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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 January 2025 (16.01.2025)

WIPO | PCT

(10) International Publication Number
WO 2025/014615 A1

(51) International Patent Classification:
G06T 15/04 (2011.01) *H04N 19/20* (2014.01)
H04N 19/103 (2014.01) *H04N 19/42* (2014.01)
H04N 19/119 (2014.01) *G06T 9/00* (2006.01)
H04N 19/176 (2014.01)

(21) International Application Number:
PCT/US2024/033672

(22) International Filing Date:
12 June 2024 (12.06.2024)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
18/348,657 07 July 2023 (07.07.2023) US

(71) Applicant: **SONY INTERACTIVE ENTERTAINMENT LLC** [US/US]; 2207 Bridgepointe Parkway, San Mateo, California 94404 (US).

(72) Inventor: **MADAMS, Thomas**; 2207 Bridgepointe Parkway, San Mateo, California 94404 (US).

(74) Agent: **ROGITZ, John L.**; 2635 Camino Del Rio South, Suite 211, San Diego, California 92108 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE,

(54) Title: OPTIMIZED PARTITION SELECTION FOR BC7 TEXTURE ENCODING

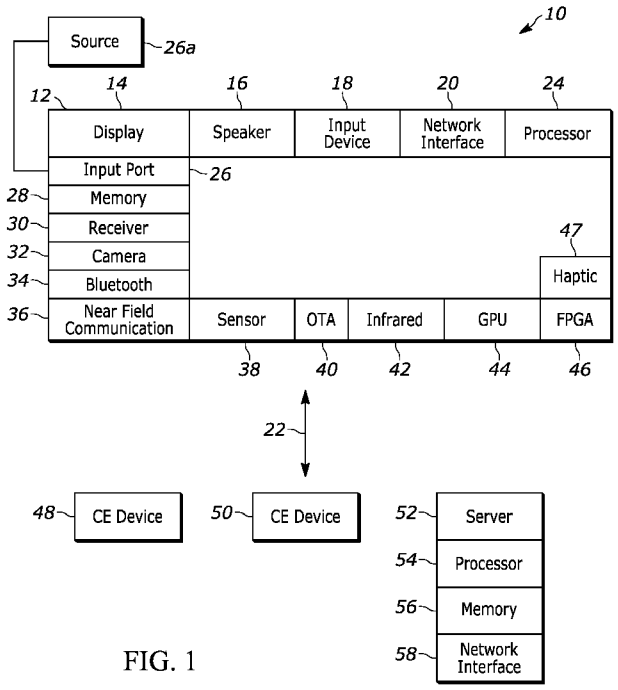

FIG. 1

(57) Abstract: Techniques are described for first generating (310) a short list of candidate partitions for BC-7 texture compression using calculations of gradient strengths in multiple directions and then selecting one of the candidate partitions using pixel extents for further processing (708) of the block.

[Continued on next page]

WO 2025/014615 A1

SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

OPTIMIZED PARTITION SELECTION FOR BC7 TEXTURE ENCODING

FIELD

The present application relates to technically inventive, non-routine solutions that are necessarily rooted in computer technology and that produce concrete technical improvements, and more specifically to optimized partition selection for BC7 texture encoding.

BACKGROUND

In computer simulations such as computer gaming, objects are rendered in part using "texture" data that describes the surfaces of the objects. The more texture data for a given object, the higher resolution the rendering can be. However, for bandwidth purposes it is desirable not to send large texture data structures to a rendering device.

SUMMARY

As understood herein, to conserve memory, texture data is usually compressed into one of a variety of block compression (BCn) modes that are natively sample-able on GPUs. BC6 and BC7 are of particular relevance to present principles.

Accordingly, an apparatus includes at least one processor assembly configured to identify M candidate partitions from N partitions, N>M, for block compression 7 (BC7) of texture data. This may be done at least in part by, for at least a first block of the texture data, computing gradient strengths between adjacent pixels in plural directions, with each direction being associated with a respective absolute gradient strength, identifying the direction having the lowest absolute gradient strength, and identifying the M candidate partitions based on the

direction having the lowest absolute gradient strength. The processor assembly is configured to use at least a first one of the M candidate partitions to process at least the first block of texture data for storage and/or transmission thereof for decompression at a rendering device to render computer simulation images.

In some implementations, the processor assembly may be configured for not using alpha values of pixels on blocks of the pixel data with alpha value not being less than 255.

In example embodiments, the plural directions can include horizontal, vertical, and two diagonal directions. In some embodiments, the processor assembly can be configured to apply a first scale factor to gradient strengths in the vertical and horizontal directions and a second scale factor to gradient strengths in the two diagonal dimensions to render scaled gradient strengths, and to use the gradient strengths to identify M candidate partitions.

In example implementations, the processor assembly may be configured to select the first one of the M candidate partitions to compress at least the first block of texture data at least in part by scoring each of the M partitions. This may be done by summing per-channel extents of pixels in each subset of the respective partition to render channel sums, weighting each channel sum by the number of pixels in each respective subset to establish the respective score of the respective partition, and selecting from among the M candidate partitions the partition with a score less than the scores of the remaining candidate partitions among the M candidate partitions.

If desired, the processor assembly can be configured to reuse at least a first calculation from a first candidate partition in executing at least one operation of a second candidate partition. For example, the first calculation can be a min/max calculation, and the processor

assembly can be configured to merge at least two min/max calculations to generate min/max bounds for plural subsets of the first candidate partition, and reuse the min/max bounds to calculate at least one gradient strength for the second candidate partition.

In non-limiting examples the processor assembly can be further configured to select a BC7 mode of compression. This may be done at least in part by, for each block in a training set of blocks of pixels, computing an error resulting in compressing the block in each of plural BC7 modes, and for each block in the training set, computing at least one feature associated with a per-channel range of pixel values within the block. The features and errors can be input to at least one machine learning (ML) model to train the model to predict per-mode compression error based on the features. The ML model is subsequently used to select a BC7 mode of compression for the texture data.

In another aspect, a device includes at least one computer storage that is not a transitory signal and that in turn includes instructions executable by at least one processor assembly to identify, for at least first and second directions relative to a computer graphics texture data structure, respective first and second values derived from pixel values of the computer graphics texture data structure. The instructions are executable to use one of the values to select a subset of M candidate partitions from N partitions, M<N. Furthermore, the instructions are executable to select a first one of the candidate partitions in the subset of M candidate partitions to encode the computer graphics texture data structure. The selection of the first one of the candidate partitions is based at least in part on pixel extents in the first one of the candidate partitions in the subset of M candidate partitions.

In another aspect, a method includes identifying, from among N possible partitions for a block compression 7 (BC7) block of texture data, M candidate partitions, M<N, based at least in part on gradient calculations in at least two directions relative to the block. The method then identifies a first one of the M candidate partitions based at least in part on extents of pixels in the block. The method includes using the first one of the M candidate partitions to process the block.

The details of the present disclosure, both as to its structure and operation, can be best understood in reference to the accompanying drawings, in which like reference numerals refer to like parts, and in which:

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of an example system including an example in consistent with present principles;

Figure 2 illustrates an example block compression (BC) system;

Figure 3 illustrates example logic in example flow chart format for selecting a small group of candidate partitions from the sixty four (64) possible partitions for the cases shown in Figures 3A and 3B;

Figure 3A illustrates sixty four (64) possible partitions for a four-by-four BC7 block divided into two subsets of texels, also referred to herein as “pixels”;

Figure 3B illustrates sixty four (64) possible partitions for a four-by-four BC7 block divided into two subsets of texels, also referred to herein as “pixels”;

Figures 4-6 graphically illustrate computation of gradient strengths in four directions consistent with the logic of Figure 3;

Figure 7 illustrates example logic in example flow chart format for selecting one of the small group of candidate partitions as the partition to be implemented;

Figure 8 illustrates example pre-processing logic in example flow chart format for optimizing the logic of Figure 7;

Figures 9-21 graphically illustrate principles of the logic of Figure 8;

Figure 22 illustrates example model training logic in example flow chart format for supporting block compression mode selection; and

Figure 23 illustrates example logic in example flow chart format for implementing BC mode selection.

DETAILED DESCRIPTION

This disclosure relates generally to computer ecosystems including aspects of consumer electronics (CE) device networks such as but not limited to computer game networks. A system herein may include server and client components which may be connected over a network such that data may be exchanged between the client and server components. The client components may include one or more computing devices including game consoles such as Sony PlayStation® or a game console made by Microsoft or Nintendo or other manufacturer, extended reality (XR) headsets such as virtual reality (VR) headsets, augmented reality (AR) headsets, portable televisions (e.g., smart TVs, Internet-enabled TVs), portable computers such as laptops and tablet computers, and other mobile devices including smart phones and

additional examples discussed below. These client devices may operate with a variety of operating environments. For example, some of the client computers may employ, as examples, Linux operating systems, operating systems from Microsoft, or a Unix operating system, or operating systems produced by Apple, Inc., or Google, or a Berkeley Software Distribution or Berkeley Standard Distribution (BSD) OS including descendants of BSD. These operating environments may be used to execute one or more browsing programs, such as a browser made by Microsoft or Google or Mozilla or other browser program that can access websites hosted by the Internet servers discussed below. Also, an operating environment according to present principles may be used to execute one or more computer game programs.

Servers and/or gateways may be used that may include one or more processors executing instructions that configure the servers to receive and transmit data over a network such as the Internet. Or a client and server can be connected over a local intranet or a virtual private network. A server or controller may be instantiated by a game console such as a Sony PlayStation®, a personal computer, etc.

Information may be exchanged over a network between the clients and servers. To this end and for security, servers and/or clients can include firewalls, load balancers, temporary storages, and proxies, and other network infrastructure for reliability and security. One or more servers may form an apparatus that implement methods of providing a secure community such as an online social website or gamer network to network members.

A processor may be a single- or multi-chip processor that can execute logic by means of various lines such as address lines, data lines, and control lines and registers and shift

registers. A processor including a digital signal processor (DSP) may be an embodiment of circuitry. A processor assembly may include one or more processors.

Components included in one embodiment can be used in other embodiments in any appropriate combination. For example, any of the various components described herein and/or depicted in the Figures may be combined, interchanged, or excluded from other embodiments.

"A system having at least one of A, B, and C" (likewise "a system having at least one of A, B, or C" and "a system having at least one of A, B, C") includes systems that have A alone, B alone, C alone, A and B together, A and C together, B and C together, and/or A, B, and C together.

Referring now to Figure 1, an example system 10 is shown, which may include one or more of the example devices mentioned above and described further below in accordance with present principles. The first of the example devices included in the system 10 is a consumer electronics (CE) device such as an audio video device (AVD) 12 such as but not limited to a theater display system which may be projector-based, or an Internet-enabled TV with a TV tuner (equivalently, set top box controlling a TV). The AVD 12 alternatively may also be a computerized Internet enabled ("smart") telephone, a tablet computer, a notebook computer, a head-mounted device (HMD) and/or headset such as smart glasses or a VR headset, another wearable computerized device, a computerized Internet-enabled music player, computerized Internet-enabled headphones, a computerized Internet-enabled implantable device such as an implantable skin device, etc. Regardless, it is to be understood that the AVD 12 is configured to undertake present principles (e.g., communicate with other CE devices to undertake present

principles, execute the logic described herein, and perform any other functions and/or operations described herein).

Accordingly, to undertake such principles the AVD 12 can be established by some, or all of the components shown. For example, the AVD 12 can include one or more touch-enabled displays 14 that may be implemented by a high definition or ultra-high definition "4K" or higher flat screen. The touch-enabled display(s) 14 may include, for example, a capacitive or resistive touch sensing layer with a grid of electrodes for touch sensing consistent with present principles.

The AVD 12 may also include one or more speakers 16 for outputting audio in accordance with present principles, and at least one additional input device 18 such as an audio receiver/microphone for entering audible commands to the AVD 12 to control the AVD 12. The example AVD 12 may also include one or more network interfaces 20 for communication over at least one network 22 such as the Internet, an WAN, an LAN, etc. under control of one or more processors 24. Thus, the interface 20 may be, without limitation, a Wi-Fi transceiver, which is an example of a wireless computer network interface, such as but not limited to a mesh network transceiver. It is to be understood that the processor 24 controls the AVD 12 to undertake present principles, including the other elements of the AVD 12 described herein such as controlling the display 14 to present images thereon and receiving input therefrom. Furthermore, note the network interface 20 may be a wired or wireless modem or router, or other appropriate interface such as a wireless telephony transceiver, or Wi-Fi transceiver as mentioned above, etc.

In addition to the foregoing, the AVD 12 may also include one or more input and/or output ports 26 such as a high-definition multimedia interface (HDMI) port or a universal serial bus (USB) port to physically connect to another CE device and/or a headphone port to connect headphones to the AVD 12 for presentation of audio from the AVD 12 to a user through the headphones. For example, the input port 26 may be connected via wire or wirelessly to a cable or satellite source 26a of audio video content. Thus, the source 26a may be a separate or integrated set top box, or a satellite receiver. Or the source 26a may be a game console or disk player containing content. The source 26a when implemented as a game console may include some or all of the components described below in relation to the CE device 48.

The AVD 12 may further include one or more computer memories/computer-readable storage media 28 such as disk-based or solid-state storage that are not transitory signals, in some cases embodied in the chassis of the AVD as standalone devices or as a personal video recording device (PVR) or video disk player either internal or external to the chassis of the AVD for playing back AV programs or as removable memory media or the below-described server. Also, in some embodiments, the AVD 12 can include a position or location receiver such as but not limited to a cellphone receiver, GPS receiver and/or altimeter 30 that is configured to receive geographic position information from a satellite or cellphone base station and provide the information to the processor 24 and/or determine an altitude at which the AVD 12 is disposed in conjunction with the processor 24.

Continuing the description of the AVD 12, in some embodiments the AVD 12 may include one or more cameras 32 that may be a thermal imaging camera, a digital camera such as a webcam, an IR sensor, an event-based sensor, and/or a camera integrated into the AVD 12

and controllable by the processor 24 to gather pictures/images and/or video in accordance with present principles. Also included on the AVD 12 may be a Bluetooth® transceiver 34 and other Near Field Communication (NFC) element 36 for communication with other devices using Bluetooth and/or NFC technology, respectively. An example NFC element can be a radio frequency identification (RFID) element.

Further still, the AVD 12 may include one or more auxiliary sensors 38 that provide input to the processor 24. For example, one or more of the auxiliary sensors 38 may include one or more pressure sensors forming a layer of the touch-enabled display 14 itself and may be, without limitation, piezoelectric pressure sensors, capacitive pressure sensors, piezoresistive strain gauges, optical pressure sensors, electromagnetic pressure sensors, etc. Other sensor examples include a pressure sensor, a motion sensor such as an accelerometer, gyroscope, cyclometer, or a magnetic sensor, an infrared (IR) sensor, an optical sensor, a speed and/or cadence sensor, an event-based sensor, a gesture sensor (e.g., for sensing gesture command). The sensor 38 thus may be implemented by one or more motion sensors, such as individual accelerometers, gyroscopes, and magnetometers and/or an inertial measurement unit (IMU) that typically includes a combination of accelerometers, gyroscopes, and magnetometers to determine the location and orientation of the AVD 12 in three dimension or by an event-based sensors such as event detection sensors (EDS). An EDS consistent with the present disclosure provides an output that indicates a change in light intensity sensed by at least one pixel of a light sensing array. For example, if the light sensed by a pixel is decreasing, the output of the EDS may be -1; if it is increasing, the output of the EDS may be a +1. No change in light intensity below a certain threshold may be indicated by an output binary signal of 0.

The AVD 12 may also include an over-the-air TV broadcast port 40 for receiving OTA TV broadcasts providing input to the processor 24. In addition to the foregoing, it is noted that the AVD 12 may also include an infrared (IR) transmitter and/or IR receiver and/or IR transceiver 42 such as an IR data association (IRDA) device. A battery (not shown) may be provided for powering the AVD 12, as may be a kinetic energy harvester that may turn kinetic energy into power to charge the battery and/or power the AVD 12. A graphics processing unit (GPU) 44 and field programmable gated array 46 also may be included. One or more haptics/vibration generators 47 may be provided for generating tactile signals that can be sensed by a person holding or in contact with the device. The haptics generators 47 may thus vibrate all or part of the AVD 12 using an electric motor connected to an off-center and/or off-balanced weight via the motor's rotatable shaft so that the shaft may rotate under control of the motor (which in turn may be controlled by a processor such as the processor 24) to create vibration of various frequencies and/or amplitudes as well as force simulations in various directions.

A light source such as a projector such as an infrared (IR) projector also may be included.

In addition to the AVD 12, the system 10 may include one or more other CE device types. In one example, a first CE device 48 may be a computer game console that can be used to send computer game audio and video to the AVD 12 via commands sent directly to the AVD 12 and/or through the below-described server while a second CE device 50 may include similar components as the first CE device 48. In the example shown, the second CE device 50 may be configured as a computer game controller manipulated by a player or a head-mounted display (HMD) worn by a player. The HMD may include a heads-up transparent or non-transparent

display for respectively presenting AR/MR content or VR content (more generally, extended reality (XR) content). The HMD may be configured as a glasses-type display or as a bulkier VR-type display vended by computer game equipment manufacturers.

In the example shown, only two CE devices are shown, it being understood that fewer or greater devices may be used. A device herein may implement some or all of the components shown for the AVD 12. Any of the components shown in the following figures may incorporate some or all of the components shown in the case of the AVD 12.

Now in reference to the afore-mentioned at least one server 52, it includes at least one server processor 54, at least one tangible computer readable storage medium 56 such as disk-based or solid-state storage, and at least one network interface 58 that, under control of the server processor 54, allows for communication with the other illustrated devices over the network 22, and indeed may facilitate communication between servers and client devices in accordance with present principles. Note that the network interface 58 may be, e.g., a wired or wireless modem or router, Wi-Fi transceiver, or other appropriate interface such as, e.g., a wireless telephony transceiver.

Accordingly, in some embodiments the server 52 may be an Internet server or an entire server "farm" and may include and perform "cloud" functions such that the devices of the system 10 may access a "cloud" environment via the server 52 in example embodiments for, e.g., network gaming applications. Or the server 52 may be implemented by one or more game consoles or other computers in the same room as the other devices shown or nearby.

The components shown in the following figures may include some or all components shown in herein. Any user interfaces (UI) described herein may be consolidated and/or expanded, and UI elements may be mixed and matched between UIs.

Present principles may employ various machine learning models, including deep learning models. Machine learning models consistent with present principles may use various algorithms trained in ways that include supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, feature learning, self-learning, and other forms of learning. Examples of such algorithms, which can be implemented by computer circuitry, include one or more neural networks, such as a convolutional neural network (CNN), a recurrent neural network (RNN), and a type of RNN known as a long short-term memory (LSTM) network. Generative pre-trained transformers (GPTT) also may be used. Support vector machines (SVM) and Bayesian networks also may be considered to be examples of machine learning models. In addition to the types of networks set forth above, models herein may be implemented by classifiers.

As understood herein, performing machine learning may therefore involve accessing and then training a model on training data to enable the model to process further data to make inferences. An artificial neural network/artificial intelligence model trained through machine learning may thus include an input layer, an output layer, and multiple hidden layers in between that that are configured and weighted to make inferences about an appropriate output.

Prior to turning to Figure 2, "textures" are data structures that can be mapped onto images to characterize the surfaces of the rendered objects. The basic data element of a texture data structure is a texture element or texel (combination of texture and pixel). Textures are

represented by arrays of texels representing the texture space. The texels are mapped to pixels in an image to be rendered to define the rendered surface of the image.

Various types of compression may be used on textures. One type is block compression, sometimes expressed as BCn compression that is a lossy texture compression which can be decompressed in-place by graphics processing units (GPUs). Block compression does not require the whole image to be decompressed, so the GPU can decompress the data structure while sampling the texture as though it was not compressed at all.

Block compression techniques compress 4x4 blocks of pixels into a single (smaller) data packet. Generally, this involves selecting two or more (depending on the BC compression type) "endpoint" colors with some information per-pixel about how to blend between those two colors at each pixel. The endpoint colors are shared for the entire 4x4 pixel block. For instance, for an image of only red, blue, and purple pixels, the compressor would likely choose one end point to be red, and the other blue. The purple pixels would have values that blend the two together.

The different BC types mostly differ in how many texture channels they have (BC4 for instance is one channel grayscale, "black and white"). BC6 and BC7 are special because they introduce the concept of modes that decide the interpretation of each block. With BC6/7 different modes allocate their bits differently on a per-block basis which allows the encoder/compressor to make different quality trade-offs in different regions of a texture.

With specific regard to BC7 and consistent with the above, textures are subdivided into fixed size 4x4 blocks, and each block is compressed to a fixed number of bits (e.g., BC7 uses 128 bits per block). Ignoring partitions for now, pixels in a block are represented by a single

pair of endpoint colors, shared between all pixels in the block and a 16 per-pixel interpolation index values, which define how much to blend between the two endpoint colors. A pixel's color in the compressed block is calculated by blending between the two endpoint colors by the amount specified by the pixel's interpolation index.

A single pair of endpoint colors can compress a block with low error if all pixels in a block are well-approximated by a blend of those two colors. On the other hand, if a block contains more than two very different colors, it is impossible to define two endpoint colors for which this approximation holds. Accordingly, to address this problem, several modes in BC7 partition a 4x4 block into two or three subsets, and each subset has its own pair of endpoint colors. Multi-subset modes necessarily have lower precision endpoints and interpolation indices because they must fit extra endpoint colors in the same 128 bits as block modes that do not use partitions. A block's partition must be one of sixty four (64) predetermined patterns that are fixed and defined in the BC7 specification. Selecting the "best" partition of the sixty four currently requires an essentially exhaustive test/search process. Techniques described herein provide an efficient way to select an effective partition.

Additionally and apart from the issue above of selecting a best partition, BC7 supports eight different compression modes, each of which makes its own respective trade-off between endpoint color precision and interpolation index precision (among other things). The mode used to encode each block is signaled in the first few bits of the encoded data. Generally, modes with higher precision endpoint colors have lower precision interpolation indices, and vice-versa. Depending on the mode used to compress a block, interpolation indices will be either 2-bit, 3-bit or 4-bit per pixel. Current techniques for selecting the best compression mode for BC7

textures is to conduct an exhaustive search, in which every mode is tested and the one selected that minimizes compression error. Techniques herein describe an efficient way to select a compression mode for a BC7 texture.

Accordingly, turn now to Figure 2, which illustrates a texture source 200 that sends textures to an encoder 202 such as a BC6 or BC7 encoder. While discussion below uses BC7 as an example, present principles apply to other compression modes such as BC6.

The encoder 202, which in a hardware implementation includes a processor assembly configured according to principles herein, processes the textures according to principles herein and stores compresses textures in one or more storages 204 and/or sends the compressed textures via a communication path 206 such as a local data bus or wired/wireless network link to a texture renderer 208, which typically includes one or more processors such as GPUs with memories to render images in accordance with image data and texture data on a display.

Note that to maximize efficiency of a hardware implementation of the encoder, data should be entirely 8 bit or 16 bit data.

Refer now to Figure 3. In example embodiments the logic may start at state 300 by a shortcut or sorts in which for each BC7 texture block to be compressed, it may be identified at decision diamond 302 whether the block is fully opaque (i.e., the alpha value for all 16 pixels in the block is 255). If so, all operations discussed below may be skipped for alpha values in this block as indicated at state 304.

The block is then processed at state 306 by computing, for each channel of each pixel, the total absolute gradient strength in four directions, namely, horizontal, vertical, and two

diagonal dimensions of the block. While ensuing graphical figures illustrate this, some further text description is in order here.

An RGBA pixel has four channels, in this example, a red channel, a green channel, a blue channel, and an alpha channel (which may be skipped as indicated above for opaque blocks). To compute a total gradient strength, for each pixel and each channel, the numerical value of that pixel for that channel is subtracted from the numerical value of an immediately adjacent pixel for that channel, and the absolute value of the remainder represents the gradient strength. Thus, for each pair of immediately adjacent pixels, four absolute gradient strengths are computed and summed to render the total absolute gradient strength for that pair. This is done for each direction, and then the total absolute gradient strengths for each pair of pixels in each given direction are summed to produce a total absolute gradient strength of the block in each of the four directions.

When done in the horizontal direction, the pixels being compared are immediately adjacent to each other in the same row of the texture block with no intervening pixels in that row between the pixels being compared. When done in the vertical direction, the pixels being compared are in the same column of the texture block with no intervening pixels in that column between the pixels being compared. When done in one of the diagonal configurations, the pixels being compared are in the same diagonal of the texture block with no intervening pixels in that diagonal between the pixels being compared. The two diagonals are offset 45 degrees from the vertical and horizontal directions, with one diagonal proceeding right and down relative to the block (termed “backward” herein) and the other diagonal proceeding right and up relative to the block (termed “forward” herein).

The logic moves from state 306 to state 308 to scale the four total absolute gradients strengths of the block under test. This is because a 4 pixel-by-4 pixel block has twelve pairs of immediately pixels in each of the vertical and horizontal directions but only nine pairs of immediately pixels in each of the diagonal directions. Accordingly, the scaling at state 308 can be implemented in some embodiments by multiplying the vertical and horizontal strengths by nine and the diagonal strengths by twelve.

Moving to state 310, the smallest absolute gradient strength after scaling is selected from among the four, i.e., from among the horizontal, vertical, and two diagonal gradient strengths. This smallest absolute gradient strength is used to select a small predefined list of candidate partitions from among the 64 partitions illustrated in Figures 3A and 3B. The predefined small lists may be generated by a human expert or by a machine learning (ML) model to select plural of the 64 possible partitions to correspond to each of the four directions by noting, for example, that some of the 64 possible partitions exhibit better compression qualities with smaller losses than others for a given direction.

Turn now to Figures 4-6 for graphical illustration of the discussion above. Figure 4 illustrates computation of the total absolute gradient strength of a block in horizontal 400, vertical 402, and both 45° diagonal 404, 406 directions. As indicated above, gradient strength is computed by accumulating the absolute differences between each pair of immediately adjacent pixels indicated by the arrows, for each pixel channel.

Figure 5 illustrates details for computing the horizontal gradient strength, it being understood that computations other directions are similar. Given the pixel labeling in Figure 5,

the horizontal direction strength is computed with the following, where a_{ch} means the numerical value in the denoted channel in pixel a:

strength = 0

for ch in {R, G, B, A}:

strength += abs(a_{ch}-b_{ch}) + abs(b_{ch}-c_{ch}) + abs(c_{ch}-d_{ch})

strength += abs(e_{ch}-f_{ch}) + abs(f_{ch}-g_{ch}) + abs(g_{ch}-h_{ch})

strength += abs(i_{ch}-j_{ch}) + abs(j_{ch}-k_{ch}) + abs(k_{ch}-l_{ch})

strength += abs(m_{ch}-n_{ch}) + abs(n_{ch}-o_{ch}) + abs(o_{ch}-p_{ch})

Turn now to Figure 6, which illustrates, for the two-subset case of Figure 3A, four lists 600, 602, 604, 606 for the respective horizontal, vertical, and two diagonal direction partitions corresponding to the absolute gradient strength being smallest in the respective direction. One of the lists shown in Figure 6 is selected at state 310 in Figure 3, namely, the list corresponding to the direction with the smallest absolute gradient strength. The different respective shades of pixels indicate the pixels of each respective subset. As can be seen, the horizontal list 600 includes candidate partitions with one subset being primarily composed of row-aligned pixels, whereas the vertical list 602 includes partitions in which pixels of a subset are primarily vertically aligned, and so on.

Present techniques do not require a fixed set of partitions, only that the partitions in the list should roughly follow their (horizontal, vertical, diagonal) direction as shown in Figure 6. Example implementations are free to use larger or smaller lists to balance performance and quality. Once the list of partitions has been selected, the gradient strengths are no longer used in the remaining steps.

Which leads to Figures 7 and 8. Commencing at state 700, for each candidate partition in the list selected at block 310 of Figure 3, the logic moves to state 702 to sum the per-channel extents of pixels in each subset of pixels in the candidate partition under test. "Pixel extent" is determined by identifying the minimum and maximum values of a channel from all pixels in a subset, and subtracting the minimum from the maximum. Pixel extent thus is a measure of the range of colors in a block or subset. A block or subset with small extent will be more uniform in color than one with large extent.

The per-channel extents are weighted (e.g., by multiplication) at state 704 by the number of pixels in the subset to produce a score, which is output at state 706. After this is done for all candidate partitions in the list selected at state 310 in Figure 3, the partition with the minimum score is selected as the partition to implement at state 708.

Below is a naïve implementation that illustrates computing the score for a single partition. Figure 8, discussed below, provides even more efficiency over the naïve implementation that follows. In the code below, "min" and "max" refer to minimum and maximum selection functions, respectively, whereas "mn" and "mx" refer to minimum and maximum numbers in the set being tested:

```
uint16 calculatePartitionScore(const uint8 subsets[NumSubsets][16][NumChannels],
uint8 subsetSizes[NumSubsets]) {
    uint16 partitionScore = 0;
    for (uint8 subset = 0; subset < NumSubsets; ++subset) {
        uint8 mn[NumChannels], mx[NumChannels];
```

```
            for (uint8 c = 0; c < NumChannels; ++c) mn[c] = mx[c] =
subsets[subset][0][c];
            for (uint8 i = 1; i < subsetSizes[subset]; ++i) {
                for (uint8 c = 0; c < NumChannels; ++c) {
                    mn[c] = min(mn[c], subsets[subset][i][c]);
                    mx[c] = max(mx[c], subsets[subset][i][c]);
                }
            }
            for (uint8 c = 0; c < NumChannels; ++c) partitionScore +=
subsetSizes[subset] * (mx[subset][c] - mn[subset][c]);
        }
        return partitionScore;
    }
```

Where:

subsets[i][16][NumChannels] contains the pixels for subset i (not all elements in the array are set);

subsetSizes[i] contains the number of valid pixels in the subsets array;

The partition with minimum partitionScore is selected at state 708 in Figure 7.

Figure 8 illustrates how to significantly reduce the number of operations performed compared to the naïve implementation above by noticing that there is significant redundancy in the partition patterns. Taking the horizontal partitions of Figure 9 for example, one subset includes pixels that occupy the entire top row of each candidate partition, while that same subset

also includes pixels that occupy the entire second row for two of the candidate partitions. The results of the min/max operations computed for the top row of the first partition can be reused when computing the second partition and so on.

Thus, at state 800 of Figure 8, instead of recomputing the partition score from scratch each time, the min/max bounds for smaller groups of pixels (e.g., "N" pixels, wherein N is an integer greater than one) are computed and at state 802 merged together. State 804 indicates that additional merges are performed to compute min/max bounds for the final "M" subsets, wherein "M: is an integer.

Figures 10-21 illustrate the principles of Figure 8 further using row direction selection as an example. It is to be understood that Figures 10-21 are non-limiting examples of potential groups of pixels and a series of merge operations. The optimal choice of groups of pixels and the order in which those groups are merged depends on the list of choice of predefined candidate partitions.

In Figure 10, the min/max bounds are computed for the four groups of pixels 1000, 1002, 1004, and 1006, since the pixels in each of these rows are all in the same subset (albeit one subset may make up one row and the other subset may make up another row).

Figures 11 and 12 illustrate that the computation of Figure 10 immediately yields the min/max bounds for two of the subsets 1100, 1200 that are required to be processed in Figure 7. Figures 13-16 illustrate merging the min/max bounds, yielding min/max bounds for four more subsets 1300, 1400, 1500, 1600. Figures 17 and 18 then illustrate two more respective merges to compute the min/max bounds for the final two subsets 1700, 1800.

Figure 19 illustrates a manually designed dependency chain that near-minimizes the number of min/max operations required to compute subsets for the six example horizontal partitions. Each block shows the pixels for which the min and max have been computed. All min/max operations in a row can be performed without a dependency on any other in the same row. Rows above should be computed before rows below. In Figure 19, pixels with shading 1900 are temporary calculations, whereas pixels with shading 1902 indicate computations of final subsets.

On the other hand, Figure 20 illustrates a manually designed dependency chain that near-minimizes the number of min/max operations required to compute subsets for the six example forward diagonal partitions discussed previously. As was the case for the horizontal example of Figure 19, in Figure 20 each block shows the pixels for which the min and max have been computed, and all min/max operations in a row can be performed without a dependency on any other in the same row. Rows above should be computed before rows below. In Figure 20, pixels with shading 2000 are temporary calculations, whereas pixels with shading 2002 indicate computations of final subsets.

Figure 21 illustrates how the principles above extends to 3-subset partitions as shown in the case of Figure 3B.

Present principles provide not only a technique for efficiently selecting a partition for BC encoding, but also a technique for efficiently selecting a compression mode in the case of BC6 and BC7 that may be used in consonance with the partition selection described above or as a standalone technique.

To amplify on discussion above, BC7 compression, as an example, works by representing the pixels in a 4x4 block (or subset for multi-subset modes) as a pair of endpoint colors and per-pixel interpolation values between the two colors. The endpoint colors are shared by all pixels in the block or subset and as such are moderate-to-high precision, whereas the interpolation values are per-pixel and as such are low-to-moderate precision (a 2, 3, or 4-bit value). Several modes are similar but make different trade-offs between endpoint & interpolation precision:

Modes 0 and 2 have three subsets:

Mode 0: 4-bit RGB endpoint colors, 3-bit interpolation values

Mode 2: 5-bit RGB endpoint colors , 2-bit interpolation values

Modes 1 and 3 have two subsets:

Mode 1: 6-bit RGB endpoint colors, 3-bit interpolation values

Mode 3: 7-bit RGB endpoint colors, 2-bit interpolation values

Modes 4 and 5 are a single subset with have two sets of interpolation values (one for RGB and the other for A):

Mode 4: 5-bit RGB, 6-bit A endpoint colors, 2 and 3-bit interpolation values

Mode 5: 7-bit RGB, 8-bit A endpoint colors, 2 and 3-bit interpolation values

Whether the 3-bit indices in mode 4 are used for RGB or A is signaled by a control bit in the compressed block.

Disclosure below is directed to efficiently selecting which mode to compress a block in without, as is currently done, conducting an exhaustive search in which every mode is tested and the one selected that minimizes compression error. As recognized by present principles,

4x4 blocks of pixels that are near-uniform in color have endpoints that are close together; such blocks will generally benefit more from high-precision endpoints than high-precision interpolation values.

With this recognition in mind, turn now to Figure 22, which illustrates an offline preprocessing step in which a large dataset of 4x4 blocks of pixels is received at state 2200. Then at state 2202, for each block, the logic moves to state 2204 to compute the error that results from compressing the block in each BC7 mode (mode 4 can be compressed twice, once with 3-bits interpolation values assigned to RGB and once with them assigned to A).

Proceeding to state 2206, for each block a small number of features discussed further below as "rgbRangeMax", "rgbRangeSum", and "aRange" are computed based on the per-channel range of pixel values within the block or subsets (for multi-subset modes) as follows, in which "ch" again refers to channel:

$range_{ch} = max_{ch} - min_{ch}$

$rgbRangeMax = max(range_R, range_G, range_B)$

$rgbRangeSum = range_R + range_G + range_B$

$aRange = range_A$

It is to be noted that the three features above (rgbRangeMax, rgbRangeSum, and aRange) apply to BC7 modes that support alpha. For BC7 modes that do not support alpha, only the first two features are used (and the model need only be trained on these two features). Note further that for multi-subset modes, the relevant features should be computed for each subset separately, then the model evaluated for each subset, and the results of the subsets for a block summed to yield the score.

Moving to state 2208, a linear machine learning (ML) model is trained that predicts the per-mode compression error from these features.

Once the model is trained, it may be used in Figure 23. At state 2300 the above features are calculated for a block to be compressed and input to state 2302 to the trained model. The model outputs at state 2304 the compression mode predicted to have the lowest error given the features input at state 2302. This mode is used to compress the block.

In an example embodiment the model may output one of three pairs of similar modes, that is, modes (0, 2); modes (1, 3); and modes (4, 5).

Thus, the features used by the model to select the compression mode relate to a channel range that is the difference between the maximum and minimum pixel values for that channel in a block or subset of a block. The maximum of the three RGB channel ranges establishes the first one of the features. The second feature is the sum of the RGB channel ranges, while the third feature is the range of alpha values in the block or subset, in other words, the range of pixel transparency values in the block or subset.

An example linear predictor that chooses between mode 4, 3-bit RGB interpolation/2-bit A interpolation and mode 4, 2-bit RGB interpolation/3-bit A interpolation and mode 5 is given below.

```
uint8 weights[3][3] = {
    { 243, 156, 110 },
    { 103, 242, 193 },
    { 225, 248, 205 },
```

```
};
uint32 biases[3] = { 26 << 8, 26 << 8, 5 << 8 };
uint8 rgbRange[3] = { rgbaMax[0] - rgbaMin[0], rgbaMax[1] - rgbaMin[1], rgbaMax[2] - rgbaMin[2] };
uint8 rgbRangeMax = max(max(rgbRange[0], rgbRange[1]), rgbRange[2]);
uint16 rgbRangeSum = rgbRange[0] + rgbRange[1] + rgbRange[2]);
uint8 aRange = rgbaMax[3] - rgbaMin[3];

uint32 score40 = rgbRangeSum * weights[0][0] + rgbRangeMax * weights[0][1] + aRange * weights[0][2] + biases[0];
uint32 score41 = rgbRangeSum * weights[1][0] + rgbRangeMax * weights[1][1] + aRange * weights[1][2] + biases[1];
uint32 score5  = rgbRangeSum * weights[2][0] + rgbRangeMax * weights[2][1] + aRange * weights[2][2] + biases[2];

uint32 score4 = min(score40, score41);
bool use3BitRgbInterpolation = score4 == score41;
bool useMode4 = score4 < score5;
```

While particular techniques are herein shown and described in detail, it is to be understood that the subject matter which is encompassed by the present application is limited only by the claims.

WHAT IS CLAIMED IS:

1. An apparatus comprising:

at least one processor assembly configured to:

identify M candidate partitions from N partitions, N>M, for block compression 7 (BC7) of texture data at least in part by:

for at least a first block of the texture data, computing gradient strengths between adjacent pixels in plural directions, each direction being associated with a respective absolute gradient strength;

identifying the direction having the lowest absolute gradient strength;

identifying the M candidate partitions based on the direction having the lowest absolute gradient strength; and

use at least a first one of the M candidate partitions to process at least the first block of texture data for storage and/or transmission thereof for decompression at a rendering device to render computer simulation images.

2. The apparatus of Claim 1, wherein the processor is assembly is configured for not using alpha values of pixels on blocks of the pixel data with alpha value not being less than 255.

3. The apparatus of Claim 1, wherein the plural directions comprise horizontal and vertical.

4. The apparatus of Claim 3, wherein the plural directions comprise two diagonal directions.

5. The apparatus of Claim 4, wherein the processor assembly is configured to:

apply a first scale factor to gradient strengths in the vertical and horizontal directions and a second scale factor to gradient strengths in the two diagonal dimensions to render scaled gradient strengths; and

use the gradient strengths to identify M candidate partitions.

6. The apparatus of Claim 1, wherein the processor assembly is configured to select the first one of the M candidate partitions to compress at least the first block of texture data at least in part by scoring each of the M partitions by:

summing per-channel extents of pixels in each subset of the respective partition to render channel sums;

weighting each channel sum by the number of pixels in each respective subset to establish the respective score of the respective partition; and

selecting from among the M candidate partitions the partition with a score less than the scores of the remaining candidate partitions among the M candidate partitions.

7. The apparatus of Claim 1, wherein the processor assembly is configured to:

reuse at least a first calculation from a first candidate partition in calculating at least one operation of a second candidate partition.

8. The apparatus of Claim 7, wherein the first calculation comprises a min/max calculation.

9. The apparatus of Claim 8, wherein the processor assembly is configured to:

merge at least two min/max calculations to generate min/max bounds for plural subsets of the first candidate partition; and

reuse the min/max bounds to execute at least one operation for the second candidate partition.

10. The apparatus of Claim 1, wherein the processor assembly is configured to:

select a BC7 mode of compression at least in part by:

for each block in a training set of blocks of pixels, computing an error resulting in compressing the block or component subsets of the block in each of plural BC7 modes;

for each block in the training set, computing at least one feature associated with a per-channel range of pixel values within the block or the component subsets;

inputting the features and errors to at least one machine learning (ML) model to train the model to predict per-mode compression error based on the features; and

subsequently using the ML model to select a BC7 mode of compression for the texture data.

11. A device comprising:

at least one computer storage that is not a transitory signal and that comprises instructions executable by at least one processor assembly to:

identify, for at least first and second directions relative to a computer graphics texture data structure, respective first and second values derived from pixel values of the computer graphics texture data structure;

use one of the values to select a subset of M candidate partitions from N partitions, M<N; and

select a first one of the candidate partitions in the subset of M candidate partitions to encode the computer graphics texture data structure, selection of the first one of the candidate partitions being based at least in part on pixel extents in the first one of the candidate partitions in the subset of M candidate partitions.

12. The device of Claim 11, wherein the computer graphics data structure comprises at least one block.

13. The device of Claim 12, wherein the block comprises a block compression (BC) 7 block.

14. The device of Claim 11, wherein the first and second directions respectively comprise horizontal and vertical.

15. The device of Claim 14, wherein the instructions are executable to:

identify for the horizontal, vertical, and two diagonal directions respective first, second, third, and fourth values derived from pixel values of the block.

16. The device of Claim 15, wherein the first through fourth values respectively establish first through fourth gradient strengths, and the instructions are executable to:

apply a first scale factor to the respective gradient strengths in the vertical and horizontal directions and a second scale factor to the respective gradient strengths in the two diagonal directions to render scaled gradient strengths; and

use the gradient strengths to identify the M candidate partitions.

17. The device of Claim 11, wherein the instructions are executable to select the first one of the candidate partitions at least in part by:

for each of the M candidate partitions, summing per-channel extents of pixels in each subset of the respective partition to render channel sums;

weighting each channel sum by the number of pixels in each respective subset to establish the respective score of the respective partition; and

selecting from among the M candidate partitions the partition with a score less than the scores of the remaining candidate partitions among the M candidate partitions.

18. The device of Claim 11, wherein the instructions are executable to:

reuse at least a first calculation from a first candidate partition in at least one operation of a second candidate partition.

19. The device of Claim 18, wherein the first calculation comprises a min/max calculation, and the instructions are executable to:

merge at least two min/max calculations to generate min/max bounds for plural subsets of the first candidate partition; and

reuse the min/max bounds to calculate at least one operation for the second candidate partition.

20. A method comprising:

identifying, from among N possible partitions for a block compression 7 (BC7) block of texture data, M candidate partitions, M<N, based at least in part on gradient calculations in at least two directions relative to the block;

identifying a first one of the M candidate partitions based at least in part on extents of pixels in the block; and

using the first one of the M candidate partitions to process the block.

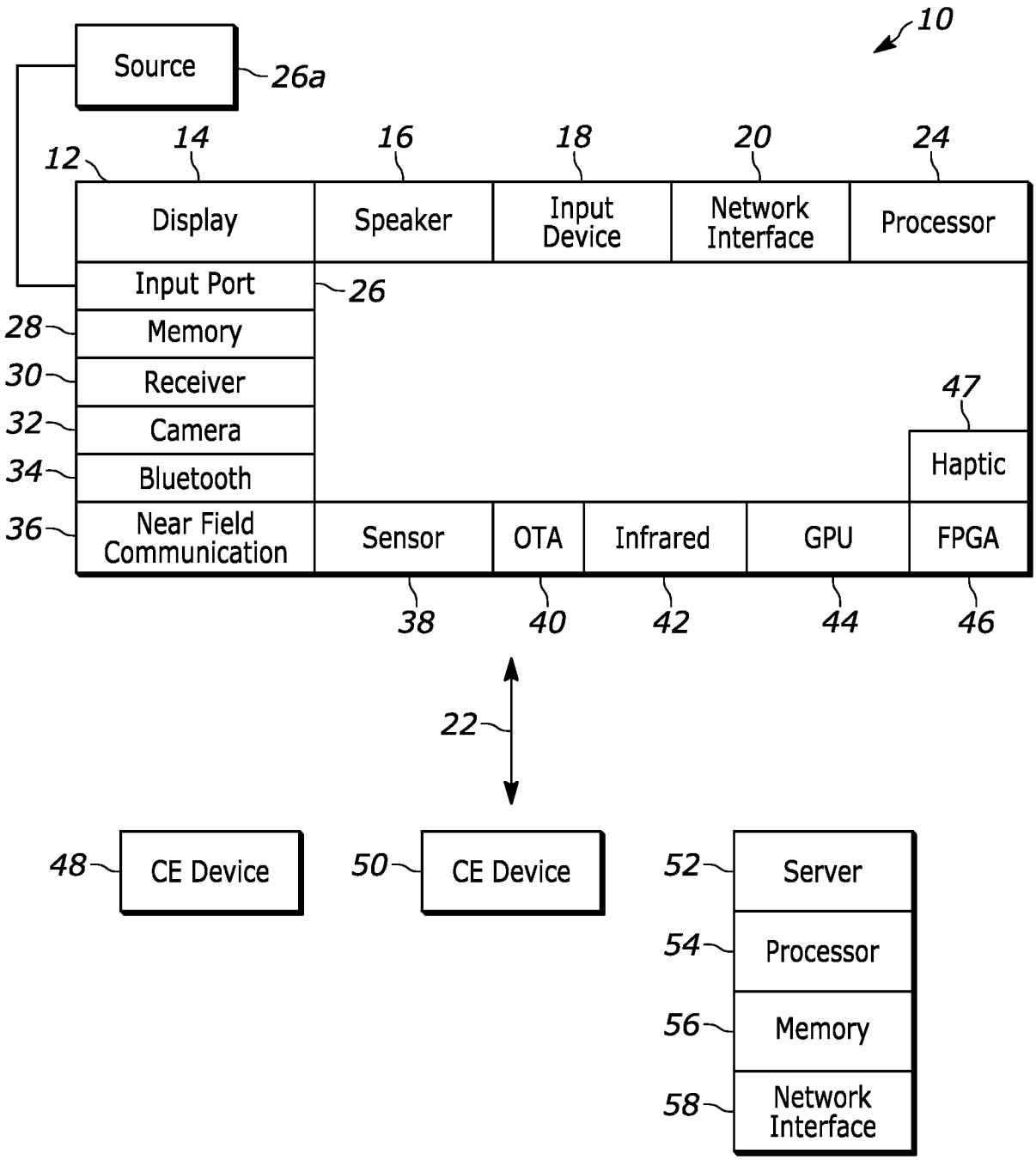
10
Source
26a
12
14
16
18
20
24
Display
Speaker
Input Device
Network Interface
Processor
Input Port
26
28
Memory
30
Receiver
47
32
Camera
Haptic
34
Bluetooth
36
Near Field Communication
Sensor
OTA
Infrared
GPU
FPGA
38
40
42
44
46
22
48
CE Device
50
CE Device
52
Server
54
Processor
56
Memory
58
Network Interface

FIG. 1

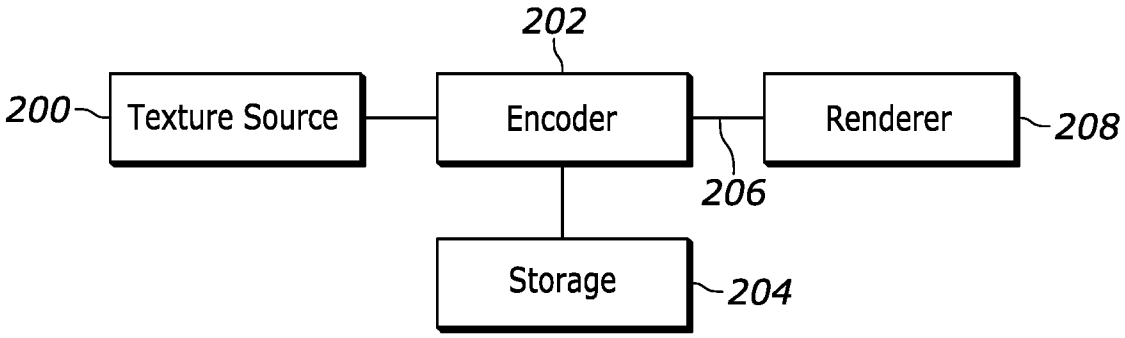
202
200
Texture Source
Encoder
Renderer
208
206
Storage
204

FIG. 2

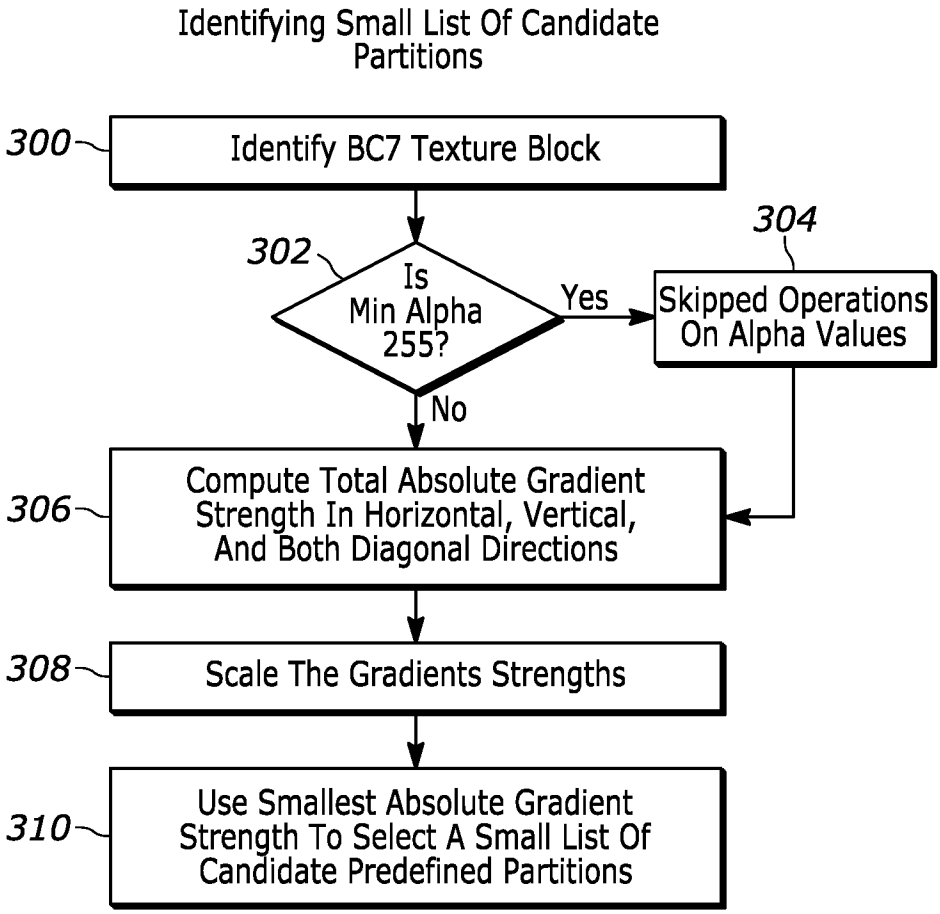
Identifying Small List Of Candidate Partitions
300
Identify BC7 Texture Block
302
Is Min Alpha 255?
Yes
304
Skipped Operations On Alpha Values
No
306
Compute Total Absolute Gradient Strength In Horizontal, Vertical, And Both Diagonal Directions
308
Scale The Gradients Strengths
310
Use Smallest Absolute Gradient Strength To Select A Small List Of Candidate Predefined Partitions

FIG. 3

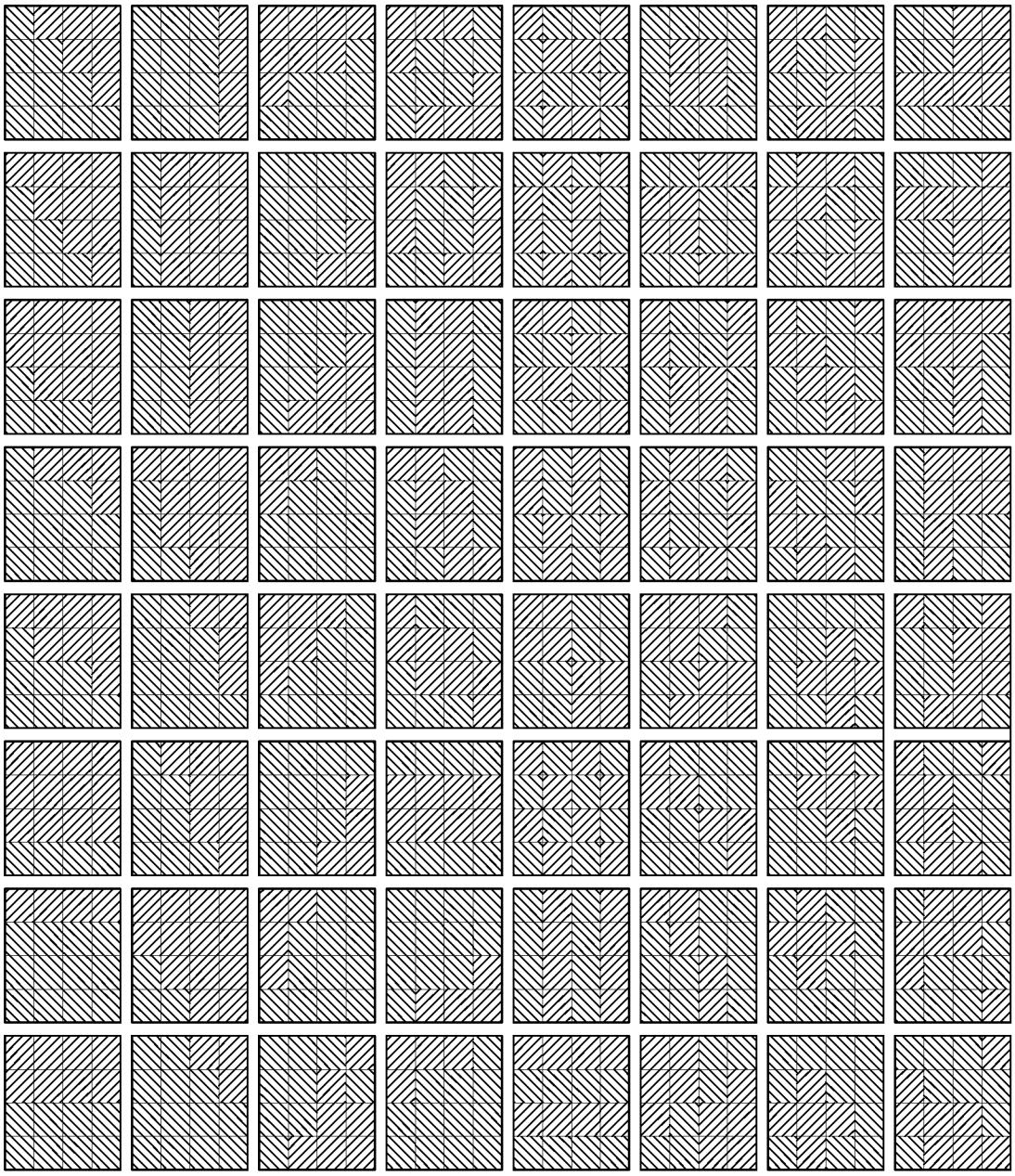

FIG. 3A

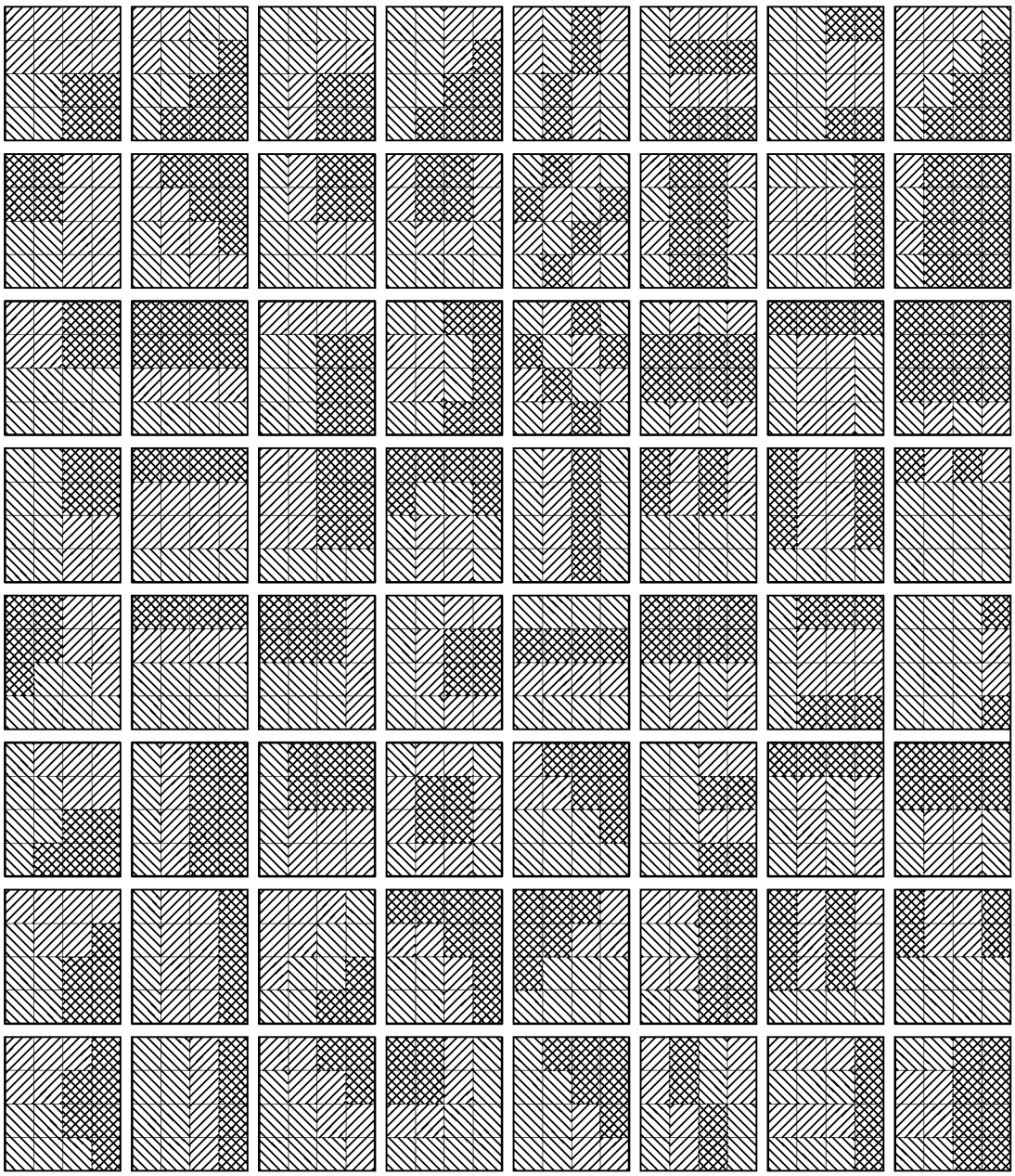

FIG. 3B

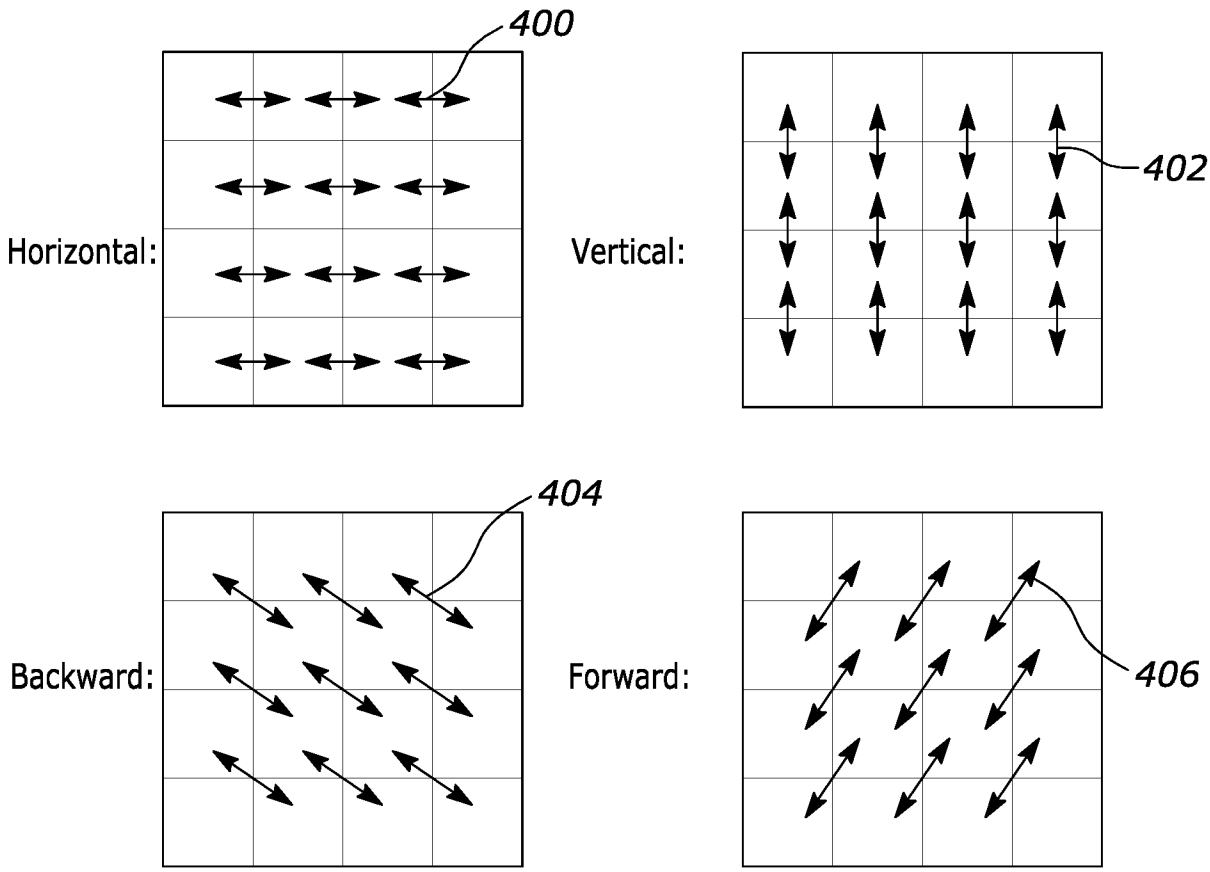
400
Horizontal:
Vertical:
402
404
Backward:
Forward:
406

FIG. 4

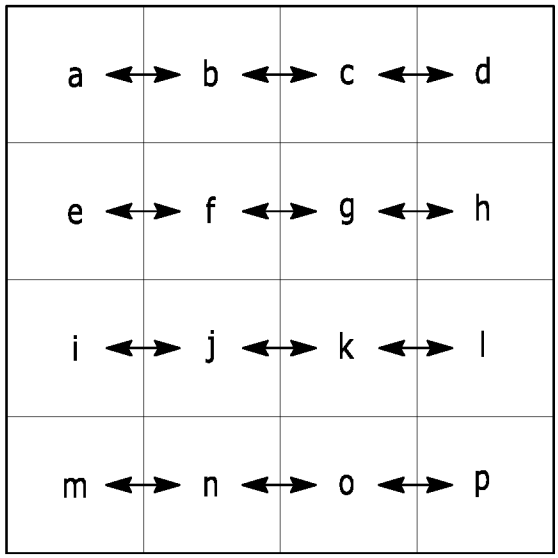
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p

FIG. 5

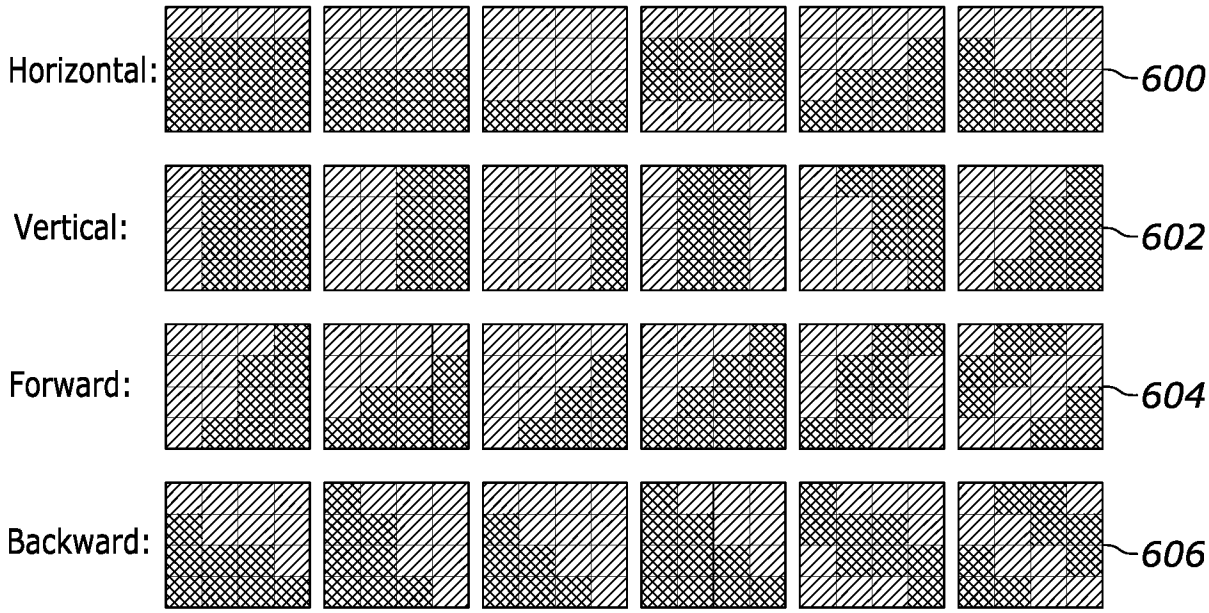

FIG. 6

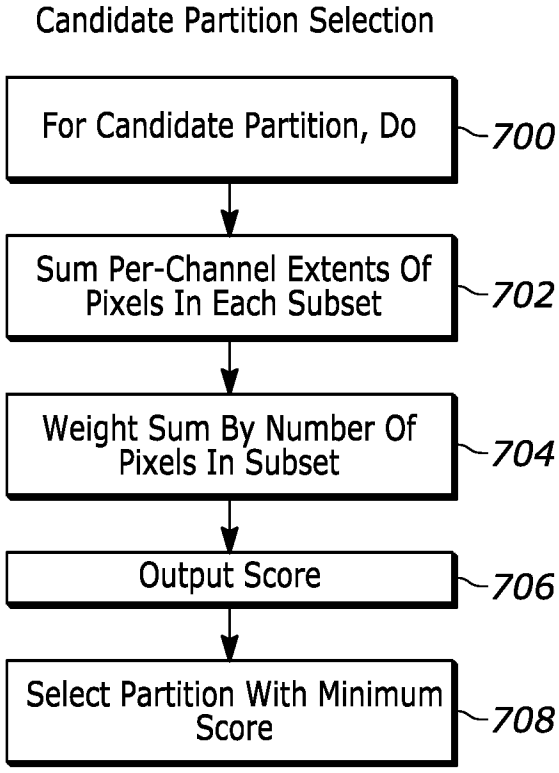
Candidate Partition Selection
For Candidate Partition, Do
700
Sum Per-Channel Extents Of Pixels In Each Subset
702
Weight Sum By Number Of Pixels In Subset
704
Output Score
706
Select Partition With Minimum Score
708

FIG. 7

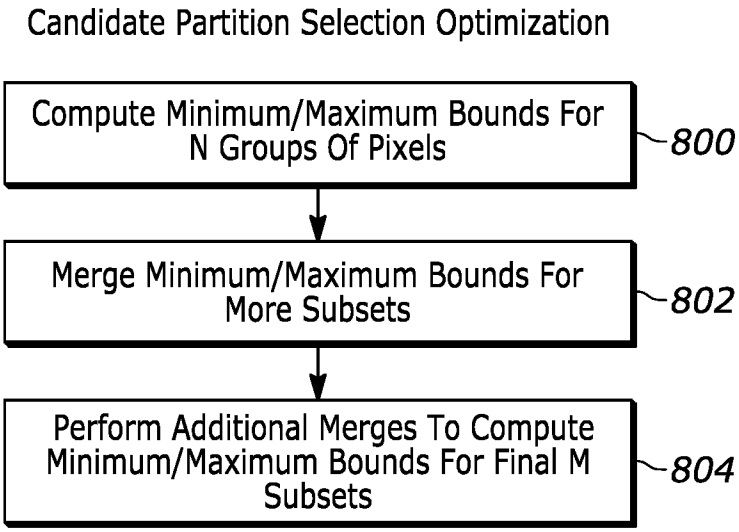
Candidate Partition Selection Optimization
Compute Minimum/Maximum Bounds For N Groups Of Pixels
800
Merge Minimum/Maximum Bounds For More Subsets
802
Perform Additional Merges To Compute Minimum/Maximum Bounds For Final M Subsets
804

FIG. 8

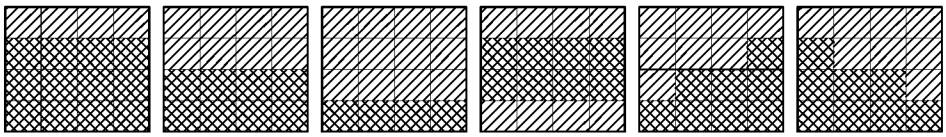

FIG. 9

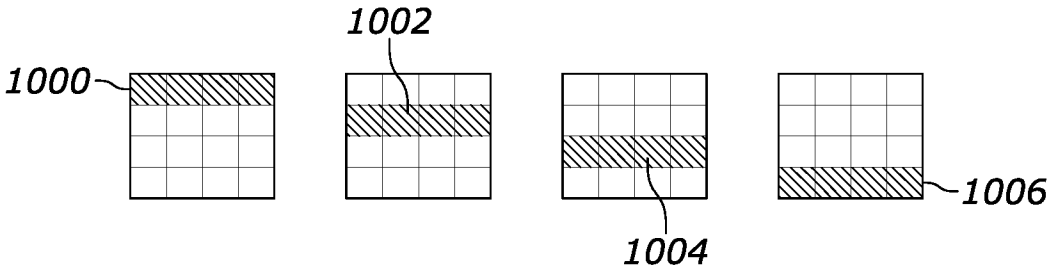

FIG. 10

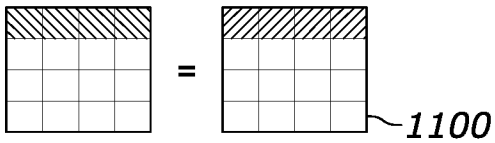

FIG. 11

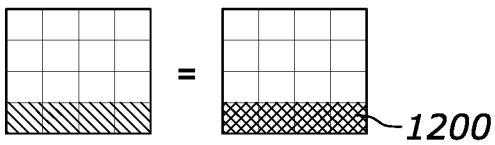

FIG. 12

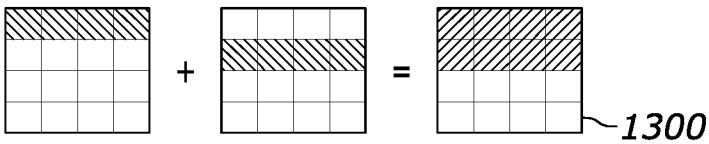

FIG. 13

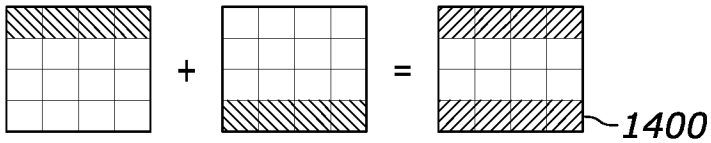

FIG. 14

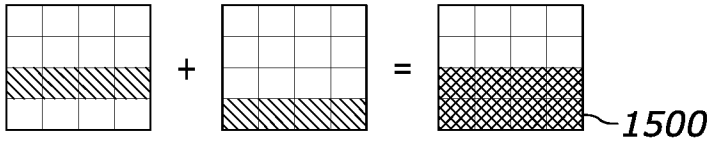

FIG. 15

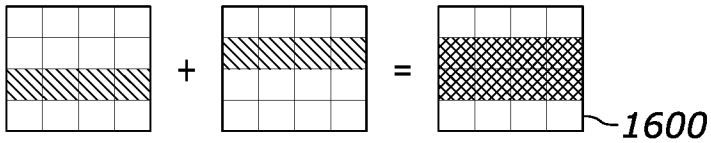

FIG. 16

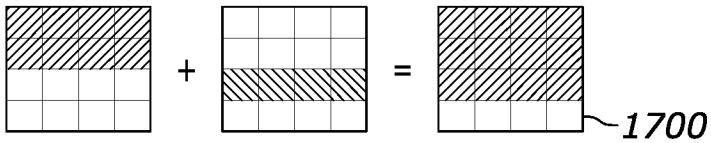

FIG. 17

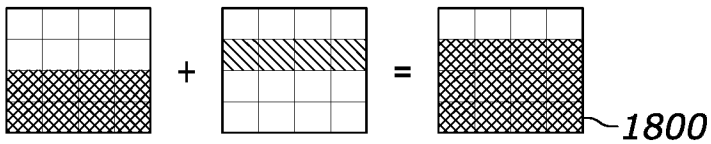

FIG. 18

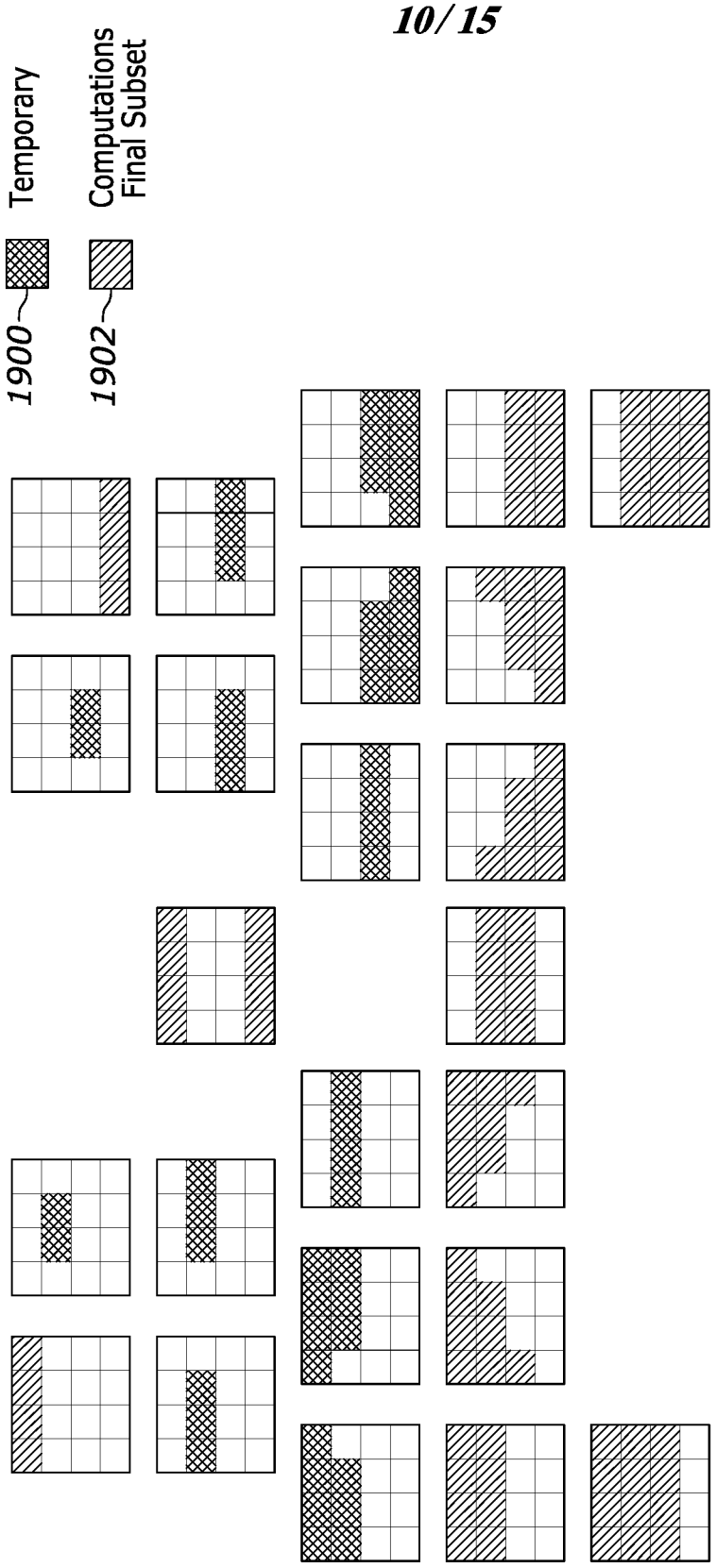

FIG. 19

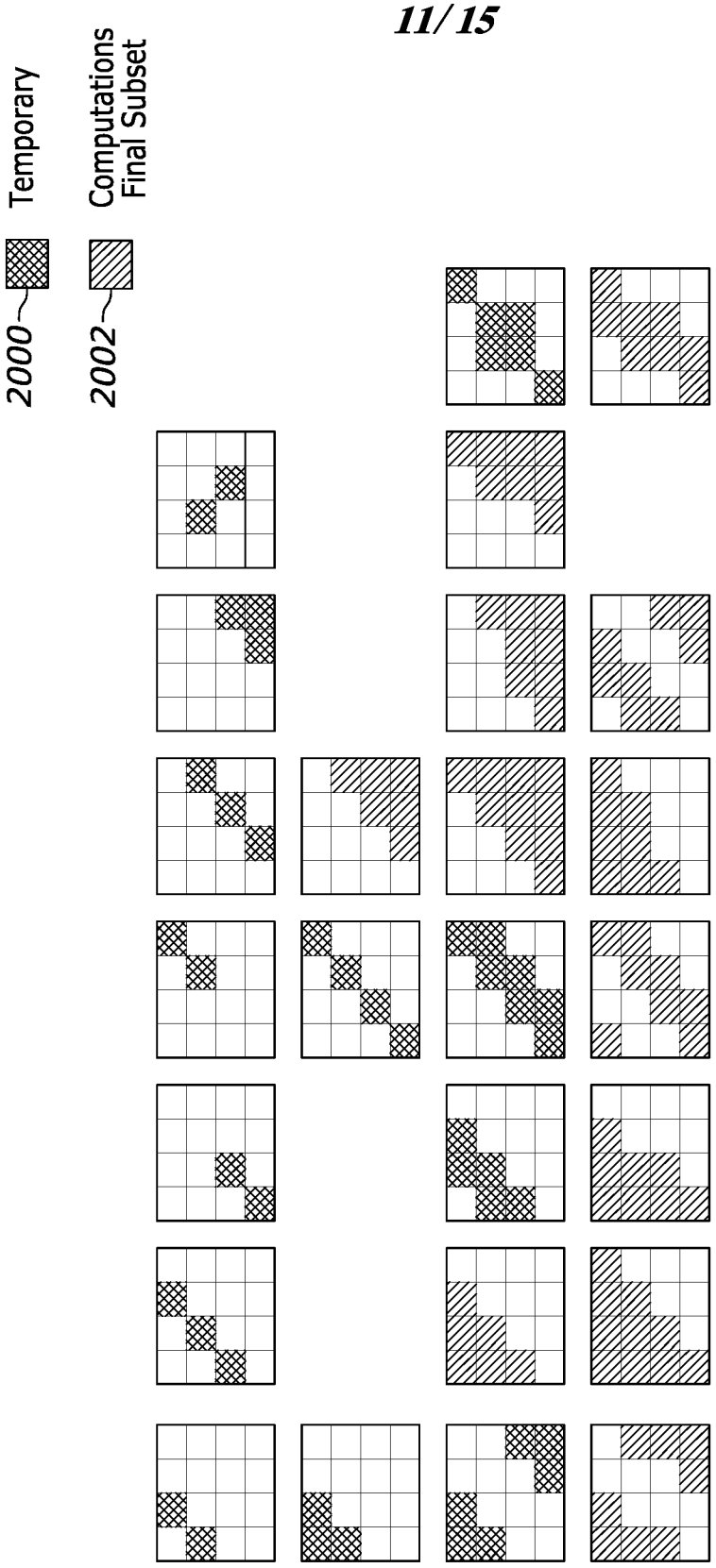

FIG. 20

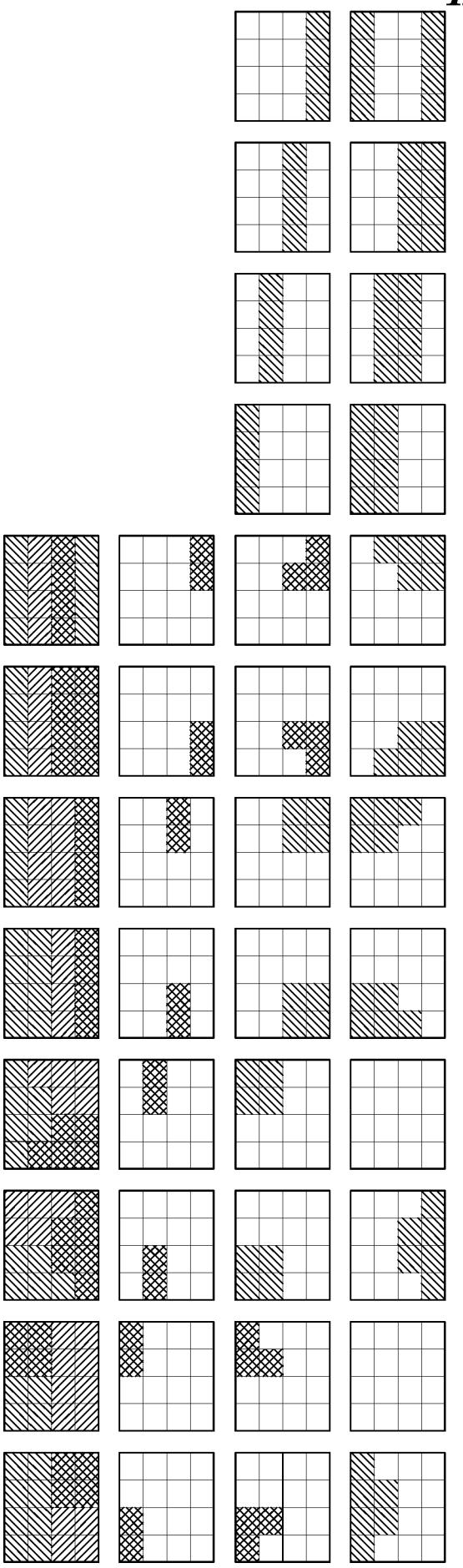

FIG. 21

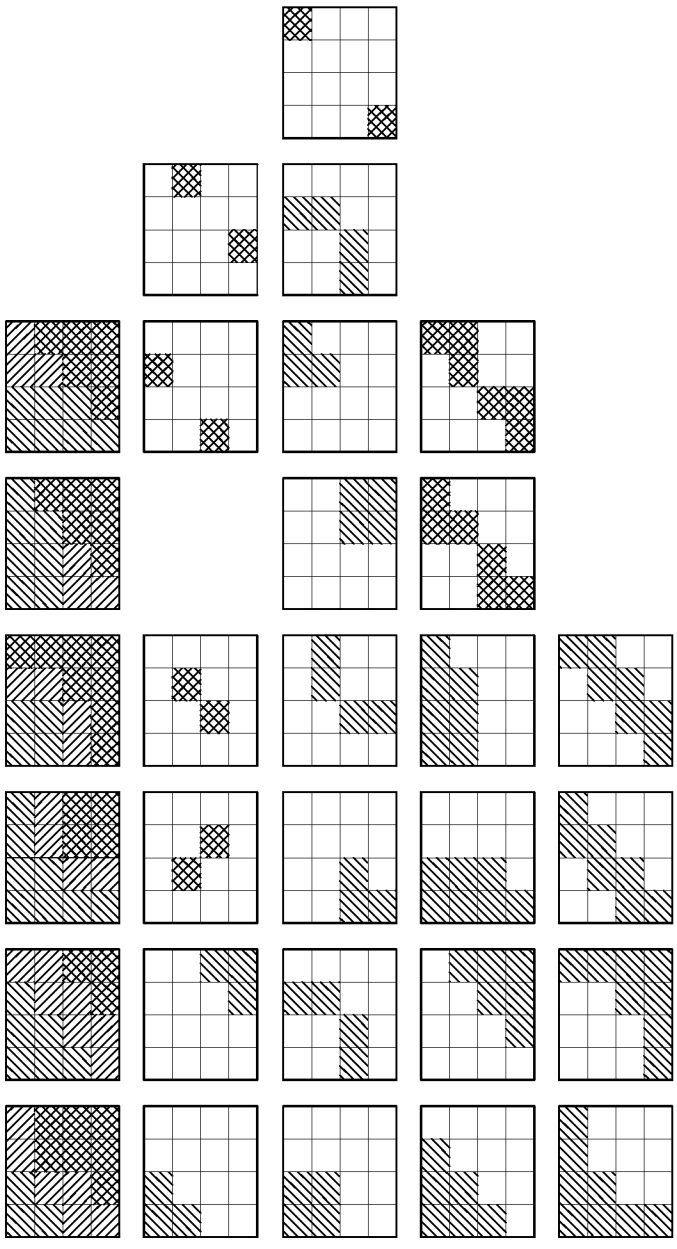

FIG. 21(Continued)

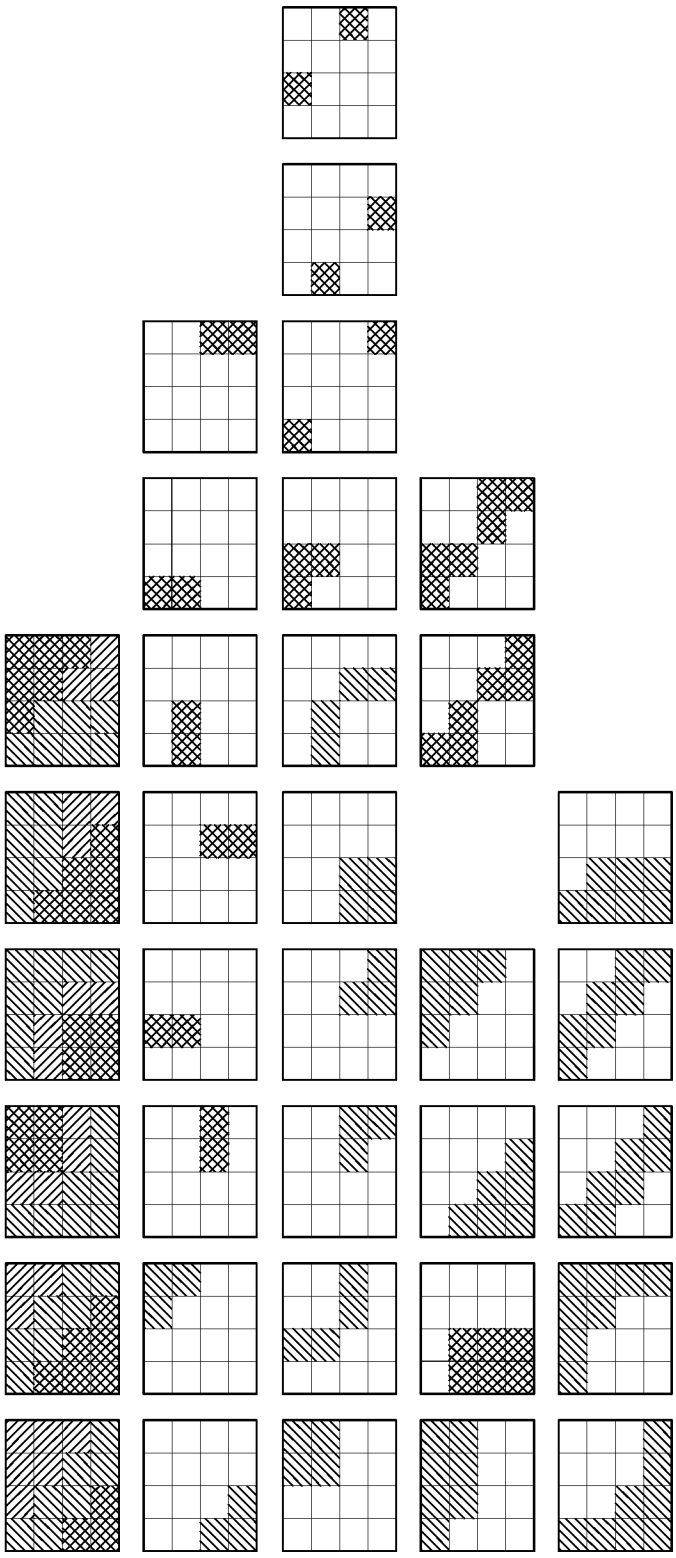

FIG. 21(Continued)

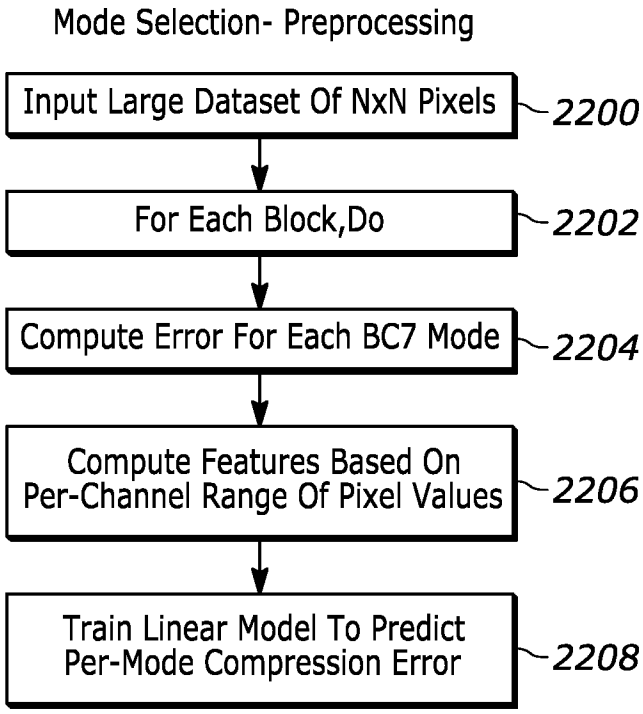
Mode Selection- Preprocessing
Input Large Dataset Of NxN Pixels
2200
For Each Block,Do
2202
Compute Error For Each BC7 Mode
2204
Compute Features Based On Per-Channel Range Of Pixel Values
2206
Train Linear Model To Predict Per-Mode Compression Error
2208

FIG. 22

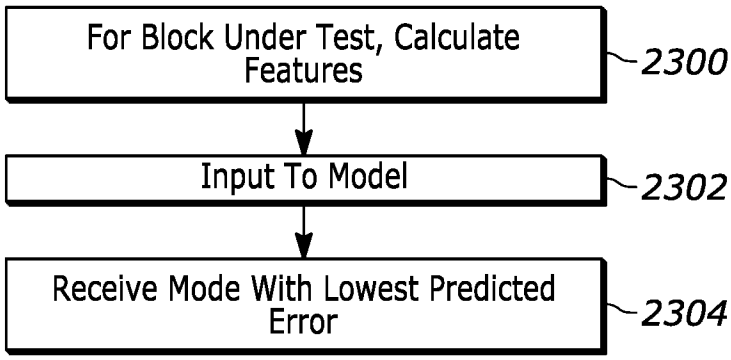
For Block Under Test, Calculate Features
2300
Input To Model
2302
Receive Mode With Lowest Predicted Error
2304

FIG. 23

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2024/033672

A. CLASSIFICATION OF SUBJECT MATTER

IPC: ***G06T 15/04*** (2024.01); ***H04N 19/103*** (2024.01); ***H04N 19/119*** (2024.01); ***H04N 19/176*** (2024.01); ***H04N 19/20*** (2024.01); *H04N 19/42* (2024.01); *G06T 9/00* (2024.01)

CPC: ***G06T 15/04; G06T 11/001; H04N 19/103; H04N 19/176; H04N 19/119; H04N 19/20;*** *G06T 9/002; H04N 19/42*

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

See Search History Document

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

See Search History Document

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

See Search History Document

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 6,972,868 B1 (GONDEK et al.) 06 December 2005 (06.12.2005) entire document	1-5, 7, 8, 10-16, 18, 20
Y	US 2020/0051287 A1 (ELECTRONIC ARTS INC.) 13 February 2020 (13.02.2020) entire document	1-5, 7, 8, 10, 13, 20
Y	US 2020/0051312 A1 (NVIDIA CORPORATION) 13 February 2020 (13.02.2020) entire document	2
Y	US 2020/0296425 A1 (QUALCOMM INCORPORATED) 17 September 2020 (17.09.2020) entire document	5, 16
Y	US 2022/0209790 A1 (IMAGINATION TECHNOLOGIES LIMITED) 30 June 2022 (30.06.2022) entire document	7, 8, 11-16, 18, 20

☑ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"D" document cited by the applicant in the international application
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
21 August 2024 (21.08.2024)	**03 September 2024 (03.09.2024)**

Name and mailing address of the ISA/US	Authorized officer
Mail Stop PCT, Attn: ISA/US **Commissioner for Patents** **P.O. Box 1450, Alexandria, VA 22313-1450**	**MATOS** **TAINA**
Facsimile No. **571-273-8300**	Telephone No. **571-272-4300**

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2024/033672

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2022/0092826 A1 (ADVANCED MICRO DEVICES INC. et al.) 24 March 2022 (24.03.2022) entire document	1-20