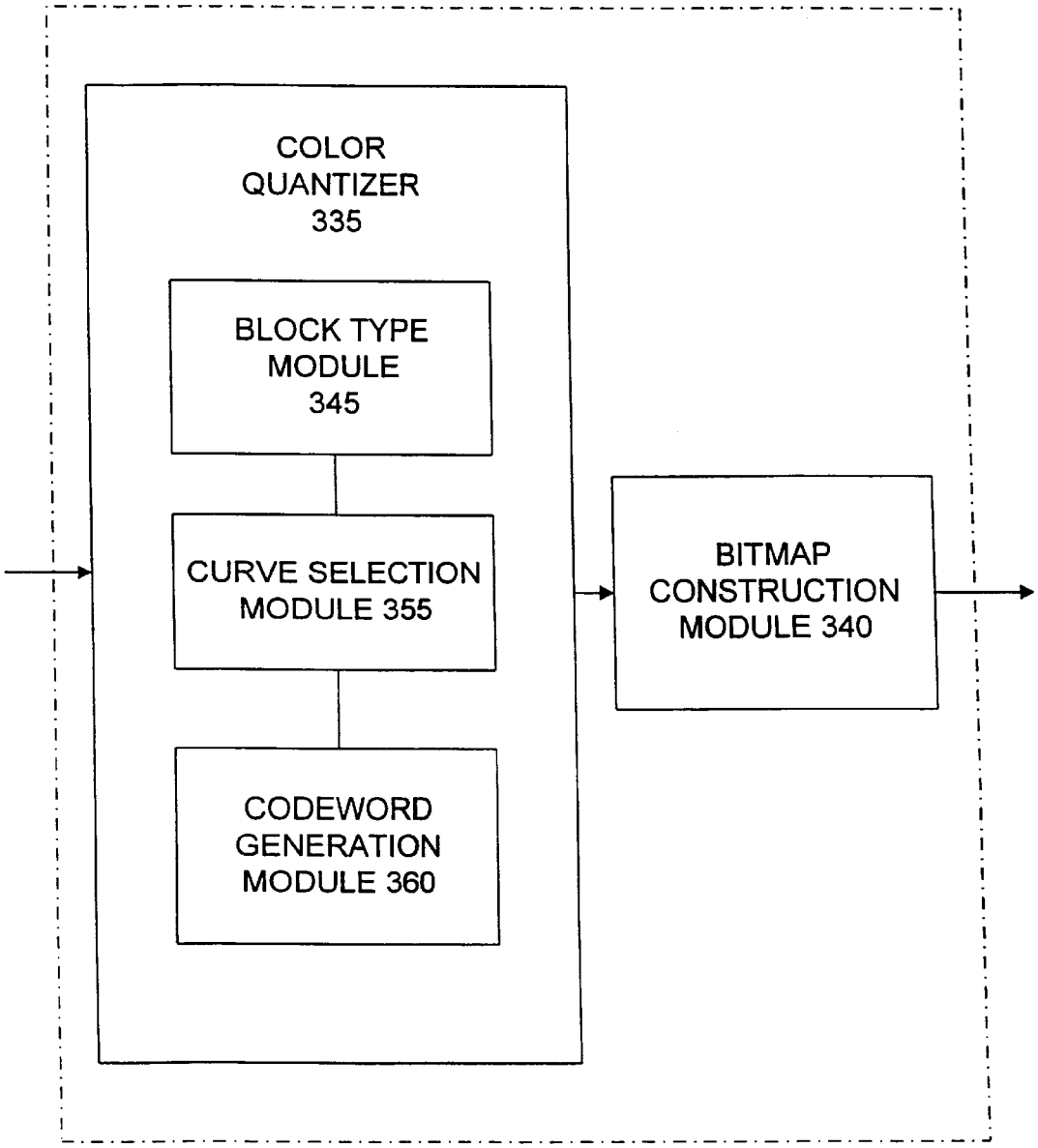
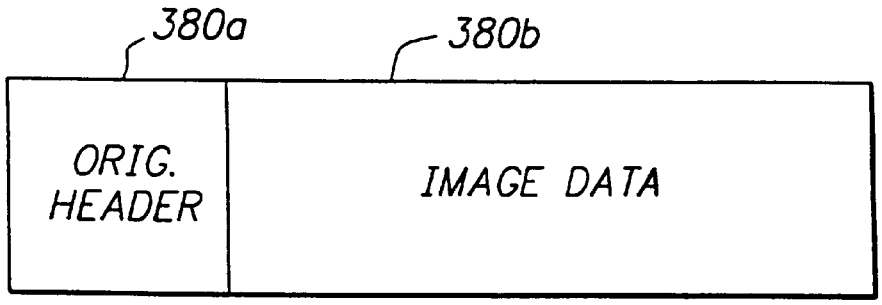
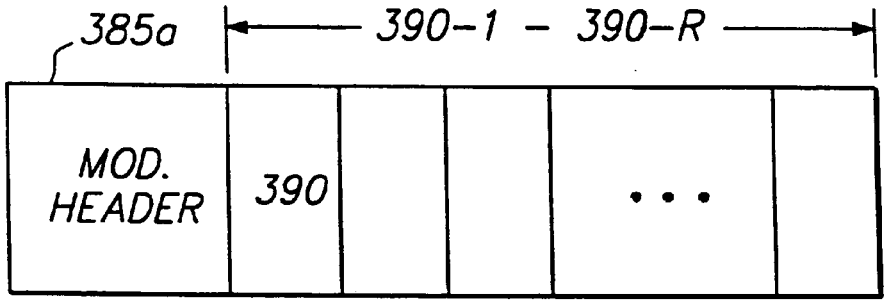

COLOR
QUANTIZER
335
BLOCK TYPE
MODULE
345
CURVE SELECTION
MODULE 355
CODEWORD
GENERATION
MODULE 360
BITMAP
CONSTRUCTION
MODULE 340
318

FIG. 3C



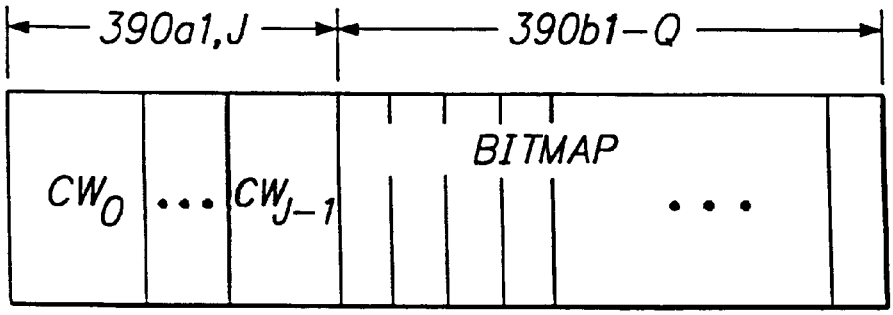
380

FIG. 3D



385

FIG. 3E



390

FIG. 3F

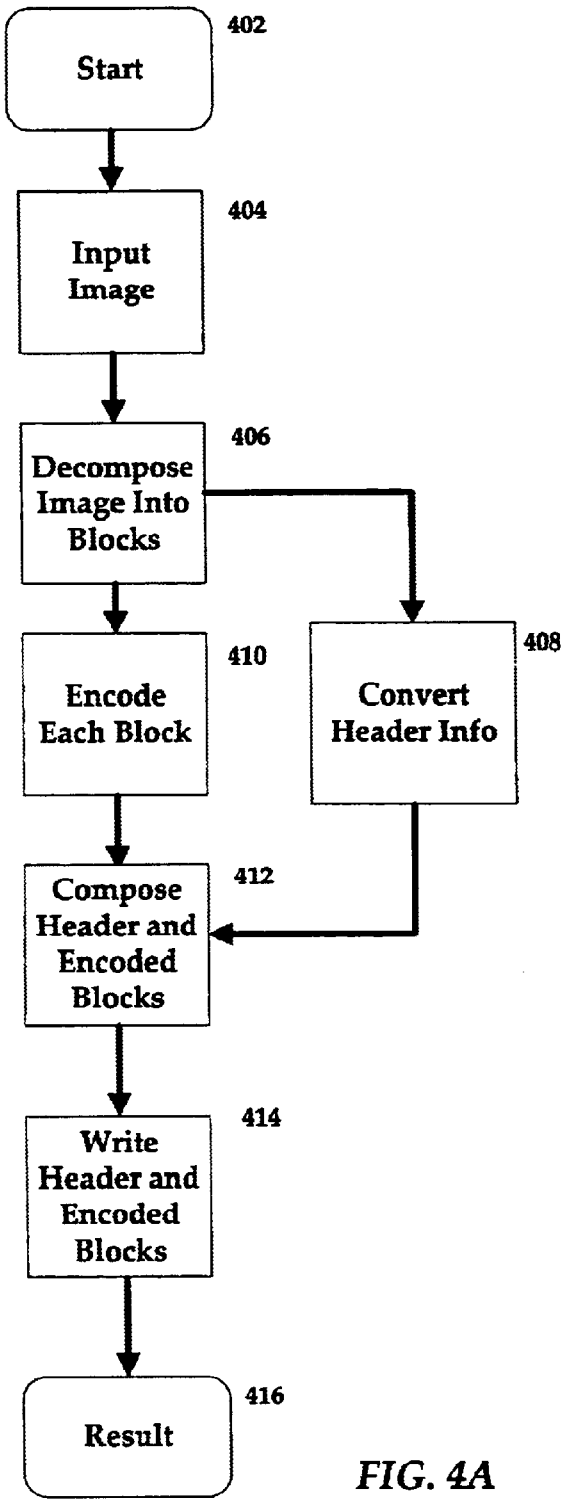

Start
402
Input Image
404
Decompose Image Into Blocks
406
Encode Each Block
410
Convert Header Info
408
Compose Header and Encoded Blocks
412
Write Header and Encoded Blocks
414
Result
416

FIG. 4A

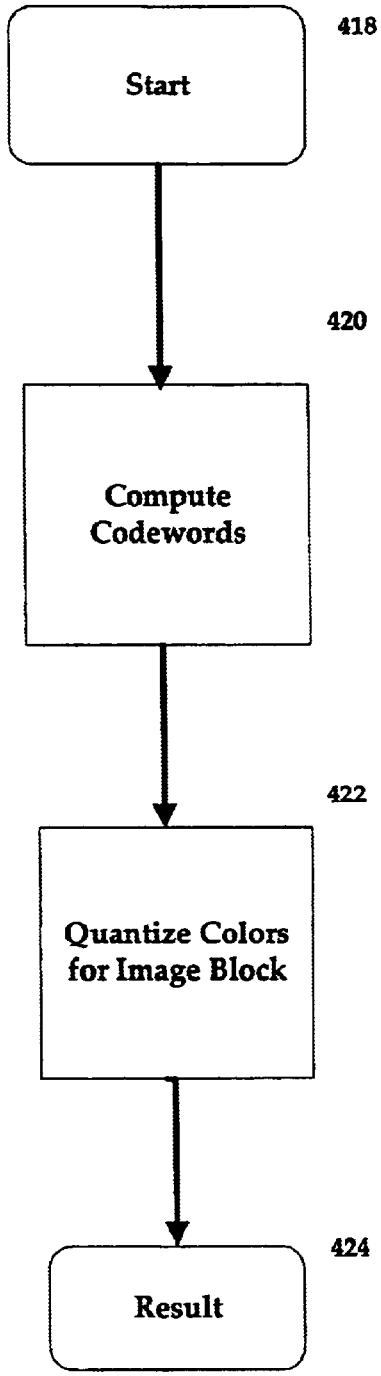

Start
418
Compute Codewords
420
Quantize Colors for Image Block
422
Result
424

FIG. 4B

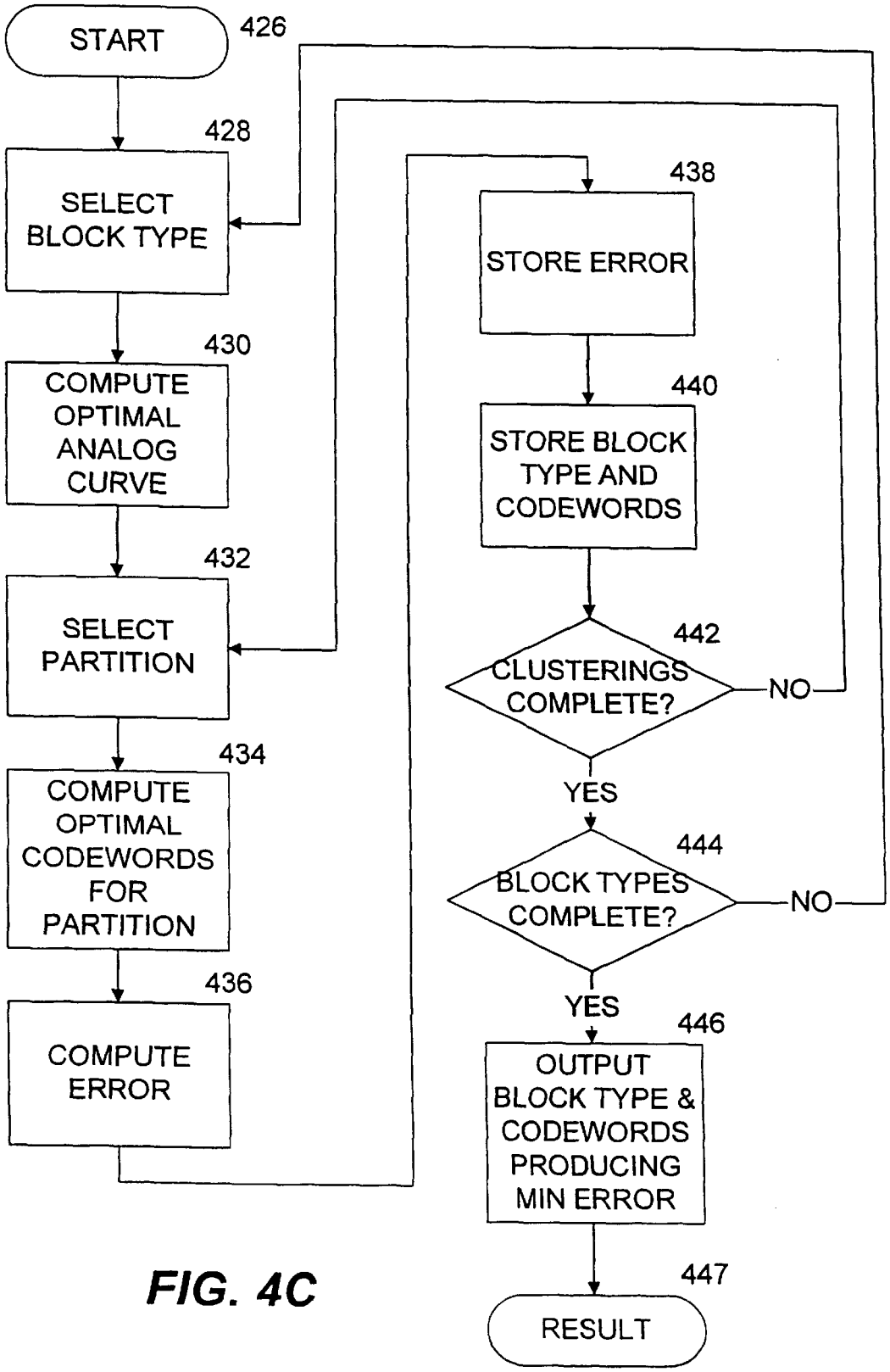

START
426
428
SELECT BLOCK TYPE
430
COMPUTE OPTIMAL ANALOG CURVE
432
SELECT PARTITION
434
COMPUTE OPTIMAL CODEWORDS FOR PARTITION
436
COMPUTE ERROR
438
STORE ERROR
440
STORE BLOCK TYPE AND CODEWORDS
442
CLUSTERINGS COMPLETE?
NO
YES
444
BLOCK TYPES COMPLETE?
NO
YES
446
OUTPUT BLOCK TYPE & CODEWORDS PRODUCING MIN ERROR
447
RESULT

FIG. 4C

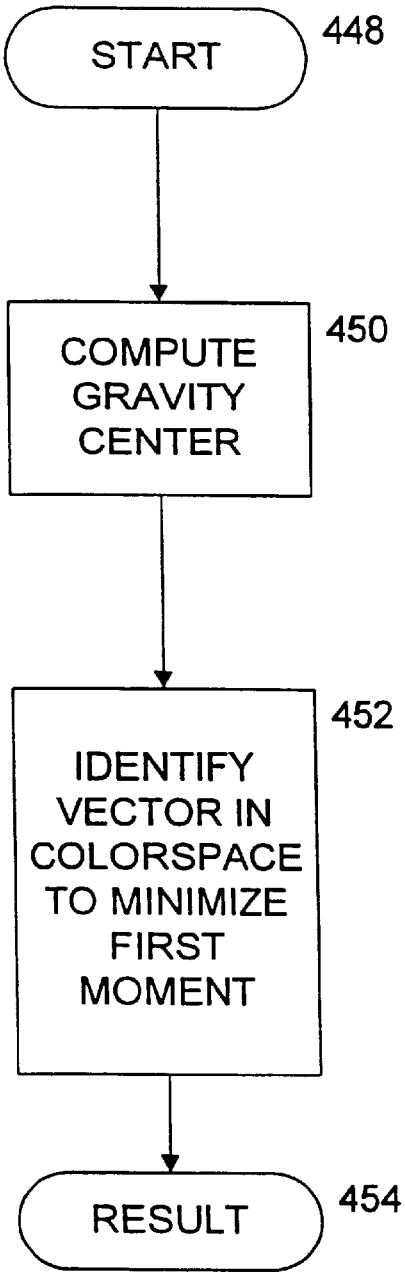

START
448
COMPUTE GRAVITY CENTER
450
IDENTIFY VECTOR IN COLORSPACE TO MINIMIZE FIRST MOMENT
452
RESULT
454

FIG. 4D

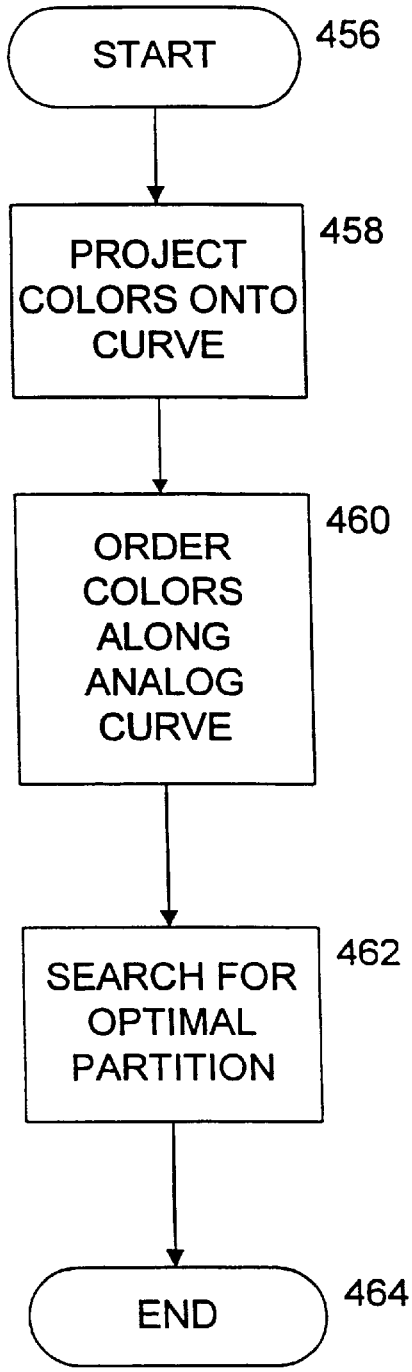

START
456
PROJECT COLORS ONTO CURVE
458
ORDER COLORS ALONG ANALOG CURVE
460
SEARCH FOR OPTIMAL PARTITION
462
END
464

FIG. 4E

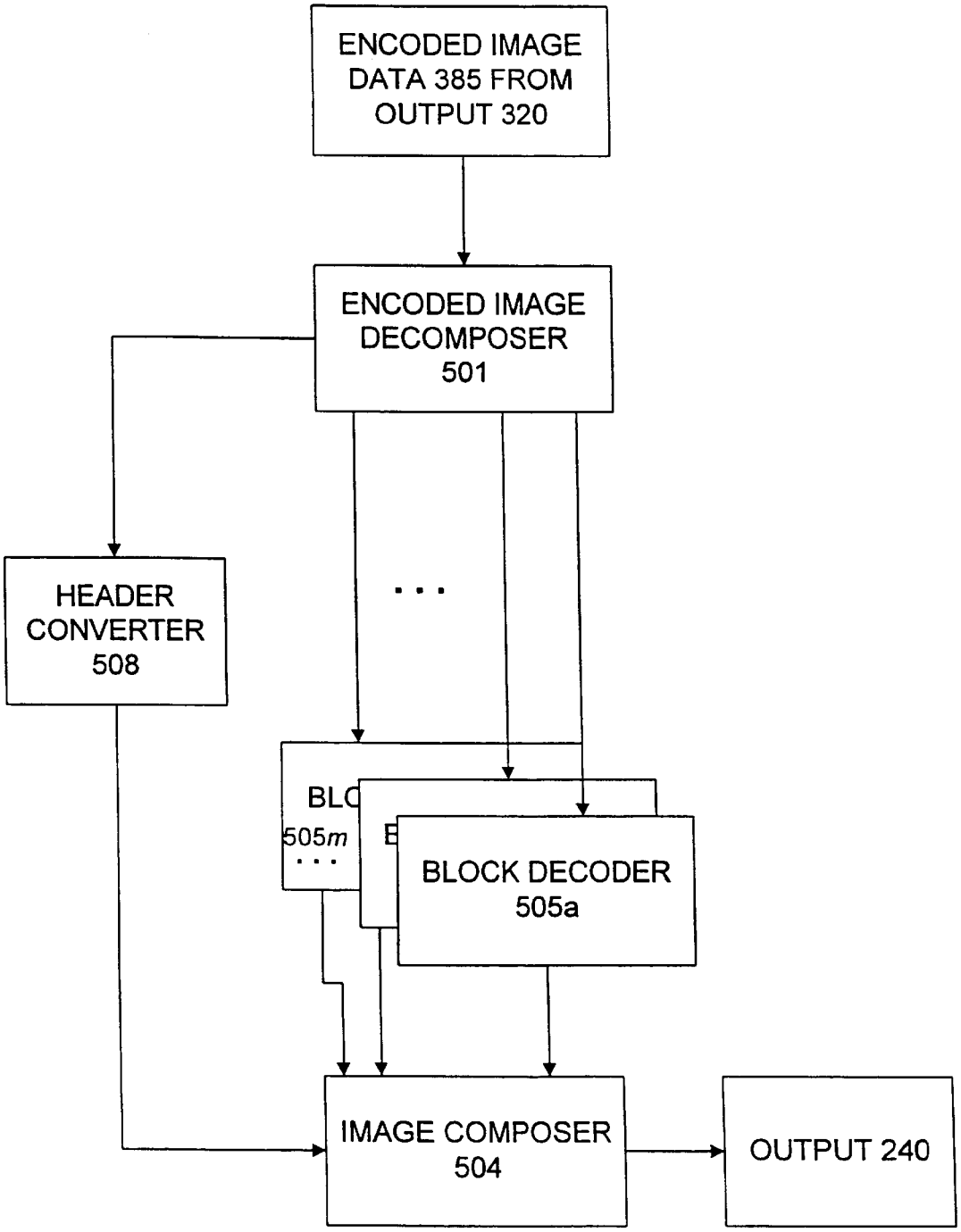

ENCODED IMAGE DATA 385 FROM OUTPUT 320
ENCODED IMAGE DECOMPOSER 501
HEADER CONVERTER 508
. . .
BLC
505m
. . .
BLOCK DECODER 505a
IMAGE COMPOSER 504
OUTPUT 240

FIG. 5A

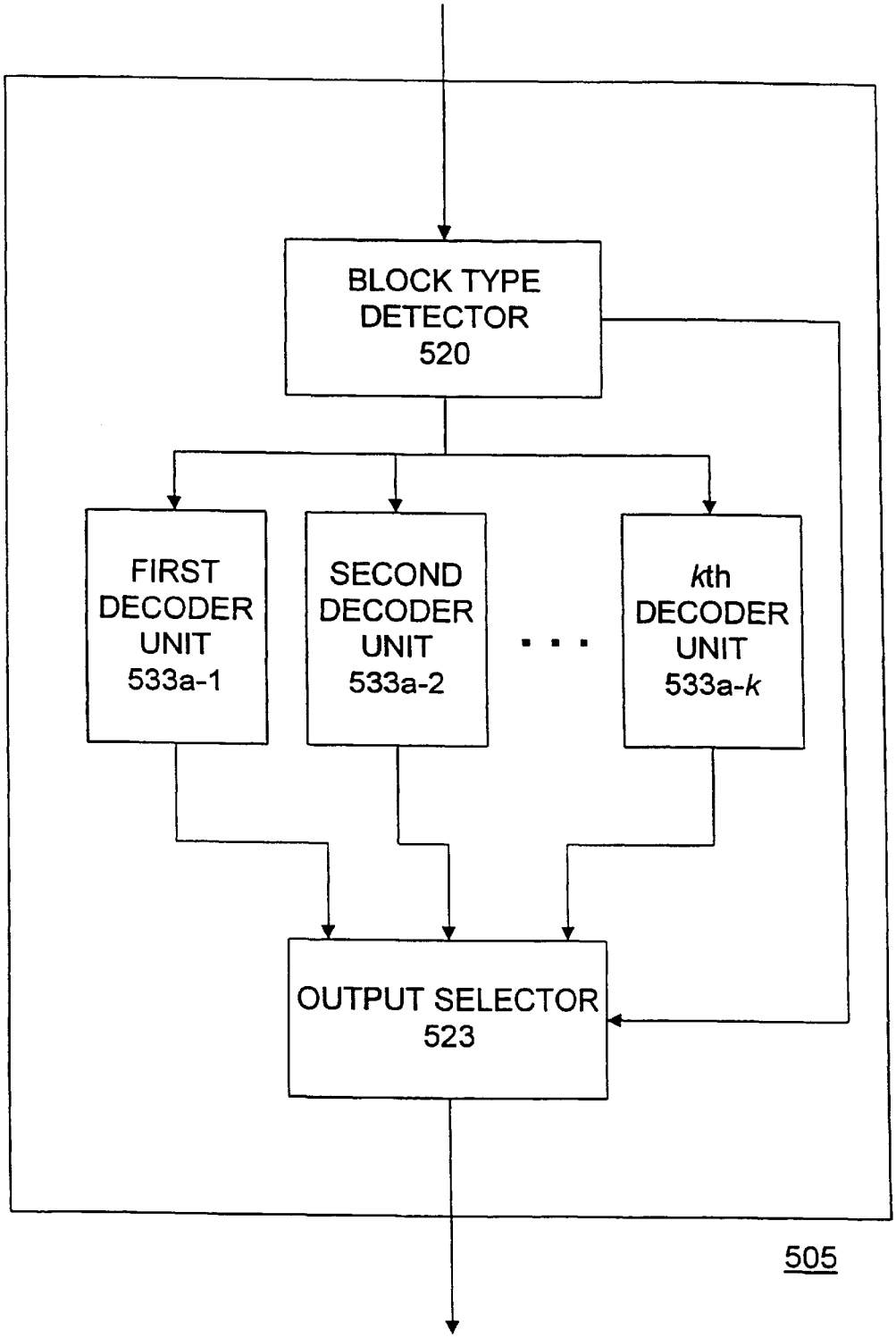

BLOCK TYPE DETECTOR 520
FIRST DECODER UNIT 533a-1
SECOND DECODER UNIT 533a-2
. . .
kth DECODER UNIT 533a-k
OUTPUT SELECTOR 523
505

FIG. 5B

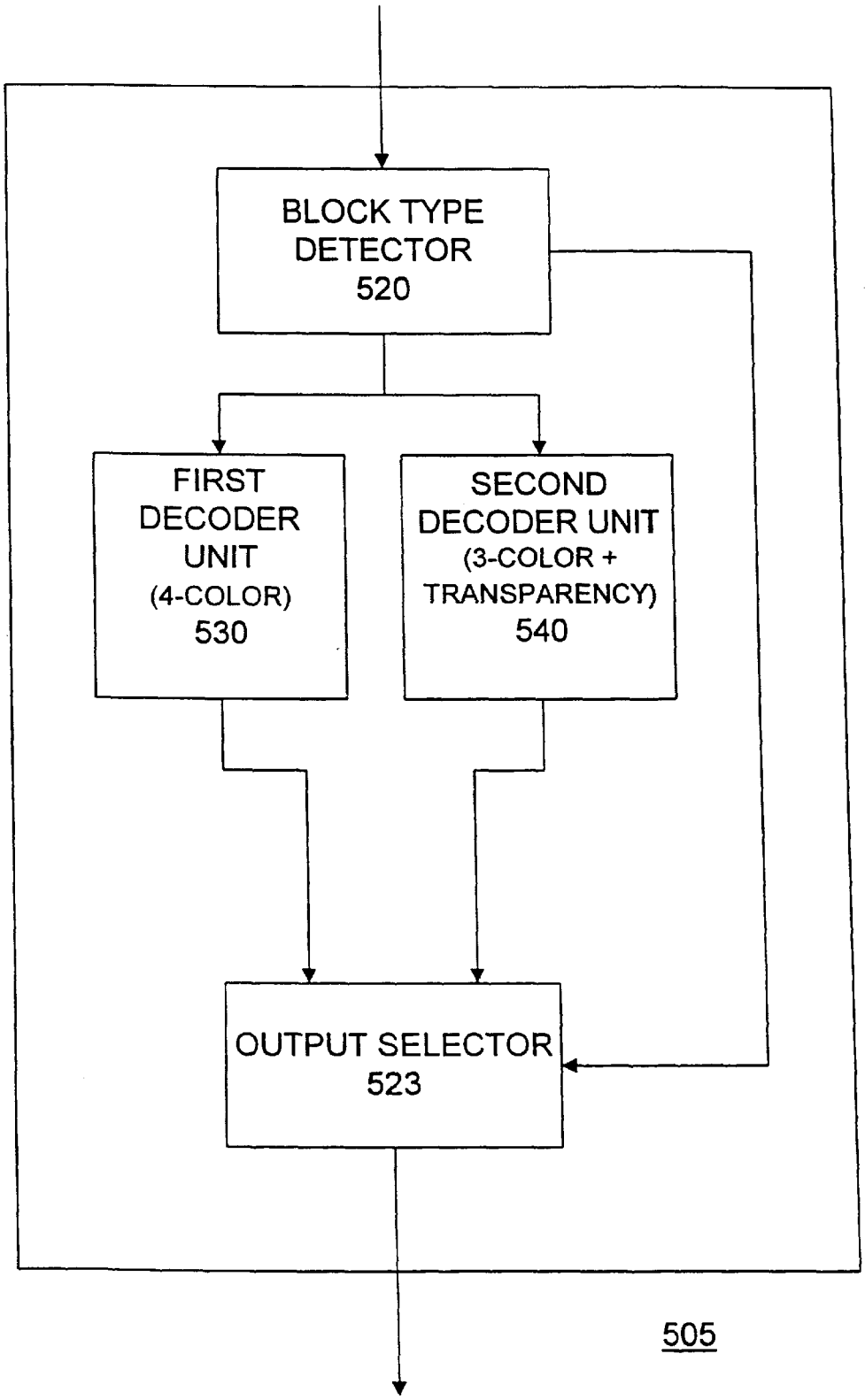

BLOCK TYPE DETECTOR 520
FIRST DECODER UNIT (4-COLOR) 530
SECOND DECODER UNIT (3-COLOR + TRANSPARENCY) 540
OUTPUT SELECTOR 523
505

FIG. 5C

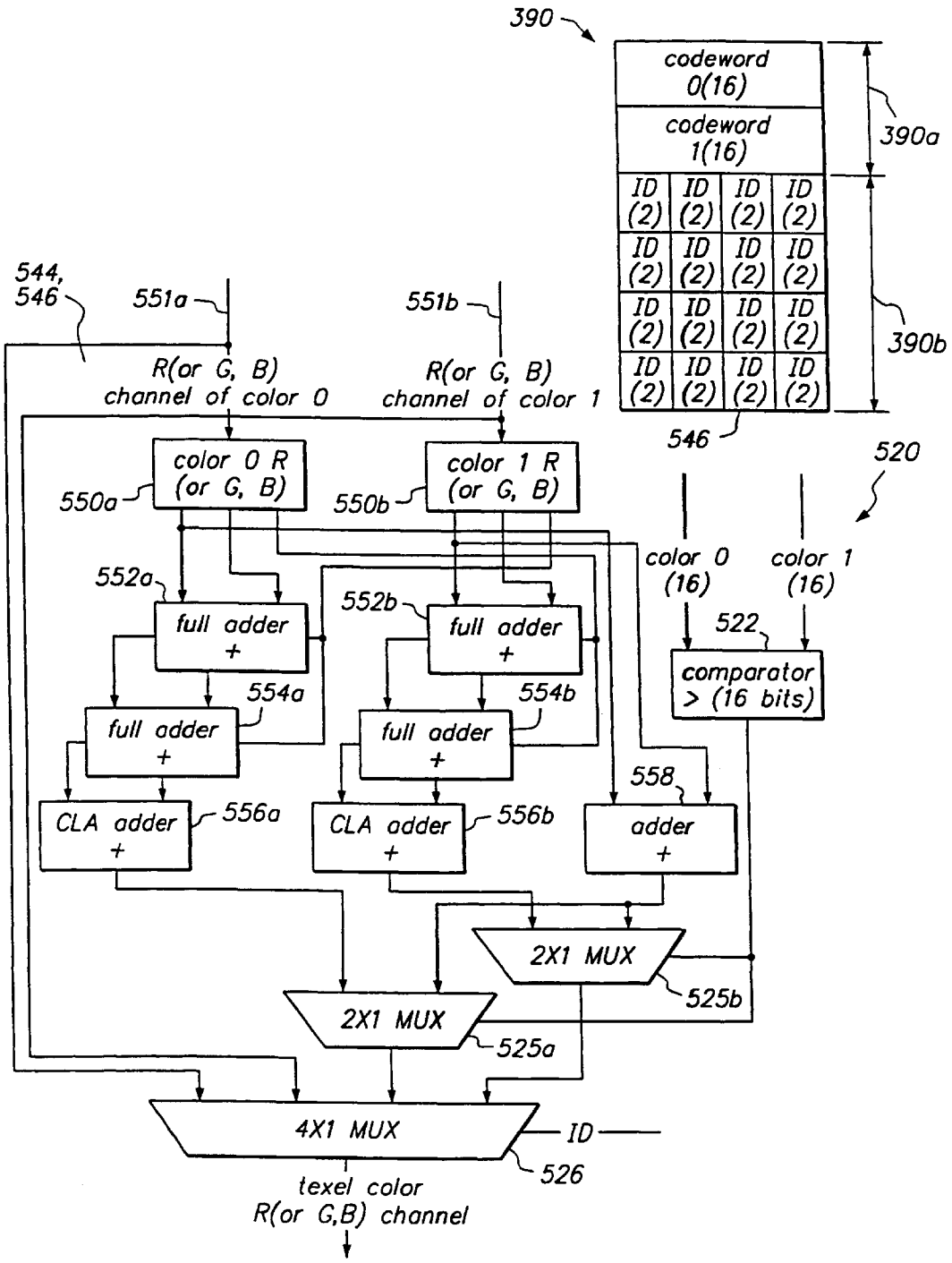

390
codeword 0(16)
codeword 1(16)
ID (2)
390a
390b
544, 546
551a
551b
R(or G, B) channel of color 0
R(or G, B) channel of color 1
546
520
color 0 R (or G, B)
color 1 R (or G, B)
550a
550b
color 0 (16)
color 1 (16)
552a
552b
522
full adder +
comparator > (16 bits)
554a
554b
558
CLA adder +
556a
556b
adder +
2X1 MUX
525b
525a
4X1 MUX
ID
526
texel color R(or G,B) channel

FIG. 5D

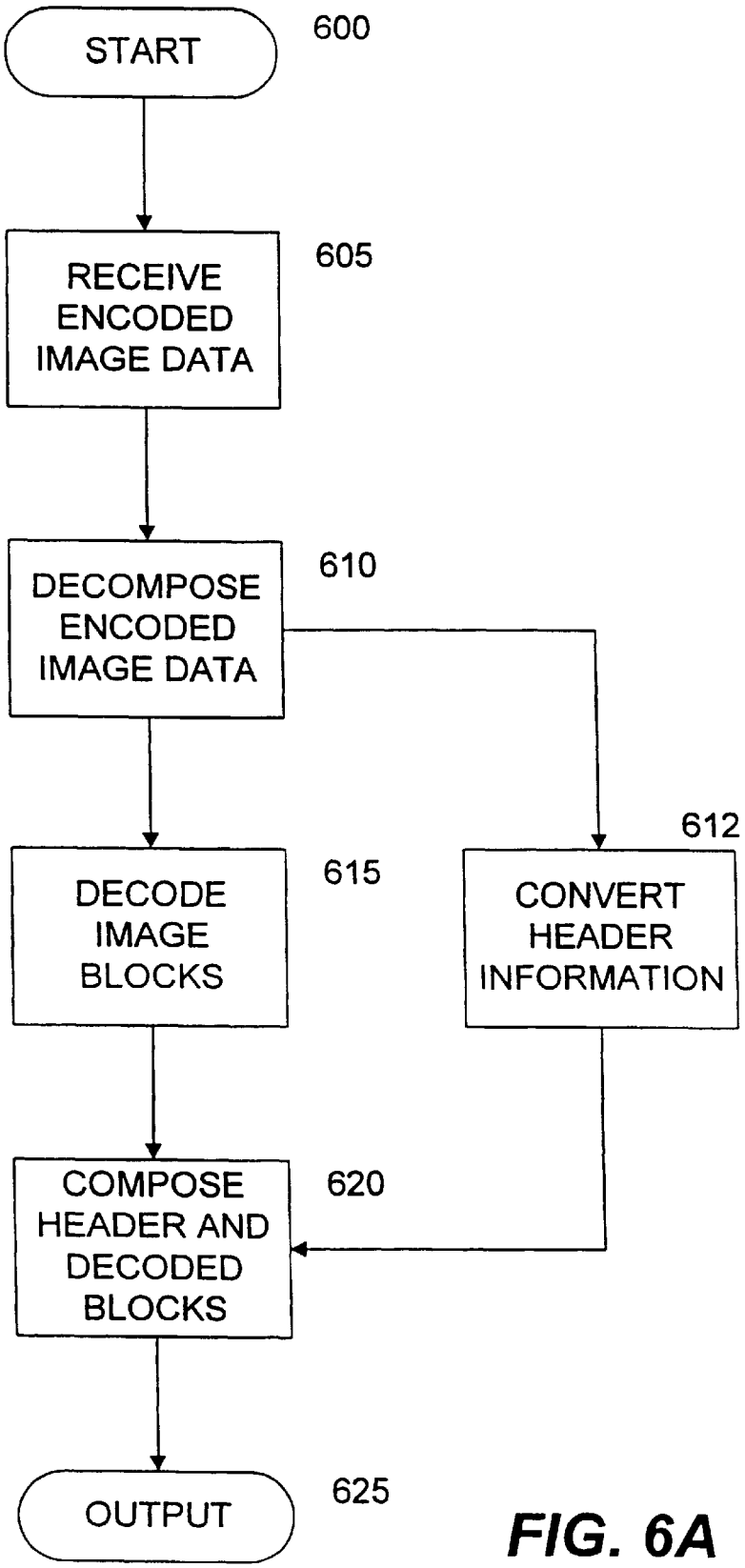

START
600
RECEIVE ENCODED IMAGE DATA
605
DECOMPOSE ENCODED IMAGE DATA
610
DECODE IMAGE BLOCKS
615
CONVERT HEADER INFORMATION
612
COMPOSE HEADER AND DECODED BLOCKS
620
OUTPUT
625

FIG. 6A

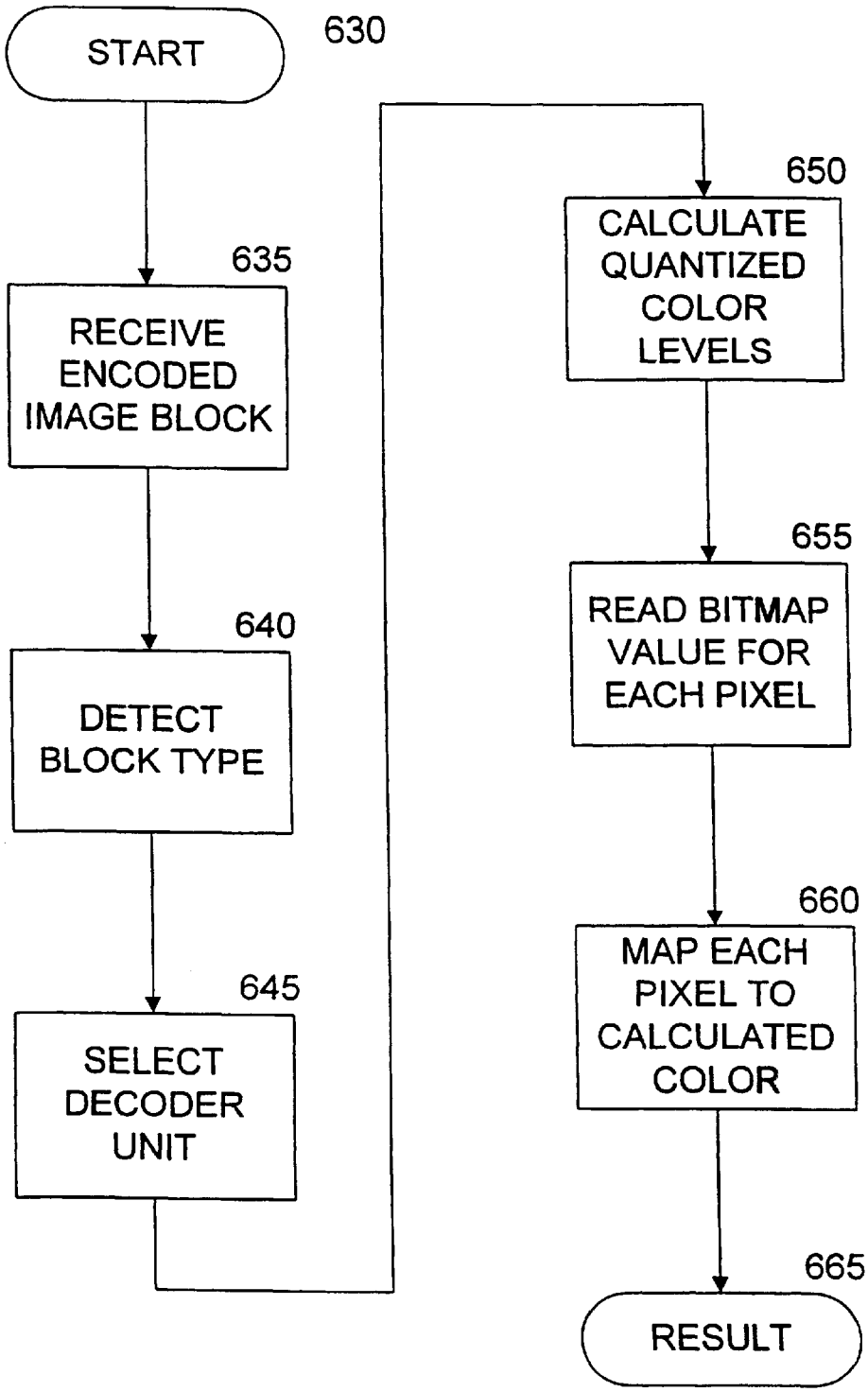

START
630
635
RECEIVE ENCODED IMAGE BLOCK
640
DETECT BLOCK TYPE
645
SELECT DECODER UNIT
650
CALCULATE QUANTIZED COLOR LEVELS
655
READ BITMAP VALUE FOR EACH PIXEL
660
MAP EACH PIXEL TO CALCULATED COLOR
665
RESULT

FIG. 6B

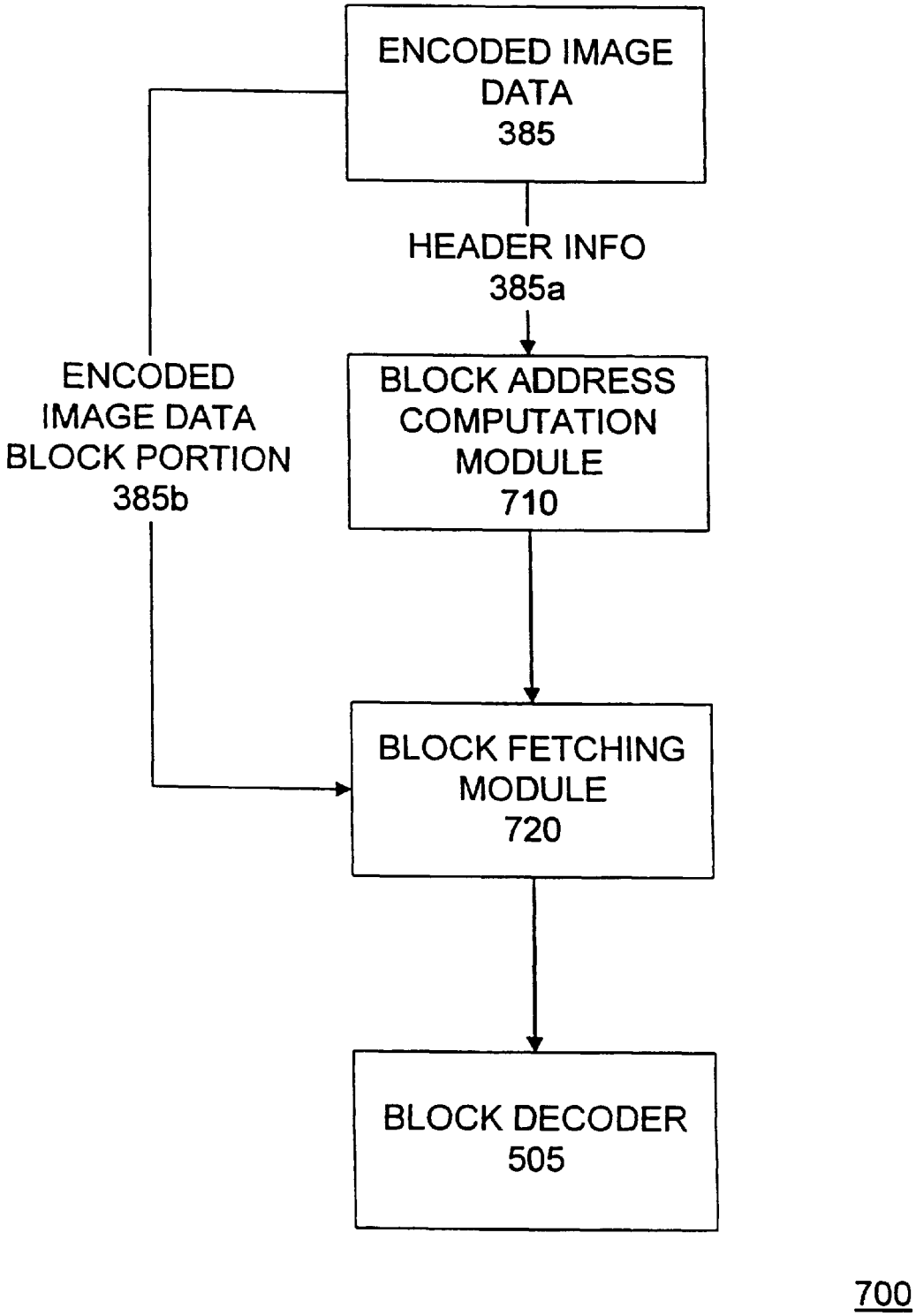

ENCODED IMAGE DATA 385
HEADER INFO 385a
ENCODED IMAGE DATA BLOCK PORTION 385b
BLOCK ADDRESS COMPUTATION MODULE 710
BLOCK FETCHING MODULE 720
BLOCK DECODER 505
700

FIG. 7A

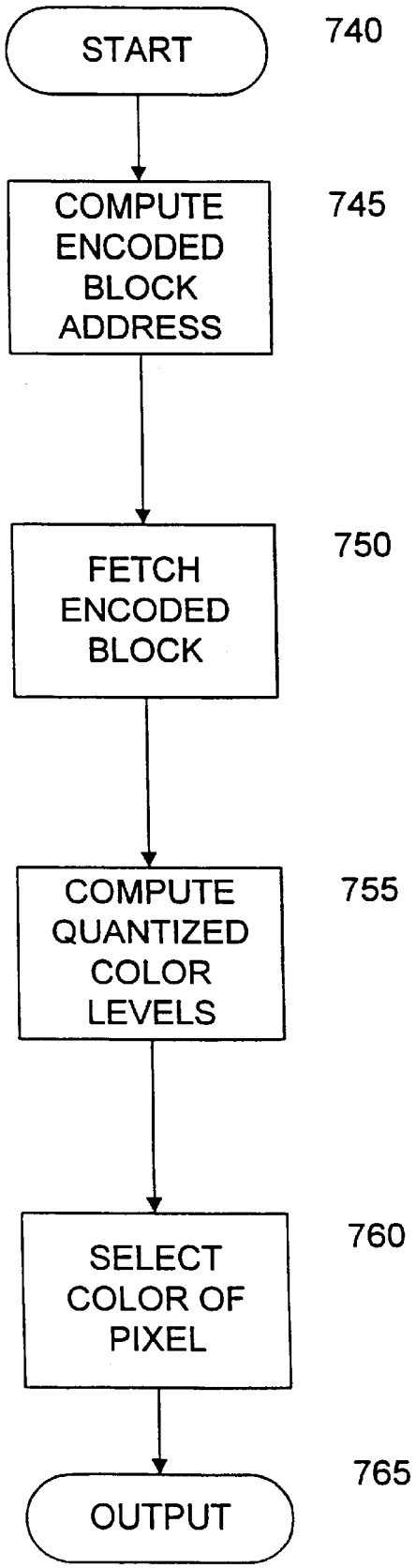

START
740
COMPUTE ENCODED BLOCK ADDRESS
745
FETCH ENCODED BLOCK
750
COMPUTE QUANTIZED COLOR LEVELS
755
SELECT COLOR OF PIXEL
760
OUTPUT
765

FIG. 7B

FIXED-RATE BLOCK-BASED IMAGE COMPRESSION WITH INFERRED PIXEL VALUES

This application is a continuation of application Ser. No. 09/351,930 filed Jul. 12, 1999, which is a continuation of application Ser. No. 08/942,860 filed Oct. 2, 1997, now U.S. Pat. No. 5,956,431 issued Sep. 21, 1999.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to image processing systems, and more specifically, to three-dimensional rendering systems using fixed-rate image compression for textures.

2. Description of the Related Art

The art of generating images, such as realistic or animated graphics on a computer is known. To generate such images requires tremendous memory bandwidth and processing power on a graphics subsystem. To reduce the bandwidth and processing power requirements, various compression methods and systems were developed. These methods and systems included Entropy or lossless encoders, discrete cosine transform or JPEG type compressors, block truncation coding, color cell compression, and others. Each of these methods and systems, however, have numerous drawbacks.

Entropy or lossless encoders include Lempel-Ziv encoders and are used for many different purposes. Entropy coding relies on predictability. For data compression using Entropy encoders, a few bits are used to encode the most commonly occurring symbols. In stationary systems where the probabilities are fixed, Entropy coding provides a lower bound for the compression than can be achieved with a given alphabet of symbols. A problem with Entropy coding is that it does not allow random access to any given symbol. The part of the compressed data preceding a symbol of interest must be first fetched and decompressed to decode the symbol which takes considerable processing time and resources as well as decreasing memory throughput. Another problem with existing Entropy methods and systems is that they do not provide any guaranteed compression factor which makes this type of encoding scheme impractical where the memory size is fixed.

Discrete Cosine Transform ("DCT") or JPEG-type compressors, allow users to select a level of image quality. With DCT, uncorrelated coefficients are produced so that each coefficient can be treated independently without loss of compression efficiency. The DCT coefficients can be quantized using visually-weighted quantization values which selectively discard the least important information.

DCT, however, suffers from a number of shortcomings. One problem with DCT and JPEG-type compressors is that they require usually bigger blocks of pixels, typically 8×8 or 16×16 pixels, as a minimally accessible unit in order to obtain a reasonable compression factor and quality. Access to a very small area, or even a single pixel involves fetching a large quantity of compressed data, thus requiring increased processor power and memory bandwidth. A second problem with DCT and JPEG-type compressors is that the compression factor is variable, therefore requiring a complicated memory management system that, in turn, requires greater processor resources. A third problem with DCT and JPEG-type compression is that using a large compression factor significantly degrades image quality. For example, the image may be considerably distorted with a form of a ringing around the edges in the image as well as noticeable color shifts in areas of the image. Neither artifact can be removed with subsequent low-pass filtering.

A fourth problem with DCT and JPEG-type compression is that such a decompressor is complex and has a significant associated hardware cost. Further, the high latency of the decompressor results in a large additional hardware cost for buffering throughout the system to compensate for the latency. Finally, a fifth problem with DCT and JPEG-type compressors is that it is not clear whether a color keyed image can be compressed with such a method and system.

Block truncation coding ("BTC") and color cell compression ("CCC") use a local one-bit quantizer on 4×4 pixel blocks. The compressed data for such a block consists of only two colors and 16-bits that indicate which one of the two colors is assigned to each of the 16 pixels. Decoding a BTC/CCC image consists of using a multiplexer with a look-up table so that once a 16-texel-block (32-bits) is retrieved from memory, the individual pixels are decoded by looking up the two possible colors for that block and selecting the color according to the associated bit from the 16 decision bits.

The BTC/CCC methods quantize each block to just two color levels resulting in significant image degradation. Further, a two-bit variation of CCC stores the two colors as eight-bit indices into a 256-entry color lookup table. Thus, such pixel blocks cannot be decoded without fetching additional information that can consume additional memory bandwidth.

The BTC/CCC methods and systems can use a three-bit per pixel scheme which store the two colors as 16-bit values (not indices into a table) resulting in pixel blocks of six bytes. Fetching such units, however, decreases system performance because of additional overhead due to memory misalignment. Another problem with BTC/CCC is that when it is used to compress images that use color keying to indicate transparent pixels, there will be a high degradation of image quality.

Therefore, there is a need for a method and system that maximizes the accuracy of compressed images while minimizing storage, memory bandwidth requirements, and decoding hardware complexities, while also compressing image data blocks into convenient sizes to maintain alignment for random access to any one or more pixels.

SUMMARY OF THE INVENTION

An image processing system includes an image encoder system and an image decoder system that are coupled together. The image encoder system includes a block decomposer and a block encoder that are coupled together. The block encoder includes a color quantizer and a bitmap construction module. The block decomposer breaks an original image into image blocks, each having a plurality of pixel values (e.g. colors) or equivalent color points. Each image block is then processed by the block encoder. Specifically, the color quantizer computes some number of base points, or codewords, that serve as reference pixel values, such as colors, from which computed or quantized pixel values are derived. The bitmap construction module then maps at least one pixel value in the image block to one of the computed or quantized colors or one of the codewords. The codewords and bitmap are output as encoded image blocks.

The decoder system includes a block decoder having one or more decoder units and an output selector. The block decoder may also include a block type detector for determining the block type of an image block. The block type determines the number of computed colors to use for map-

ping each pixel color from an image block. Using the codewords of the encoded data blocks, the comparator and the decoder units determine the computed colors for the encoded image block and map each pixel to one of the computed colors. The output selector outputs the appropriate color, which is ordered in an image composer with the other decoded blocks to output an image representative of the original image.

The present invention also includes a method of compressing an original image block having a set of original colors. The method includes: computing a set of codewords from the set of original colors; computing a set of computed colors using the set of codewords; and mapping each original color to one of the computed colors or one of the codewords to produce an index for each original color.

The compressed or encoded image block, which has a first set of indices and a set of codewords, where a set is equal to or greater than one, is decoded by: computing at least one computed color using the set of codewords; and mapping an index within the first set of indices to one of the computed colors or one of the codewords.

Those of ordinary skill in the art will readily recognize that the present invention may be practiced using any general purpose computer system, such as the computer system described below, or any "hardwired" device specifically designed to perform the method, such as but not limited to devices implemented using ASIC or FPGA technology and the like.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a data processing system in accordance with the present invention;

FIG. **2**A is a block diagram of an image processing system in accordance with the present invention;

FIG. **2**B is a graphical representation of an image block in accordance with the present invention;

FIG. **3**A is a block diagram of a first embodiment an image encoder system in accordance with the present invention;

FIG. **3**B is a block diagram of a second embodiment of an image encoder system in accordance with the present invention;

FIG. **3**C is a block diagram of an image block encoder in accordance with the present invention;

FIG. **3**D is a data sequence diagram of an original image in accordance with the present invention;

FIG. **3**E is a data sequence diagram of encoded image data of the original image output from the image encoder system in accordance with the present invention;

FIG. **3**F is a data sequence diagram of an encoded image block from the image block encoder in accordance with the present invention;

FIGS. **4**A–**4**E are flow diagrams illustrating an encoding process in accordance with the present invention;

FIG. **5**A is a block diagram of an image decoder system in accordance with the present invention;

FIG. **5**B is a block diagram of a first embodiment of a block decoder in accordance with the present invention;

FIG. **5**C is a block diagram of a second embodiment of a block decoder in accordance with the present invention;

FIG. **5**D is a logic diagram illustrating a first embodiment of a decoder unit in accordance with the present invention;

FIGS. **6**A–**6**B are flow diagrams illustrating a decoding process in accordance with the present invention;

FIG. **7**A is a block diagram of a subsystem for random access to a pixel or an image block in accordance with the present invention; and

FIG. **7**B is a flow diagram illustrating random access to a pixel or an image block in accordance with the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

FIG. **1** is a block diagram of a data processing system **105** constructed in accordance with the present invention. The data processing system **105** includes a processing unit **110**, a memory **115**, a storage device **120**, an input device **125**, an output device **130**, and a graphics subsystem **135**. In addition, the data processing system **105** includes a data bus **145** that couples each of the other components **110**, **115**, **120**, **125**, **130**, **135** of the data processing system **105**.

The data bus **145** is a conventional data bus and while shown as a single line it may be a combination of a processor bus, a PCI bus, a graphical bus, and an ISA bus. The processing unit **110** is a conventional processing unit such as the Intel Pentium processor, Sun SPARC processor, or Motorola PowerPC processor, for example. The processing unit **110** processes data within the data processing system **105**. The memory **115**, the storage device **120**, the input device **125**, and the output device **130** are also conventional components as recognized by those skilled in the art. The memory **115** and storage device **120** store data within the data processing system **105**. The input device **125** inputs data into the system while the output device **130** receives data from the data processing system **105**.

FIG. **2**A is a block diagram of an image processing system **205** constructed in accordance with the present invention. In one embodiment, the image processing system **205** runs within the data processing system **105**. The image processing system **205** includes an image encoder system **220** and an image decoder system **230**. The image processing system **205** may also include a unit for producing an image source **210** from which images are received, and an output **240** to which processed images are forwarded for storage or further processing. The image encoder system **220** is coupled to receive an image from the image source **210**. The image decoder system **230** is coupled to output the image produced by the image processing system **205**. The image encoder system **220** is coupled to the image decoder system **230** through a data line and may be coupled via a storage device **120** and/or a memory **115**, for example.

Within the image encoder system **220**, the image is broken down into individual blocks and processed before being forwarded to, e.g., the storage device **140**, as compressed or encoded image data. When the encoded image data is ready for further data processing, the encoded image data is forwarded to the image decoder system **230**. The image decoder system **230** receives the encoded image data and decodes it to generate an output that is a representation of the original image that was received from the image source **210**.

FIGS. **3**A and **3**B are block diagrams illustrating two separate embodiments of the image encoder system **220** of the present invention. The image encoder system **220** includes an image decomposer **315**, a header converter **321**, one or more block encoders **318** (**318***a*–**318***n*, where n is the nth encoder, n being any positive integer), and an encoded image composer **319**. The image decomposer **315** is coupled to receive an original image **310** from a source, such as the image source **210**. The image decomposer **315** is also coupled to the one or more block encoders **318** and to the

header converter **321**. The header converter **321** is also coupled to the encoded image composer **319**. Each block encoder **318** is also coupled to the encoded image composer **319**. The encoded image composer **319** is coupled to the output **320**.

The image decomposer **315** receives the original image **310** and forwards information from a header of the original image **310** to the header converter **321**. The header converter **321** modifies the original header to generate a modified header, as further described below. The image decomposer **315** also breaks, or decomposes, the original image **310** into R number of image blocks, where R is some integer value. The number of image blocks an original image **310** is broken into may depend on the number of image pixels. For example, in a preferred embodiment an image **310** comprised of A image pixels by B image pixels will typically be (A/4)*(B/4) blocks, where A and B are integer values. For example, where an image is 256 pixels by 256 pixels, there will be 64×64 blocks. In other words, the image is decomposed such that each image block is 4 pixels by 4 pixels (16 pixels). Those skilled in the art will recognize that the number of pixels or the image block size may be varied, for example m×n pixels, where m and n are positive integer values.

Briefly turning to FIG. **2**B, there is illustrated an example of a single image block **260** in accordance with the present invention. The image block **260** is comprised of pixels **270**. The image block **260** may be defined as an image region W pixels **270** in width by H pixels **270** in height, where W and H are integer values. In a preferred embodiment, the image block **260** is comprised of W=4 pixels **270** by H=4 pixels **270** (4×4).

Turning back to FIGS. **3**A and **3**B, each block encoder **318** receives an image block **260** from the image decomposer **315**. Each block encoder **318** encodes or compresses each image block **260** that it receives to generate an encoded or compressed image block. Each encoded image block is received by the encoded image composer **319** which orders the encoded blocks in a data file. The data file from the encoded image composer **319** is concatenated with a modified header from the header converter **321** to generate an encoded image data file that is forwarded to the output **320**. Further, it is noted that having more than one block encoder **318***a*–**318***n* allows for encoding multiple image blocks simultaneously, one image block per block encoder **318***a*–**318***n*, within the image encoder system **220** to increase image processing efficiency and performance.

The modified header and the encoded image blocks together form the encoded image data that represents the original image **310**. The function of each element of the image encoder system **220**, including the block encoder **318**, will be further described below with respect to FIGS. **4**A–**4**E.

The original image **310** may be in any one of a variety of formats including red-green-blue ("RGB"), YUV **420**, YUV **422**, or a proprietary color space. It may be useful in some cases to convert to a different color space before encoding the original image **310**. It is noted that in one embodiment of the present invention, each image block **260** is a 4×4 set of pixels where each pixel **270** is 24-bits in size. For each pixel **270** there are 8-bits for a Red(R)-channel, 8-bits for a Green(G)-channel, and 8-bits for a Blue(B)-channel in a red-green-blue ("RGB") implementation color space. Further, each encoded image block is also a 4×4 set of pixels, but, each pixel is only 2-bits in size and has an aggregate size of 4-bits as will be further described below.

FIG. **3**C is a block diagram illustrating a block encoder **318** of the present invention in greater detail. The block encoder **318** includes a color quantizer **335** and a bitmap construction module **340**. The color quantizer **335** is coupled to the bitmap construction module **340**. Further, the color quantizer **335** further emphasizes a block type module **345**, a selection module **355**, and a codeword generation module **360**. The block type module **345** is coupled to the selection module **355**. The selection module **355** is coupled to the codeword generation module **360**.

Each image block **260** of the decomposed original image **310** is received and initially processed by the color quantizer **335** before being forwarded to the bitmap construction module **340** for further processing. The bitmap construction module **340** outputs encoded image blocks for the encoded image composer **319** to order. The bitmap construction module **340** and the color quantizer **335**, including the block type module **345**, the selection module **355**, and the codeword generation module **360**, are further discussed below in FIGS. **4**A–**4**E.

Briefly, FIG. **3**D is a diagram of a data sequence or string **380** representing the original image **310** that is received by the block decomposer **315**. The data string **380** of the original image **310** includes an a-bit header **380***a* and a b-bit image data **380***b*, where a and b are integer values. The header **380***a* may include information such as the pixel width of the image **310**, the pixel height of the image **310**, and the format of the image **310**, e.g., the number of bits to the pixel in RGB or YUV format, for example, as well as other information. The image data is the data **380***b* representing the original image **310** itself.

FIG. **3**E is a diagram of a data sequence or string **385** representing encoded image data **385** that is generated and output **320** by the image encoder system **220**. The data string for the encoded image data **385** includes a modified header portion **385***a* and an encoded image block portion **390-1**–**390**-R. The modified header portion **385***a* is generated by the header converter **321** from the original header **380***a* for the original image **310**. The modified header generated by the header converter **321** includes information about file type, a number of bits per pixel of the original image **310**, addressing into the original image **310**, other miscellaneous encoding parameters, as well as the width and height information indicating the size of that original image **310**. The encoded image block portion **390-1**-R includes the encoded image blocks **390-1**–**390**-R from the block encoders **318**, where R is an integer value that is the number of blocks resulting from the decomposed original image **310**.

FIG. **3**F is a diagram of a data sequence or string **390** representing an encoded image block in accordance with the present invention. It is understood that the data string **390** representing the encoded image block may be similar to any one of the encoded image blocks **390-1**–**390**-R shown in the encoded image data string **385**.

The data string **390** of the encoded image block includes a codeword section **390***a* which includes J codewords, where J is an integer value, and a bitmap section **390***b*. The codeword section **390***a* includes J codewords **390***a* that are used to compute the colors indexed by the bitmap **390***b*. A codeword is a n-bit data string, where n is an integer value, that identifies a pixel property, for example a color component. In a preferred embodiment, there are two 16-bit codewords **390***a*, CW**0**, CW**1** (J=2). The bitmap is a Q-bit data portion and is further discussed below in FIG. **4**B.

Further, in a preferred embodiment, each encoded image block is 64-bits, which includes two 16-bit codewords and

a 32-bit (4×4×2 bit) bitmap **395**. Encoding the image block **260** as described provides greater system flexibility and increased data processing efficiency as will be further discussed below.

FIGS. **4**A–**4**E describe the operation of the image encoder system **220**. FIG. **4**A describes the general operation of the image encoder system **220**. At the start **402** of operation, data string **380** of the original image **310**, that includes the a-bit header **380***a* and the b-bit image data **380***b*, is input **404** into the block decomposer **315** from the image source **210**. The block decomposer **315** decomposes **406** the original image **310** to extract the a-bit header **380***a* and it to the header converter **321**. The block decomposer also **315** decomposes, **406** the original image **310** into image blocks. Each image block **260** is independently compressed, or encoded, **410** in the one or more block encoders **318**.

The header converter **321** converts **408** the a-bit header to generate a modified header **385***a*. The modified header **385***a* is forwarded to the encoded image composer **319**. Simultaneous with the header converter **321** converting **408** the a-bit header, each image block is encoded **410** by the one or more image encoders **318***a*–**318***n* to generate the encoded image blocks **390-1**–**390**-R. Again, it is noted that each image block **260** may be processed sequentially in one block encoder **318**a or multiple image blocks **260** may be processed in parallel in multiple block encoders **318***a*–**318***n*.

The encoded image blocks **390** are output from the block encoders **318** and are placed into a predefined order by the encoded image composer **319**. In a preferred embodiment, the encoded image blocks **390** are ordered in a file from left to right and top to bottom in the same. order in which they were broken down by the block decomposer **315**. The image encoder system **220** continues by composing **412** the modified header information **385***a* from the header converter **321** and the encoded image blocks **390**. Specifically, the modified header **385***a* and the ordered encoded image blocks **390** are concatenated to generate the encoded image data file **385**. The encoded image data file **385** is written **414** as encoded output **320** to the memory **115**, the storage device **120**, or the output device **130**, for example.

FIG. **4**B shows the encoding process **410** for the encoder system **220** described above in FIG. **2**. At the start **418** of operation, codewords are computed **420**. As discussed above in FIG. **3**F, in a preferred embodiment there are two codewords **390***a*, CW**0**, CW**1**. The process for computed codewords is further described below in FIG. **4**C.

Once the codewords are computed **420** pixel values or properties, such as colors, for the image block **260** are computed or quantized **422**. Specifically, the codewords **390***a* provide points in a pixel space from which M quantized pixel values may be inferred, where M is an integer value. The M quantized pixel values are a limited subset of pixels in a pixel space that are used to represent the current image block. The process for quantizing pixel values, and more specifically colors, will be described below in FIGS. **4**D and **4**E. Further, it is noted that the embodiments will now be described with respect to colors of a pixel value although one skilled in the art will recognize that in general any pixel value may be used with respect to the present invention.

In a preferred embodiment, each pixel is encoded with two bits of data which can index one of M quantized colors (M=4). Further, in a preferred embodiment the four quantized colors are derived from the two codewords **390***a* where two colors are the codewords themselves and the other two colors are inferred from the codewords, as will be described below. It is also possible to use the codewords **390***a* so that there is one index to indicate a transparent color and three indices to indicate colors, of which one color is inferred.

In a preferred embodiment, the bitmap **390***b* is a 32-bit data string.

The bitmap **390***b* and codewords **390***a* are output **424** as a 64-bit data string representing an encoded image block **390**. Specifically, the encoded image block **390** includes the two 16-bit codewords **390***a* (n=16) and a 32-bit bitmap **390***b*. Each codeword **390***a* CW**0**, CW**1** that is a 16-bit data string includes a 5-bit red-channel, 6-bit green-channel, and 5-bit blue-channel.

Each of the encoded image blocks **390** is placed together **390***a***1**–**390***a*R, and concatenated with header information **385***a* derived from the original header **380***a* of the original image **310**. The resulting **424** output is the encoded image data **385** representing the original image **310**.

FIG. **4**C describes the process for computing the codewords for the image blocks **260** in more detail. At the start **426** of the process, the color quantizer **335** uses the block type module **345** to select **428** the first block type for the image block **260** that is being processed. For example, one block type selected **428** may be a four-color and another block type selected **428** may be a three-color plus transparency, where the colors within the particular block type have equidistant spacing in a color space.

Those of ordinary skill in the art will readily recognize that selecting a block type for each image is not intended to be limiting in any way Instead, the present invention may be limited to processing image blocks that are of a single block type. This eliminates the need to distinguish between different block types, such as the three and four color block types discussed above. Consequently, the block type module **345** in FIG. **3**B and reference number **428** in FIG. **4**C are optional and are not intended to limit the present invention in any way.

Once the block type is selected **428**, the process computes **430** an optimal analog curve for the block type. Computation **430** of the optimal analog curve **430** will be further described below in FIG. **4**D. The analog curve is used to simplify quantizing of the colors in the image block. After computing **430** the optimal analog curve, the process selects **432** a partition of the points along the analog curve. A partition may be defined as a grouping of indices {1 . . . (W×H)} into M nonintersecting sets. In a preferred embodiment, the indices (1 . . . 16) are divided into three or four groups, or clusters, (M=3 or 4) depending on the block type.

Once a partition is selected **432**, the optimal codewords for that particular partition are computed **434**. Computation **434** of the optimal codewords is further described below in FIG. **4**E. In addition to computing **434** the codewords, an error value (squared error as describe below) for the codewords is also computed **436**. Computation **436** of the error values is further described below with respect to FIG. **4**E also. If the computed **436** error value is the first error value it is stored. Otherwise, the computed **436** error value is stored **438** only if it is less than the previously stored error value. For each stored **438** error value, the corresponding block type and codewords are also stored **440**. It is noted that the process seeks to find the block type and codewords that minimize the error function.

The process continues by determining **442** if the all the possible partitions are complete. If there are more partitions possible, the process selects **432** the next partition and once again computes **434** the codewords, computes **436** the

associated error value, and stores **438** the error value and stores **440** associated block type and codewords only if the error value is less than the previously stored error value.

After all the possible partitions are completed, the process determines **444** whether all the block types have been selected. If there are more block types, the process selects **428** the next block type. Once again, the process will compute **430** the optimal analog curve, select **432**, **442** all the possible partitions, for each partition it will compute **434**, **436** the codewords and associated error value, and store **438**, **440** the error value and associated block type and codeword only if the error value is less than the previously stored error value. After the last block type is processed, the process outputs **446** a result **447** of the block type and codewords **390***a* having the minimum error.

In an alternative embodiment, the optimal analog curve may be computed **430** before searching the block type. That is, the process may compute **430** the optimal analog curve before proceeding with selecting **428** the block type, selecting **432** the partition, computing **434** the codewords, computing **436** the error, storing **438** the error, and storing **440** the block type and codeword. Computing **430** the optimal analog curve first is useful if all the, block types use the same analog curve and color space because the analog curve does not need to be recomputed for each block type.

FIG. **4**D further describes the process of identifying the optimal analog curve. The selection module **355** starts **448** the process by computing a center of gravity **450** for pixel **270** colors of an image block **260**. Computing **450** the center of gravity includes averaging the pixel **270** colors of the image block **260**. Once the center of gravity is computed **450**, the process identifies **452** a vector in color space to minimize the first moment of the pixel **270** colors of the image block **260**.

Specifically, for identifying **452** the vector the process fits a straight line to a set of data points, which are the original pixel **270** colors of the image block **260**. A straight line is chosen passing through the center of gravity of the set of points such that it minimizes the "moment of inertia" (the means square error). For example, for three pixel properties, to compute the direction of the line minimizing the moment of inertia, tensor inertia, T, is calculated from the individual colors as follows:

$$T = \sum \begin{matrix} C_{1i}^2 + C_{2i}^2 & -C_{0i}C_{1i} & -C_{0i}C_{2i} \\ -C_{0i}C_{1i} & C_{0i}^2 + C_{2i}^2 & -C_{1i}C_{2i} \\ -C_{0i}C_{2i} & -C_{2i}C_{1i} & C_{0i}^2 + C_{1i}^2 \end{matrix}$$

where C_0, C_1, and C_2 represent pixel properties, for example color components in RGB or YUV, relative to a center of gravity. In a preferred embodiment of an RGB color space, C_{0i} is the value of red, C_{1i} is the value of green, and C_{2i} is the value of blue for each pixel, i, of the image block. Further, i takes on integer values from 1 to W×H, so that if W=4 and H=4, i ranges from 1 to 16.

The eigenvector of tensor, T, with the smallest eigenvalue is calculated using conventional methods known to those skilled in the art. The eigenvector direction along with the calculated gravity center, defines the axis that minimizes the moment of inertia. This axis is used as the optimal analog curve, which in a preferred embodiment is a straight line. Those of ordinary skill in the art will readily recognize that the term optimal analog curve is not limited solely to a straight line but may include a set of parameters, such as pixel values or colors, that minimizes the moment of inertia or means square error when fitted to the center of gravity of the pixel colors in the image block. The set of parameters may define any geometric element, such as but not limited to a curve, a plane, a trapezoid, or the like.

FIG. **4**E illustrates the process undertaken by the codeword generation module **360** for selecting **432** the partitions, computing **434**, **436** the codewords for the partitions and the associated error, and storing **438**, **440** the error value, block type, and codeword if the error value is less than a previously stored error value. The process starts **456** with the codeword generation module **360** projecting **458** the W×H color values onto the previously constructed optimal analog curve. The value of W×H is the size in number of pixels **270** of an image block **260**. In a preferred embodiment, where W and H are both 4 pixels, W×H is 16 pixels.

Once the colors are projected **458** onto the analog curve, the colors are ordered **460** sequentially along that analog curve based on the position of the color on the one-dimensional analog curve. After the colors are ordered **460**, the codeword generation module **360** searches **462** for optimal partitions. That is, the codeword generation module **360** takes the W×H colors (one color associated with each pixel) that are ordered **460** along the analog curve and partitions, or groups, them into a finite number of clusters with a predefined relative spacing. In a preferred embodiment, where W=4 and H=4, so that W×H is 16, the 16 colors are placed in three or four clusters (M=3 or 4).

In conducting the search **462** for the optimal partition, the color selection module **360** finds the best M clusters for the W×H points projected onto the optimal curve, so that the error associated with the selection is minimized. The best M clusters are determined by minimizing the mean square error with the constraint that the points associated with each cluster are spaced to conform to the predefined spacing.

In a preferred embodiment, for a block type of four equidistant colors, the error may be defined as a squared error along the analog curve, such as

$$E^2=\Sigma_{cluster0}(x_i-p_0)^2+\Sigma_{cluster1}(x_i-((2/3)p_0+((1/3)p_1))^2+\Sigma_{cluster2}(x_i-((1/3)p_0+(2/3)p_1))^2+\Sigma_{cluster3}(x_i-p_1)^2$$

where E is the error for the particular grouping or clustering, p_0 and p_1 are the coded colors, and x_i are the projected points on the optimal analog curve.

In instances where the block type indicates three equidistant colors, the error may be defined as a squared error along the analog curve, such as

$$E^2=\Sigma_{cluster}0(x_i-p_0)^2+\Sigma_{cluster1}(x_i-((1/2)p_0+(1/2)p_1))^2+\Sigma_{cluster2}(x_i-p_1)^2$$

where, again, E is the error for the particular grouping or clustering, p_0 and p_1 are the coded colors, and x_i are the projected points on the optimal analog curve.

After the resulting **447** optimal codewords **390***a* are identified, they are forwarded to the bitmap construction module **340**. The bitmap construction module **340** uses the codewords **390***a* to identify the M colors that may be specified or inferred from those codewords **390***a*. In a preferred embodiment, the bitmap construction module **340** uses the codewords **390***a*, e.g., CW**0**, CW**1**, to identify the three or four colors that may be specified or inferred from those codewords **390***a*.

The bitmap construction module **340** constructs a block bitmap **390***b* using the codewords **390***a* associated with the image block **260**. Colors in the image block **260** are mapped to the closest color associated with one of the quantized colors specified by, or inferred from, the codewords **390***a*.

The result is a color index, referenced as ID, per pixel in the block identifying the associated quantized color.

Information indicating the block type is implied by the codewords **390***a* and the bitmap **390***b*. In a preferred embodiment, the order of the codewords **390***a* CW**0**, CW**1**, indicate the block type. If a numerical value of CW**0** is greater than a numerical value of CW**1**, the image block is a four color block. Otherwise, the block is a three color plus transparency block.

As discussed above, in a preferred embodiment, there are two image block types. One image block type has four equidistant colors, while the other image block type has three equidistant colors with the fourth color index used to specify that a pixel is transparent. For both image block types the color index is two bits.

The output of the bitmap construction module **340** is an encoded image block **390** having the M codewords **390***a* plus the bitmap **390***b*. Each encoded image block **390** is received by the encoded image composer **319** that, in turn, orders the encoded image blocks **390** in a file. In a preferred embodiment, the encoded image blocks **390** are ordered from left to right and from top to bottom in the same order as the blocks were broken down by the block decomposer **315**. The ordered file having the encoded image blocks **390** is concatenated with the header information **385***a* that is derived from the header **380***a* of the original image **310** to generate the encoded image data **385** that is the image encoder system **220** output **320**. The image encoder system **220** output **320** may be forwarded to the memory **115**, the storage device **120**, or the output device **130**, for example.

The image encoder system **220** of the present invention advantageously reduces the effective data size of an image, for example, from 24-bits per pixel to 4-bits per pixel. Further, the present invention beneficially addresses transparency issues by allowing for codewords to be used with a transparency identifier.

FIG. **5**A is a block diagram of an image decoder system **230** in accordance with the present invention. The image decoder system **230** includes an encoded image decomposing unit **501**, a header converter **508**, one or more block decoders **505** (**505***a*–**505***m*, where m is any positive integer value representing the last block decoder), and an image composer **504**. The encoded image decomposer **501** is coupled to receive the encoded image data **385** that was output **320** from the image encoder system **220**. The encoded image decomposer **501** is coupled to the one or more block decoders **505***a*–**505***m*. The one or more block decoders **505***a*–**505***m* are coupled to the image composer **504** that, in turn, is coupled to the output 240.

The encoded image decomposer **501** receives the encoded image data **385** and decomposes, or breaks, it into its header **385***a* and the encoded image blocks **390-1**–**390**-R. The encoded image decomposer **501** reads the modified header **385***a* of the encoded image data **385** and forwards the modified header **385***a* to the header converter **508**. The encoded image decomposer **501** also decomposes the encoded image data **385** into the individual encoded image blocks **390-1**–**390**-R that are forwarded to the one or more block decoders **505***a*–**505***m*.

The header converter **508** converts the modified header **385***a* to an output header. Simultaneously, the encoded image blocks **390-1**–**390**-R are decompressed or decoded by the one or more block decoders **505***a*–**505***m*. It is noted that the each encoded image block **390** may be processed sequentially in one block decoder **505***a* or multiple encoded image blocks **390-1**–**390**-R may be processed in parallel with one block decoder **505***a*–**505***m* for each encoded image block **390-1**–**390**-R. Thus, multiple block decoders **505***a*–**505***m* allows for parallel processing that increases the processing performance and efficiency of the image decoder system **230**.

The image composer **504** receives each decoded image block from the one or more block decoders **505***a*–**505***m* and orders them in a file. Further, the image composer **504** receives the converted header from the header converter **508**. The converted header and the decoded image blocks are placed together to generate output **240** data representing the original image **310**.

FIG. **5**B is a block diagram of a first embodiment of a block decoder **505** in accordance with the present invention. Each block decoder **505***a*–**505***m* includes a block type detector **520**, one or more decoder units, e.g., **533***a***-1** to **533***a*-k (k is any integer value), and an output selector **523**. The block type detector **520** is coupled to the encoded image decomposer **501**, the output selector **523**, and each of the one or more decoder units, e.g., **533***a***-1**–**533***a*-k. Each of the decoder units, e.g., **533***a***-1**–**533***a*-k, is coupled to the output selector **523** that, in turn, is coupled to the image composer **504**.

The block type detector **520** receives the encoded image blocks **390** and determines the block type for each encoded image block **390**. Specifically, the block type detector **520** passes a selector signal to the output selector **523** that will be used to select an output corresponding to the block type detected. The block type is detected based on the codewords **390***a*. After the block type is determined, the encoded image blocks **390** are passed to each of the decoder units, e.g., **533***a***-1**–**533***a*-k The decoder units, e.g., **533***a***-1**–**533***a*-k, decompress or decode each encoded image block **390** to generate the colors for the particular encoded image block **390**. The decoder units, e.g., **533***a***-1**–**533***a*-k, may be c-channels wide (one channel for each color component (or pixel property) being encoded), where c is any integer value. Using the selector signal, the block type detector **520** enables the output selector **523** to output the color of the encoded image block **390** from one of the decoder units, e.g., **533***a***-1**–**533***a*-k that corresponds with the block type detected by the block type detector **520**. Alternatively, using the selector signal, the appropriate decoder unit **533** could be selected so the encoded block is processed through that decoder unit only.

FIG. **5**C is a block diagram of a second embodiment of a block decoder **505** in accordance with the present invention. In a second embodiment, the block decoder **505** includes a block type detector **520**, a first and a second decoder unit **530**, **540**, and the output selector **523**. The block type detector **520** is coupled to receive the encoded image blocks **390** and is coupled to the first and the second decoder units **530**, **540** and the output selector **523**.

The block type detector **520** receives the encoded image blocks **390** and determines, by comparing the codewords **390***a* of the encoded image block **390**, the block type for each encoded image block **390**. For example, in a preferred embodiment, the block type is four quantized colors or three quantized colors and a transparency. Once the block type is selected and a selector signal is forwarded to the output selector **523**, the encoded image blocks **390** are decoded by the first and the second decoder units **530**, **540**. The first and the second decoder units **530**, **540** decode the encoded image block **390** to produce the pixel colors of each image block. The output selector **523** is enabled by the block type detector **520** to output the colors from the decoder unit **530**, **540** that corresponds to the block type selected.

FIG. **5**D is a logic diagram illustrating one embodiment of a decoder unit through a red-channel of the that decoder unit

in accordance with the present invention. Specifically, the decoder unit is similar to the decoder units **530**, **540** illustrated in FIG. **5**C. Moreover, the functionality of each of those decoder units **530**, **540** is merged into the single logic diagram illustrated in FIG. **5**D. Further, those skilled in the art will understand that although described with respect to the red-channel of the decoder units **530**, **540** the remaining channels, e.g., the green-channel and the blue-channel, in each decoder unit **530**, **540** are similarly coupled and functionally equivalent.

The logic diagram illustrating the decoder units **530**, **540** is shown to include portions of the block type detector **520**, for example a comparator unit **522**. The comparator unit **522** works with a first 2×1 multiplexer **525***a* and a second 2×1 multiplexer **525***b*. The comparator unit **522** is coupled to the first and the second 2×1 multiplexers **525***a*, **525***b*. Both 2×1 multiplexers **525***a*, **525***b* are coupled to a 4×1 multiplexer **526** that serves to select the appropriate color to output.

The red-channel **544**, **546** of the first decoder unit **530** includes a first and a second red-channel line **551***a*, **551***b* and a first and a second red-color block **550***a*, **550***b*. Along the path of each red-color block **550***a*, **550***b* is a first full adder **552***a*, **552***b*, a second full adder **554***a*, **554***b*, and a CLA ("carry-look ahead") adder **556***a*, **556***b*. The first and the second red-channel lines **551***a*, **551***b* are coupled to the first and the second red-color blocks **550***a*, **550***b*, respectively. Each red-color block **550***a*, **550***b* is coupled to the first full adder **552***a*, **552***b* associated with that red-color block **550***a*, **550***b*. Each first full adder **552***a*, **552***b* is coupled to the respective second full adder **554***a*, **554***b*. Each second full adder **554***a*, **554***b* is coupled to the respective CLA adder **556***a*, **556***b*.

The second decoder unit **540** comprises the first and the second red-channel lines **551***a*, **551***b* and the respective first and second red-color blocks **550***a*, **550***b* and an adder **558**. The first and the second channel lines **551***a*, **551***b* are coupled to their respective red-color blocks **550***a*, **550***b* as described above. Each red-color block **550***a*, **550***b* is coupled to the adder **558**.

The CLA adder **556***a* from the path of the first red-color block **550***a* of the first decoder unit **530** is coupled to the first 2×1 multiplexer **525***a* and the CLA adder **556***b* from the path of the second red-color block **550***b* of the first decoder unit **530** is coupled to the second 2×1 multiplexer **525***b*. The adder **558** of the second decoder unit **540** is coupled to both the first and the second 2×1 multiplexers **525***a*, **525***b*.

The 4×1 multiplexer **526** is coupled to the first and the second red-channel lines **551***a*, **551***b*, as well as to the first and the second 2×1 multiplexers **525***a*, **525***b*. The 4×1 multiplexer **526** is also coupled to receive a transparency indicator signal that indicates whether or not a transparency (no color) is being sent. The 4×1 multiplexer **526** selects a color for output based on the value of the color index, referenced as the ID signal, that references the associated quantized color for an individual pixel of the encoded image block **390**.

FIG. **6**A is a flow diagram illustrating operation of the decoder system **230** in accordance with the present invention. For purposes of illustration only, the process for the decoder system **230** will be described with a single block encoder **505** having two decoding units, e.g., **530**, **540**. Those skilled in the art will recognize that the process is functionally equivalent for decoder systems having more than one block decoder **505** and more than one decoder units, e.g., **533***a*-**1**–**533***a*-k.

The process starts **600** with the encoded image decomposer **501** receiving **605** the encoded, or compressed, image data **385** from the encoder system **220**, for example, through the memory **115** or the storage device **120**. The encoded image decomposer **501** decomposes **610** the encoded image data **385** by forwarding the modified header **385***a* to the header converter **508**. In addition, the encoded image decomposer **501** also decomposes **610** the encoded image data **385** into the individual encoded image blocks **390**-**1**–**390**-R.

The header converter **508** converts **612** the header information to generate an output header that is forwarded to the image composer **504**. Simultaneously, the one or more block decoders **505***a*–**505***m* decodes **615** the pixel colors for each encoded image block **390**. It is again noted that each encoded image block **390** may be decoded **615** sequentially in one block decoder **505***a* or multiple encoded image blocks **390**-**1**–**390**-R may be PATENT decoded **615** in parallel in multiple block decoders **505***a*–**505***m*, as described above. The process for decoding the encoded image blocks **390** is further described in FIG. **6**B. Each decoded **615** image block is then composed **620** into a data file with the converted **612** header information by the image composer **504**. The image composer **504** generates the data file as an output **625** that represents the original image **310**.

FIG. **6**B is a flow diagram illustrating operation of the block encoder **505** in accordance with the present invention. Once the process is started **630**, each encoded image block **390** is received by the block decoder **505** and the block type for each encoded image block **390** is detected **640**. Specifically, for a preferred embodiment the first and the second codewords **390***a*, CW**0**, CW**1**, respectively, are received **635** by the block type detector **520** of the block decoder **505**. As discussed above, comparing the numerical values of CW**0** and CW**1** reveals the block type.

In addition, the first five bits of each codeword **390***a*, e.g., CW**0**, CW**1**, that represent the red-channel color are received by the red-channel **545** of each of the first and the second decoder units **530**, **540**, the second 6-bits of each codeword **390***a* CW**0**, CW**1** that represent the green-channel color are received by the green-channel of each of the first and the second decoder units **530**, **540**, and the last 5-bits of each codeword **390***a* CW**0**, CW**1** that represent the blue-channel color are received by the blue-channel of each of the first and the second decoder units **530**, **540**.

The block type detector **520** detects **640** the block type for an encoded image block **390**. Specifically, the comparator **522** compares the first and the second codewords **390***a*, CW**0**, CW **1**, and generates a flag signal to enable the first 2×1 multiplexers **525***a* or the second 2×1 multiplexers **525***b* which, in turn, selects **645** either the first decoding unit **530** or the second decoding unit **540**, respectively. The process then calculates **650** the quantized color levels for the decoder units **530**, **540**.

To calculate **650** the quantized color levels, the first decoding unit **530** calculates the four colors associated with the two codewords **390***a*, CW**0**, CW**1**, using the following relationship:

$$CW0=\text{first codeword}=\text{first color};$$

$$CW1=\text{second codeword}=\text{second color};$$

$$CW2=\text{third color}=(2/3)CW0+(1/3)CW1;$$

$$CW3=\text{fourth color}=(1/3)CW0+(2/3)CW1.$$

In one embodiment, the first decoder unit **530** may estimate the above equations for CW**2** and CW**3**, for example, as follows:

$$CW2=(5/8)CW0+(3/8)CW1;\text{ and}$$

$CW3=(3/8)CW0+(5/8)CW1$.

The red-color blocks **550***a*, **550***b* serve as a one-bit shift register to get (½)CW**0** or (½)CW**1** and each full adder **552***a*, **552***b*, **554***a*, **554***b* also serves to shift the signal left by 1-it. Thus, the signal from the first full adders **552***a*, **552***b* is (¼)CW**0** or (¼)CW**1**, respectively, because of a two-bit overall shift and the signal from the second full adders **554***a*, **554***b* is (⅛)CW**0** or (⅛)CW**1**, respectively, because of a three-bit overall shift. These values allow for the above approximations for the color signals.

The second decoder unit **540** calculates **650** three colors associated with the codewords **390***a*, CW**0**, CW**1**, and includes a fourth signal that indicates a transparency is being passed. The second decoder unit **540** calculates colors, for example, as:

$CW0$=first codeword=first color;

$CW1$=second codeword=second color;

$CW3$=third color=$(1/2)CW0+(1/2)CW1$; and

T=Transparency.

In one embodiment the second decoder unit **540** has no approximation because the signals received from the red-color blocks **550***a*, **550***b* is shifted left by one-bit so that the color is already calculated to (½)CW**0** and (½)CW**1**, respectively.

After the quantized color levels for the selected **645** decoder unit **530**, **540** have been calculated **650**, each bitmap value for each pixel is read **655** from the encoded image data block **385**. As each index is read **655** it is mapped **660** to one of the four calculated colors if the first decoder unit **530** is selected **645** or one of the three colors and transparency if the second decoder unit **540** is selected. The mapped **660** colors are selected by the 4×1 multiplexer **526** based on the value of the ID signal from the bitmap **390***b* of the encoded image block **390**. As stated previously, a similar process occurs for selection of colors in the green-channel and the blue-channel.

As the colors are output from the red-, green-, and blue-channels, the output is received by the image composer **504**. The image composer **504** orders the output from the block encoders **505** in the same order as the original image **310** was decomposed. The resulting **665** image that is output from the image decoder system **230** is the original image that is forwarded to an output source **240**, e.g., a computer screen, which displays that image.

The system and method of the present invention beneficially allows for random access to any desired image block **260** within an image, and any pixel **270** within an image block **260**. FIG. **7**A is a block diagram of a subsystem **700** that provides random access to a pixel **270** or an image block **260** in accordance with the present invention. PATENT The random access subsystem **700** includes a block address computation module **710**, a block fetching module **720**, and the one or more block decoders **505**. The block address computation module **710** is coupled to receive the header information **385***a* of the encoded image data **385**. The block address computation module **710** is also coupled to the block fetching module **720**. The block fetching module **720** is coupled to receive the encoded image block portion **390-1**-R of the encoded image data **385**. The block fetching module **720** is also coupled to the block decoders **505**.

FIG. **7**B is a flow diagram illustrating a process of random access to a pixel **270** or an image block **260** using the random access subsystem **700** in accordance with the present invention. When particular pixels **270** have been identified for decoding, the process starts **740** with the image decoder system **230** receiving the encoded image data **385**. The modified header **385***a* of the encoded image data **385** is forwarded to the block address computation module **710** and the encoded image block portion **390-1**-R of the encoded image data **385** is forwarded to the block fetching module **720**.

The block address computation module **710** reads the modified header **385***a* to compute **745** the address of the encoded image block portion **390-1**-R having the desired pixels **270**. The address computed **745** is dependent upon the pixel coordinates within an image. Using the computed **745** address, the block fetching module **720** identifies the encoded image block **390** of the encoded image block portion **390-1**-R that has the desired pixels **270**. Once the encoded image block **390** having the desired pixels **270** has been identified, only the identified encoded image block **390** is forwarded to the block decoders **505** for processing.

Similar to the process described above in FIG. **6**B, the block decoders **505** compute **755** the quantized color levels for the identified encoded image blocks **390** having the desired pixels. After the quantized color levels have been computed **755**, the color of the desired pixel is selected **760** and output **765** from the image decoder system **230**.

Random access to pixels **270** of an image block **260** advantageously allows for selective decoding of only needed portions or sections of an image. Random access also allows the image to be decoded in any order the data is required. For example, in three-dimensional texture mapping only portions of the texture may be required and these portions will generally be required in some non-sequential order. Thus, the present invention increases processing efficiency and performance when processing only a portion or section of an image.

The present invention beneficially encodes, or compresses, the size of an original image **310** from 24-bits per pixel to an aggregate 4-bits per pixel and then decodes, or decompresses the encoded image data **385** to get a representation of the original image **310**. Further, the claimed invention uses, for example, two base points or codewords from which additional colors are derived so that extra bits are not necessary to identify a pixel **270** color.

Moreover, the present invention advantageously accomplishes the data compression on an individual block basis with the same number of bits per block so that the compression rate can remain fixed. Further, because the blocks are of fixed size with a fixed number of pixels **270**, the present invention beneficially allows for random access to any particular pixel **270** in the block. The present invention provides for an efficient use of system resources because entire blocks of data are not retrieved and decoded to display data corresponding to only a few pixels **270**.

In addition, the use of a fixed-rate 64-bit data blocks in the present invention provides the advantage of having simplified header information that allows for faster processing of individual data blocks. Also, a 64-bit data block allows for data blocks to be processed rapidly, e.g., within one-clock cycle, as the need to wait until a full data string is assembled is eliminated. Further, the present invention also reduces the microchip space necessary for a decoder system because the decoder system only needs to decode each pixel to a set of colors determined by, e.g., the two codewords.

While particular embodiments and applications of the present invention have been illustrated and described, it is to be understood that the invention is not limited to the precise construction and components disclosed herein and that vari-

ous modifications, changes and variations which will be apparent to those skilled in the art may be made in the arrangement, operation and details of the method and apparatus of the present invention disclosed herein without departing from the spirit and scope of the invention as defined in the appended claims.

What is claimed is:

1. A data format for representing an original image block having a pixel color set, comprising:

a codeword portion for storing at least two codewords;

a bitmap portion for storing a set of indices, the bitmap portion constructed by a bitmap construction module utilizing the codeword portion associated with the bitmap portion; and

wherein said codewords define at least three colors that approximate the pixel color set, and said indices map the pixel color set to at least one of said at least three colors.

2. The data format of claim **1**, wherein said set of indices includes a predefined index.

3. The data format of claim **2**, wherein said predefined index is for mapping a transparency identifier associated with the original image block.

4. The data format of claim **2**, wherein said predefined index is for mapping an alpha value associated with the original image block.

5. The data format of claim **2**, wherein said predefined index is for mapping a color key value associated with the original image block.

6. The data format of claim **1**, wherein said codeword portion includes a first portion for storing a first codeword and a second portion for storing a second codeword; and wherein said first codeword and said second codeword are used to indicate a block type for the original image block.

7. The data format of claim **1**, wherein said codeword portion includes a first portion for storing a first codeword and a second portion for storing a second codeword; and wherein said at least three colors includes at least two computed colors if said first codeword is greater than said second codeword.

8. The data format of claim **1**, wherein said codeword portion includes a first portion for storing a first codeword and a second portion for storing a second codeword; and wherein said at least three colors includes at least one computed color and said set of indices includes a predefined index if said first codeword is less than said second codeword.

9. The data format of claim **1**, wherein said at least three colors are computed using a geometric element fitted to said pixel color set so that said geometric element has a minimal moment of inertia.

10. The data format of claim **1**, wherein said at least three colors includes one of said at least two codewords.

11. A data format for representing an original image block having a pixel color set, comprising:

a codeword portion for storing at least one codeword;

a bitmap portion for storing a set of indices, said set of indices includes an available index for representing a transparency identifier, the bitmap portion constructed by a bitmap construction module utilizing the codeword portion associated with the bitmap portion; and

wherein said codeword defines a set of colors that approximate the pixel color set, and said indices map the pixel color set to at least one color in said set of colors.

12. The data format of claim **11**, wherein said set of colors includes said at least one codeword and a computed color.

13. The data format of claim **11**, wherein said set of colors includes at least three colors.

14. A data format for representing an original image block having a pixel color set, comprising:

a codeword portion for storing at least one codeword;

a bitmap portion for storing a set of indices;

wherein said at least one codeword defines a set of colors that approximate the pixel color set, and said indices map the pixel color set to at least one color in said set of colors; and

wherein said set of colors are computed using a geometric element fitted to said pixel color set so that said geometric element has a minimal moment of inertia.

15. An encoded image data format for representing an original image partitioned into at least two image blocks, said image blocks each having a corresponding pixel color set, the data format comprising:

at least two encoded image block portions, one of said encoded image block portions having a codeword portion for storing at least two codewords, and a bitmap portion for storing a set of indices, the bitmap portion constructed by a bitmap construction module utilizing the codeword portion associated with the bitmap portion; and

wherein said at least two codewords define at least three colors that approximate the pixel color set of one of the original image blocks, and said indices map the pixel color set to at least one of said at least three colors.

16. The data format of claim **15**, further including a header portion.

17. The data format of claim **15**, wherein said set of indices includes a predefined index.

18. The data format of claim **17**, wherein said predefined index is for mapping a transparency identifier associated with the original image block.

19. The data format of claim **17**, wherein said predefined index is for mapping an alpha value associated with the original image block.

20. The data format of claim **17**, wherein said predefined index is for mapping a color key value associated with the original image block.

21. The data format of claim **15**, wherein said set of colors are computed using a geometric element fitted to said pixel color set so that said geometric element has a minimal moment of inertia.

22. The data format of claim **15**, wherein said at least three colors includes one of said at least two codewords.

23. An encoded image data format for representing an original image partitioned into at least a first image block having a first pixel color set and a second image block having a second pixel color set, the data format comprising:

a first encoded image block having a first portion for storing a first codeword, a second codeword, and a first bitmap portion for storing a first set of indices;

a second encoded image block having a second portion for storing a third codeword, a fourth codeword, and a second bitmap portion for storing a second set of indices;

wherein said first and second codewords define a first set of colors that approximate the first pixel color set, and said first set of indices map the first set of colors to the first pixel color set; and

wherein said third and fourth codewords define a second set of colors that approximate the second pixel color

set, and said second set of indices map the second set of colors to the second pixel color set.

24. The encoded image data format of claim **23**, wherein said first set of colors includes at least three colors.

25. The data format of claim **23**, wherein said first set of colors includes at least said first codeword.

26. The data format of claim **23**, wherein said first set of colors includes said second codeword.

27. The data format of claim **23**, wherein said second set of colors includes at least three colors.

28. The data format of claim **23**, wherein said second set of colors includes said third codeword.

29. The data format of claim **23**, wherein said second set of colors includes said fourth codeword.

* * * * *

US006683978C1

(12) EX PARTE REEXAMINATION CERTIFICATE (9188th)

United States Patent

Iourcha et al.

(10) Number: US 6,683,978 C1

(45) Certificate Issued: Aug. 14, 2012

(54) **FIXED-RATE BLOCK-BASED IMAGE COMPRESSION WITH INFERRED PIXEL VALUES**

(76) Inventors: **Konstantine I. Iourcha**, San Jose, CA (US); **Krishna S. Nayak**, Stanford, CA (US); **Zhou Hong**, San Jose, CA (US)

Reexamination Request:
No. 90/009,846, Nov. 5, 2010

Reexamination Certificate for:

Patent No.:	**6,683,978**
Issued:	**Jan. 27, 2004**
Appl. No.:	**09/442,114**
Filed:	**Nov. 17, 1999**

Related U.S. Application Data

(63) Continuation of application No. 09/351,930, filed on Jul. 12, 1999, now Pat. No. 6,658,146, which is a continuation of application No. 08/942,860, filed on Oct. 2, 1997, now Pat. No. 5,956,431.

(51) **Int. Cl.**
G06K 9/00 (2006.01)

(52) **U.S. Cl.** **382/166**; 382/232; 725/146

(58) **Field of Classification Search** ..... None
See application file for complete search history.

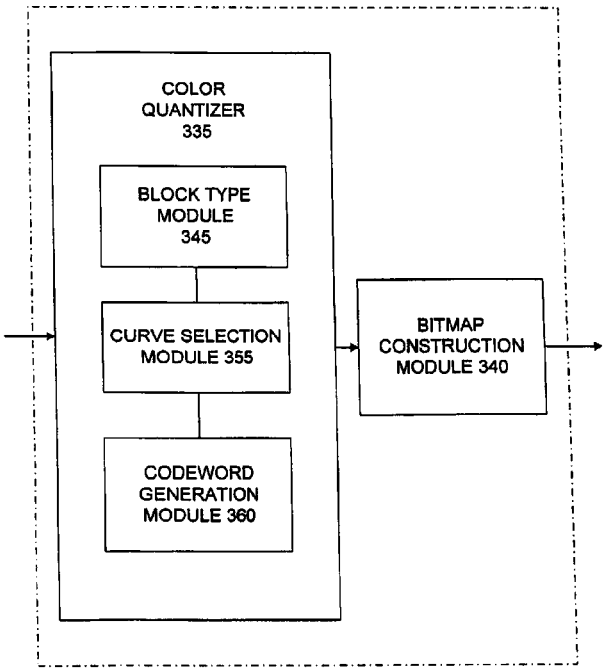
(56) **References Cited**

To view the complete listing of prior art documents cited during the proceeding for Reexamination Control Number 90/009,846, please refer to the USPTO's public Patent Application Information Retrieval (PAIR) system under the Display References tab.

Primary Examiner—Henry N Tran

(57) **ABSTRACT**

An image processing system includes an image encoder system and a image decoder system that are coupled together. The image encoder system includes a block decomposer and a block encoder that are coupled together. The block encoder includes a color quantizer and a bitmap construction module. The block decomposer breaks an original image into blocks. Each block is then processed by the block encoder. Specifically, the color quantizer selects some number of base points, or codewords, that serve as reference pixel values, such as colors, from which quantized pixel values are derived. The bitmap construction module then maps each pixel colors to one of the derived quantized colors. The codewords and bitmap are output as encoded image blocks. The decoder system includes a block decoder. The block decoder includes a block type detector, one or more decoder units, and an output selector. Using the codewords of the encoded data blocks, the comparator and the decorder units determine the quantized colors for the encoded image block and map each pixel to one of the quantized colors. The output selector outputs the appropriate color, which is ordered in an image composer with the other decoded blocks to output an image representative of the original image. A method for encoding an original image and for decoding the encoded image to generate a representation of the original image is also disclosed.



318

EX PARTE REEXAMINATION CERTIFICATE ISSUED UNDER 35 U.S.C. 307

THE PATENT IS HEREBY AMENDED AS INDICATED BELOW.

Matter enclosed in heavy brackets [ ] appeared in the patent, but has been deleted and is no longer a part of the patent; matter printed in italics indicates additions made to the patent.

AS A RESULT OF REEXAMINATION, IT HAS BEEN DETERMINED THAT:

The patentability of claims **11** and **14** is confirmed.

Claim **15** is cancelled.

Claim **16** is determined to be patentable as amended.

New claims **30-59** are added and determined to be patentable.

Claims **1-10**, **12**, **13** and **17-29** were not reexamined.

16. **[**The data format of claim **15**, further including**]** *An encoded image data format for representing an original image partitioned into at least two image blocks, said image blocks each having a corresponding pixel color set, the data format comprising:*

a header portion*;*

at least two encoded image block portions, one of said encoded image block portions having a codeword portion for storing at least two codewords, and a bitmap portion for storing a set of indices, said set of indices includes an available index for representing a transparency identifier, the bitmap portion constructed by a bitmap construction module utilizing the codeword portion associated with the bitmap portion; and

wherein said at least two codewords define at least three colors that approximate the pixel color set of one of the original image blocks, and said indices map the pixel color set to at least one of said at least three colors.

30. A data format produced by an image processor for representing an original image block having a pixel color set, comprising:

a codeword portion for storing at least one codeword;

a bitmap portion for storing a set of indices, said set of indices includes an available index for representing a transparency identifier, the bitmap portion constructed by a bitmap construction module utilizing the codeword portion associated with the bitmap portion; and

a block type portion encoded to specify one of a first block type or a second block type, each of the first block type and the second block type being 64 bits,

wherein said codeword defines a set of colors that approximate the pixel color set, and said indices map the pixel color set to at least one color in said set of colors.

31. The data format according to claim 30, wherein the first block type is encoded to specify an index value that indicates the transparency identifier.

32. The data format according to claim 30, wherein each index in the set of indices is encoded with at least two bits per pixel.

33. The data format according to claim 30, wherein the first block type indicates encoding of the bitmap portion with four colors and the second block type indicates encoding of the bitmap portion with three colors and the transparency identifier.

34. The data format according to claim 30, wherein the first block type indicates encoding of the bitmap portion with four equidistant colors and the second block type indicates encoding of the bitmap portion with three equidistant colors and the transparency identifier.

35. The data format according to claim 30, wherein the codeword portion includes at least two associated codewords, the order of which indicating a block type,

the order of the at least two associated codewords indicating either a four-color block type or a three-color plus transparency block type upon comparison of a numerical value of a first codeword from the at least two associated codewords and a numerical value of a second codeword from the at least two associated codewords.

36. The data format according to claim 30, wherein the bitmap portion corresponds to a 4×4 image block of image elements.

37. The data format according to claim 30, wherein the at least one codeword is encoded to specify a color in RGB format.

38. The data format according to claim 30, wherein the at least one codeword includes a 5-bit portion specifying a red color, a 6-bit portion specifying a green color, and a 5-bit portion specifying a blue color.

39. The data format according to claim 30, wherein the at least one codeword is encoded to specify a color in YUV format.

40. A data format produced by an image processor for representing an original image block having a pixel color set, comprising:

a codeword portion for storing at least one codeword;

a bitmap portion for storing a set of indices; and

a block type portion encoded to specify one of a first block type or a second block type, each of the first block type and the second block type being 64 bits,

wherein said at least one codeword defines a set of colors that approximate the pixel color set, and said indices map the pixel color set to at least one color in said set of colors, and

wherein said set of colors are computed using a geometric element fitted to said pixel color set so that said geometric element has a minimal moment of inertia.

41. The data format according to claim 40, wherein the first block type is encoded to specify an index value that indicates a transparent color.

42. The data format according to claim 40, wherein each index in the set of indices is encoded with at least two bits per pixel.

43. The data format according to claim 40, wherein the first block type indicates encoding of the bitmap portion with four colors and the second block type indicates encoding of the bitmap portion with three colors and a transparent color.

44. The data format according to claim 40, wherein the first block type indicates encoding of the bitmap portion with four equidistant colors and the second block type indicates encoding of the bitmap portion with three equidistant colors and a transparent color.

45. The data format according to claim 40, wherein the codeword portion includes at least two associated codewords, the order of which indicating a block type,

the order of the at least two associated codewords indicating either a four-color block type or a three-color plus

transparency block type upon comparison of a numerical value of a first codeword from the at least two associated codewords and a numerical value of a second codeword from the at least two associated codewords.

46. The data format according to claim 40, wherein the bitmap portion corresponds to a 4×4 image block of image elements.

47. The data format according to claim 40, wherein the at least one codeword is encoded to specify a color in RGB format.

48. The data format according to claim 40, wherein the at least one codeword is encoded to specify a 5-bit portion specifying a red color, a 6-bit portion specifying a green color, and a 5-bit portion specifying a blue color.

49. The data format according to claim 40, wherein the at least one codeword is encoded to specify a color in YUV format.

50. An encoded image data format produced by an image processor for representing an original image partitioned into at least two image blocks, said image blocks each having a corresponding pixel color set, the data format comprising:

at least two encoded image block portions, one of said encoded image block portions having a codeword portion for storing at least two associated codewords, and a bitmap portion for storing a set of indices, the bitmap portion constructed by a bitmap construction module utilizing the codeword portion associated with the bitmap portion; and

a block type portion encoded to specify one of a first block type or a second block type, each of the first block type and the second block type being 64 bits,

wherein said at least two associated codewords define at least three colors that approximate the pixel color set of one of the original image blocks, and said indices map the pixel color set to at least one of said at least three colors.

51. The data format according to claim 50, wherein the first block type is encoded to specify an index value that indicates a transparent color.

52. The data format according to claim 50, wherein each index in the set of indices is encoded with at least two bits per pixel.

53. The data format according to claim 50, wherein the first block type indicates encoding of the bitmap portion with four colors and the second block type indicates encoding of the bitmap portion with three colors and a transparent color.

54. The data format according to claim 50, wherein the first block type indicates encoding of the bitmap portion with four equidistant colors and the second block type indicates encoding of the bitmap portion with three equidistant colors and a transparent color.

55. The data format according to claim 50, wherein

the order of the at least two associated codewords indicating either a four-color block type or a three-color plus transparency block type upon comparison of a numerical value of a first codeword from the at least two associated codewords and a numerical value of a second codeword from the at least two codewords.

56. The data format according to claim 50, wherein the bitmap portion corresponds to a 4×4 image block of image elements.

57. The data format according to claim 50, wherein each codeword from the at least two associated codewords is encoded to specify a color in RGB format.

58. The data format according to claim 50, wherein each codeword from the at least two associated codewords includes a 5-bit portion specifying a red color, a 6-bit portion specifying a green color, and a 5-bit portion specifying a blue color.

59. The data format according to claim 50, wherein each codeword from the at least two associated codewords is encoded to specify a color in YUV format.

* * * * *