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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
7 November 2002 (07.11.2002)

PCT

(10) International Publication Number
WO 02/089486 A2

(51) International Patent Classification[7]: H04N 7/24

(21) International Application Number: PCT/CA02/00641

(22) International Filing Date: 1 May 2002 (01.05.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
2,345,878 1 May 2001 (01.05.2001) CA

(71) Applicant *(for all designated States except US)*: **DESTINY SOFTWARE PRODUCTIONS INC.** [CA/CA]; 950-555 West Hastings Street, Vancouver, British Columbia V6B 4N6 (CA).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **VESTERGAARD, Steve** [CA/CA]; 6330 Chatham Street, West Vancouver, British Columbia V7W 2E2 (CA). **TSUE, Che-Wai (William)** [CA/CA]; 1398 El Camino Drive, Coquitlam, British Columbia V3E 2W6 (CA).

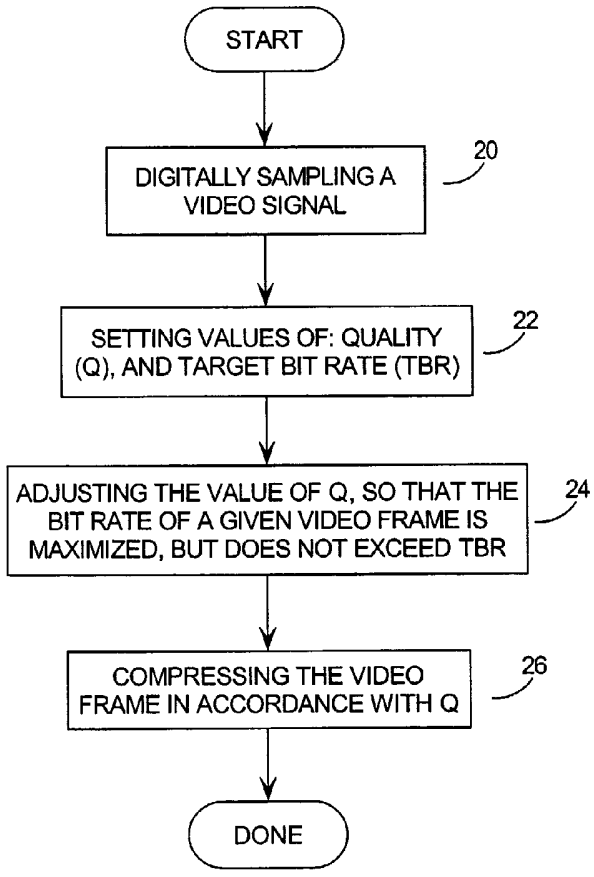
(74) Agents: **LEDWELL, Kent, M.** et al.; Gowling Lafleur Henderson LLP, 160 Elgin Street, Suite 2600, Ottawa, Ontario K1P 1C3 (CA).

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

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM),

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR VIDEO COMPRESSION AND DISTRIBUTION



(57) Abstract: Use of the Internet is widespread, particularly using dial-up modems. However, there is currently no method of compressing high quality video content that optimises the quality for a given, available bandwidth. As well, there is no method for compressing video content for that can be decompressed on a computing device with minimal processing power. Typical video compression/decompression systems require a software application or browser plugin on the end user's computer because their decompression requires a great deal of processing. The invention provides a method and system in which video data can be decompressed in real time, in a Java environment, so executable video applets can be posted on Web sites, and be universally accessible. This end users do not have to download, configure and upgrade multiple software applications or plugins.

WO 02/089486 A2

European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

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

Method and System for Video Compression and Distribution

The present invention relates generally to communications, and more specifically, to a method and system for compressing high quality video content for transfer over communication networks, and subsequently decompressing it on a computing device with minimal processing power.

Background of the Invention

Over the last two decades, tremendous advances have been made in the availability and capability of communication networks and devices. Hard-wired telephone systems have evolved to include wireless telephone and pager networks based on satellite, cellular, wireless local loop, line of sight and other wireless technologies. Data communication networks such as the Internet, Wide Area Networks (WANs) and Local Area Networks (LANs) have also become widespread, and support many different devices including personal and laptop computers, personal digital assistants (PDAs) and television set top boxes. There are also devices and networks which operate in both telephony and data environments, combining technologies of various types.

These telephony and data networks are generally converging to a model in which data are transferred in multiple packets of digital symbols. These data packets may travel independently of one another both in terms of the routings they take, and the time they take to travel from the originator to the destination. This model has proven to be very successful and is the basis, for example, of the protocols used in the Internet.

An area of particular interest is the communication of video content between devices and over networks. Communicating high quality video content requires a great deal of bandwidth over the network that is transferring the content.

Full-motion video is essentially a sequence of still images that are replaced so fast that a human perceives them to be in continuous and smooth motion - 60 images per second being considered high quality and 20 images per second generally being accepted as the lower limit of continuous-motion. However, even a relatively poor quality image requires a great deal of bandwidth to view; for example a video sequence that has dimensions of 200 x 200 pixels, with 24 bits of colour resolution, at a rate of 10 frames per second, will require 9.6 megabits per second

(MBPS) of data. Thus, it is clearly desirable to compress video data which is to be stored or transmitted.

Techniques are known for compressing video content to minimize the demands on the network resources, but these techniques have two fundamental problems:

1. they place large computational demands on the devices performing the decompression, so they must be implemented by dedicated media players or browser plugins on the computer playing the content. This causes logistical problems in the distribution of media files; and
2. they apply a certain set of compression techniques to the input content without regard for the bandwidth of the channel that is to carry the content.

The most commonly used video compression standard is the MPEG - 1 standard (MPEG for Moving Pictures Experts Group), though there are other formats including the AVI, Real, and Quicktime formats. Other less common and proprietary formats are also known. These and similar techniques generally use a software player which exists as an independent software application on the receiver's personal computer, or as a plugin to an Internet browser. Either way, the decompression software must be compatible with the end user's operating system and software platform, and the format of the video file being downloaded. Thus, the end user must obtain the correct software and configure it appropriately for his computer, making periodic upgrades as required. These downloading and configuration tasks can be slow and frustrating, which presents a barrier to access even if the software is available to the end user at no cost.

As there is no single format which has emerged as an industry standard, an end user must perform this exercise for multiple plugins and software applications. Clearly, it is impractical to expect end users to maintain a large number of such formats on their computer. Therefore, a Web site that includes such video content could not be universally accessible.

The MPEG formats are the most commonly used video compression formats, but they are very resource intensive. The original objective of MPEG-1, for example, was to deliver video and video at a data rate of 1.86 megabits per second (MBPS). More advanced video compression techniques such as MPEG-4 and MPEG-7 offer better compression ratios but require much more computation in compression and/or decompression to provide higher quality (MPEG-4 was designed to work in low to high bitrate range).

MPEG videos which require very little bandwidth could be generated simply by limiting the dimensions of the display, or the resolution of the video content being supplied. However, the end user will still require a fully function MPEG player on his personal computer to view the content. Thus, the MPEG player must have the capacity to process the very high bandwidth signals that are part of the MPEG standards.

These very high bandwidth signals require a great deal of complex processing on the end user's computer. In order to decompress MPEG files quickly enough to appear in real time, executable machine code must therefore be used to view MPEG files. This explains why MPEG players are invariably provided as dedicated software applications on the end user's computer or other device: this is the only way they can process the data quickly enough to provide real-time video decompression.

As noted above, the second main problem with the current video compression and decompression software is that there is no optimisation of the data stream for the resources available. Although many compression/decompression systems will allow the user to set certain parameters, such as the maximum bandwidth required to transmit the resulting video file, they do not take into account the original data being compressed and dynamically adjust the compression techniques being use.

As a result, given a fixed set of parameters, low quality video content will be compressed using the same techniques as high quality original video content, when in fact, it may be possible to compress the low quality video content to fit the available bandwidth without any loss in quality. Overall, quality therefore suffers.

There is therefore a need for a video compression and decompression system which provides high quality reproduction, optimised for the bandwidth available, without placing excessive processing demands on the end user's computer.

Summary of the Invention

It is therefore an object of the invention to provide a method and system which obviates or mitigates at least one of the disadvantages described above.

One aspect of the invention is broadly defined as a method of compressing video comprising the steps of: a method of compressing video comprising the steps of: digitally sampling an video signal; setting the value of a quality parameter (Q) and

the value of a targeted bit rate (TBR); calculating a predicted bit rate (PBR) for a video frame of the video signal; responding to the PBR being greater than the TBR by reducing the value of Q for the video frame; and compressing the video frame in accordance with the value of the Q parameter.

Brief Description of the Drawings

These and other features of the invention will become more apparent from the following description in which reference is made to the appended drawings in which:

Figure 1 presents a flow chart of a method of compression in a broad embodiment of the invention;

Figure 2 presents a block diagram of a system for compression, transfer and decompression of video files in a preferred embodiment of the invention;

Figure 3 presents a flow chart of an overall method of handling video files and generating executable Java applets in a preferred embodiment of the invention;

Figures 4A, 4B and **4C** present a flow chart of a method of video file compression in a preferred embodiment of the invention; and

Figure 5 presents a flow chart of a method of video file decompression in a preferred embodiment of the invention.

Description of the Invention

A methodology which addresses the objects outlined above, is presented as a flow chart in **Figure 1**. This figure presents a method of compressing video data in which the compression technique and quality parameters are modified dynamically as compression proceeds, so that the quality is optimised in view of a fixed bandwidth that the content is to traverse.

This compression method is generally effected as follows:

1. first, by digitally sampling a video signal in some manner, at step **20**; then
2. setting the value of a quality parameter (Q) and the value of a targeted bit rate (TBR) at step **22**;
3. adjusting the value of the Q parameter so that the bit rate for a video frame of the video signal is maximized, but will not exceed the TBR at step **24**; and
4. compressing the video frame in accordance with the value of the Q parameter at step **28**.

The video content being compressed can take on any form, analogue or digital. In most computer-based applications, the video content will already be in a digital form, either the product of a digital camera or scanner, or the output of a piece of digital imaging software. Thus, the step of digitally sampling the video signal at step **20** may already have been performed by another component in the system, rather than the core software.

The range and values of the quality parameter (Q) established at step **22**, will depend on the specifics of the software application itself. An example is provided in the description of the preferred embodiment of the invention which follows, but clearly, any suitable framework could be used.

Similar, the invention is not restricted by the value of the targeted bit rate (TBR), or the manner in which it is established. Typically, the software operator will generate several executable video packages which are optimized for different downloading rates. As the bottleneck in the downloading process is generally the "last mile" (i.e. the end user's connection to his Internet Service Provider), these executable video packages would be optimised for the most common connection systems, such as 56.6 kbps (kilobit per second) dial-up modems, and various DSL (digital subscriber line) standards. Thus, the end user can select an executable video package that his Internet connection can carry in real time. (Note that the TBR is not the actual connection speed, but the bandwidth that a video package will not be expected to exceed).

The step of adjusting the Q value to maximize the bit rate can also be effected in a number of manners. As noted above, digital video is generally implemented as a series of still images or frames. Thus, the data required for a finite period of time can always be determined, regardless of whether the data are compressed. The Q value could be determined by trail and error, extrapolated from two or more test points, determined in a feed-forward manner or in a similar manner which would be clear to one skilled in the art from the teachings herein.

Finally, the actual compression of the video frame could be effected in any manner known in the art - most compression techniques allow the quality level to be determined in a manner which would be compatible with the invention. A particularly effective compression technique is described in the preferred embodiment hereinafter.

As noted in the Background, it is desirable to generate video files which use as much of the available bandwidth as allowable. Typical compression systems

simply do not do this, but rather they allow the operator to set certain parameters governing the compression techniques themselves, or fix the quality level. This approach is misguided - by fixing the quality level, there is no certainty that the video will be able to stream over a given channel, in real time. Similarly, by fixing the compression technique without regard for the video data being fed into the software, the quality will inevitably be sub-optimal and the bandwidth the output data will require, will be unpredictable.

The method of the invention keeps the quality of the compressed video as high as possible for the bandwidth that is available. Thus, the compressed file or files will use as much of the available bandwidth as possible, without exceeding its capacity.

The preferred embodiment of the invention described hereinafter also provides further advantages over the prior art.

Detailed Description of Preferred Embodiments of the Invention

The preferred embodiment of the invention is presented by means of the block diagram in **Figure 2**, and the flow charts of **Figures 3** through **5**. **Figure 2** identifies the relevant parties in a transaction of the preferred embodiment of the invention, while the specific processing steps are presented in detail in **Figures 3** through **5**.

In the preferred embodiment, the invention is applied to an Internet and Web site environment. The owner of a Web site (the "software operator") can purchase the software of the invention, use it to compress video clips, and post those video clips on his Web site. These compressed video clips are packaged as executable Java applets, and are represented by an icon on the software operator's Web site. When an end user clicks on the icon, the video file is streamed to the end user's computer or similar device, and immediately begins to play.

In the preferred embodiment a compressed video clip actually consists of two software components: an executable Java applet, and a data file containing the compressed video content. This way, the executable Java applet can be downloaded and begin executing, while the data can be "streamed" separately. This allows the video content to be displayed as soon as sufficient data has been received. If the two files were combined into one, then streaming could not be done - one would have to wait for the entire file to download before it could be executed. This process is described in greater detail hereinafter.

Note that the Java applet and compressed video data files are frequently referred to herein as a single file. This has been done in the interest of simplicity only. It is clearly preferable to implement the invention using separate files to take advantage of the streaming process.

Figure 2 presents an exemplary layout of an Internet communications system **30** in a preferred embodiment of the invention. Generally, the Internet **32** is described as a system of routers interconnected by an Internet backbone network, which allows two parties to communicate via whatever entities happen to be interconnected at any particular time. However, it would be known to one skilled in the art that the Internet **32** is far more complex, consisting of a vast interconnection of computers, servers, routers, computer networks and public telecommunication networks.

End users **34** may access the Internet **32** in a number of manners including modulating and demodulating data signals over a telephone line using audio frequencies, which requires a modem and connection to the Public Switched Telephone Network, which in turn connects to the Internet **32** via an Internet Service Provider **36**. Another manner of connection is the use of set top boxes or cable modems which modulate and demodulate data onto high frequencies which pass over existing telephone or television cable networks and are connected directly to the Internet **32** via Hi-Speed Internet Service Providers. Generally, these high frequency signals are transmitted outside the frequencies of existing services passing over these telephone or television cable networks.

An end user **34** may also obtain access to the Internet **32**, using a digital cellular telephone, pager, or personal digital assistant.

Internet Service Providers (ISPs) **36** or Internet Access Providers (IAPs), are companies that provide access to the Internet. ISPs **36** are considered by some to be distinguished from IAPs in that they also provide content and services to their subscribers, but in the context of this disclosure the distinction is irrelevant. For a monthly fee, ISPs **36** generally provider end users **34** with the necessary software, user name, password and physical access. Equipped with a telephone line modem, cable modem or set top box, one can then log on to the Internet **32** and browse the World Wide Web, and send and receive e-mail.

Web servers **38**, **40** are computers which provide text, graphic or multimedia content, or software applications, to other parties over the Internet **32**. In the discussion of the invention which follows hereinafter, the software operator **42** will

obtain software from the compressor/decompressor software server **38**, and the operator's Web site **40** will provide executable, compressed video files and other content to the end users **34**.

Of course, the invention may be applied to almost any communication network known in the art, and may be applied to a system of several different networks working together. Such networks could include: wireless networks such as cellular telephone networks, the public switched telephone network, cable television networks, the Internet, ATM networks, frame relay networks, local area networks (LANs) and wide area networks (WANs).

Compression

As explained above, the owner of a Web site simply obtains a copy of the encoding software electronically, and uses it to generate executable video applets. These video applets can then be included with any Web page, linked to it using a graphic icon. End users **34** who visit these Web pages download and execute the video applets by clicking on these icons.

In the preferred embodiment of the invention "asymmetric" compression and decompression is used, that is, only simple computations need to be performed during decompression on the end user's device, while more complex processing may be performed during compression. The power and speed of the processing during compression is not limiting because it need not be performed in real time, unlike the decompression.

The preferred embodiment of the routine for generating compressed video files is presented in the flow charts of **Figures 3** and **4**. **Figure 3** focuses on the handling of the original video data, and how the executable Java applet is generated, while **Figure 4** presents the details of the compression routine itself.

The routine begins at step **130** of **Figure 3**, where, in response to the compression software being launched, the algorithm seeks out the video data file that is to be compressed. The targeted file or files may be communicated to the compression software using any technique known in the art, including dragging files to an icon or identifying the file or files in a command line.

The software operator will now be queried to provide the available data rate that the compressed video is intended for, and the quality level that is desired, at step **132**. It may also be desirable to query the software operator for other data, such as the name of the file, where it is to be stored, who generated it and other

such details. The software operator will generally choose a targeted bit rate that corresponds to commonly available bit rates that prospective end users may have. For example, the most common dial up modems have bit rates of 56.6 kbps (kilo bits per second), while common data rates for DSL (digital subscriber line) connections include 128, 256, 300 and 500 kbps. The software operator would therefore enter one of these values and identify the file as such, so that this targeted bit rate can be posted on the Web site along with the compressed file.

The quality level is adjusted automatically by the program, so the software operator can set a high value so that the algorithm attempts to maximize the quality at all times. The default setting will also be to the highest level for the TBR.

Next, the algorithm will consider each frame of the targeted file successively at step **134**, and perform the compression routine of **Figure 4** on that frame, at step **136**. As noted above, all digital video data can ultimately be described as a series of individual frames which are successively displayed to the end user.

While the invention will typically be applied to frames of digital RGB data (Red Green Blue data is a standard format for digital video data) in MPEG format, it may also be applied to other input formats. The handling of the RGB MPEG data will be described in greater detail with respect to **Figure 4**.

Once all of the frames have been compressed, the routine will have generated a Java-based executable, compressed video file which may be stored on the Web server or other storage device at step **140**. The Java-based executable, compressed video file may now be accessed from the Internet simply by placing an icon on a Web page, which is linked to stored Java code on the server (at step **142**).

The details of the compression routine will now be described with respect to the flow chart of **Figure 5**.

As noted at steps **134** and **136** above, each frame of the input video data is considered individually. First, each frame is prepared for the compression routine at step **150** by converting it from the input format (say, for example, from RGB format), to YUV format.

RGB format is a video format that is based on the hues red, green and blue. While this format may seem logical because it correlates intuitively with the images being displayed, it is not the most data efficient manner in which to define an image.

YUV is a colour encoding scheme for images in which luminance and chrominance are separate. The human eye is less sensitive to colour variations than

to intensity variations. YUV is therefore advantageous because it allows the luminance (parameter Y) information to be handled at full bandwidth while the chrominance (parameters U and V) information, to which the human eye is less sensitive, can be compressed. The YUV format is well known in the art of video imaging technology.

While there is no scientifically fixed relationship between the RGB format and the YUV format, the following is a standard relationship as used in the art:

$$Y = (0.299 * R) + (0.587 * G) + (0.114 * B)$$

$$V = (-0.169 * R) - (0.331 * G) + (0.500 * B) + 128$$

$$U = (0.500 * R) - (0.419 * G) - (0.081 * B) + 128$$

For high contrast frames, "blurred YUV" data is also generated, where adjacent pixels are averaged together. The blurred YUV frames are used in motion estimation only, and not for the final encoding of the pixels.

It has been found that motion estimation does not work very well on high contrast or noisy source video because the noise and the edges generate a lot of error. If the motion estimation error threshold is raised to accommodate, it increases the number of matches as well as false matches. However, by blurring the source video before the motion estimation, it enables the algorithm to detect more matches without increasing a lot any false matches.

Next, the algorithm calculates a predicted bit rate (PBR) for the frame being considered, by averaging the bit rate of the previous three seconds of data, at step **152**. Other time periods could also be used to calculate this moving average, as could weighted averaging of the data, or similar techniques. Similarly, one could look ahead to future frames and begin to adjust the quality ahead of time.

The PBR is then compared to the TBR at step **154**, and three cases are established:

1. if the PBR < TBR (that is, the predicted bit rate is less than the bandwidth assumed to be available), then the level of the quality parameter (Q) may be increased at step **156**. While an assignment of Q = Q + 1 is used in the preferred embodiment, clearly another assignment could also be used;
2. if the PBR is approximately equal to the TBR but is still less than it, the Q should be decreased by a small amount at step **158**. In the preferred embodiment, an assignment of Q = Q - 1 was used; and

3. if the PBR is greater than the TBR, the Q must be reduced so that the compressed data will be able to flow over the available channel at step **160**. An assignment of Q = Q + 1 was used in this case.

Control now passes to step **164** of **Figure 4B**, where the frame is now considered in terms of blocks of 8 x 8 pixels.

Each 8 x 8 block in the prepared new frame is now compared with the corresponding block in a history frame at step **166**, and a "motion estimate" is calculated (see step **194** for a discussion on how the history frame is generated). In general, the concept of motion estimation is well know in the art, and is used in various standards including the MPEG standard. The invention adds several improvements to the traditional process though.

Each frame is divided into 8 x 8 blocks of pixels. For each 8 x 8 block, the routine searches the surrounding pixels in the history frame for the closest match. When the closest match is found, an offset vector (x, y), is calculated, and an error level is assigned at step **168**. The error level for the block match is a relative measure comparing the closest matching block to the corresponding block in the history frame. It is only used to perform the decision in step **170**, so clearly a number of different models could be used.

Specifically, the search range is +/- 31 pixels along the X and Y axes, thus there are 63 x 63 = 3969 comparisons in a full search (+0 and -0 are the same, hence 63). To speed up the search, the invention employs a quick hierarchical search algorithm to narrow down the search range in 5 rounds. In the first round, the block is compared with the surrounding eight blocks with offsets (-31, -31), (0, -31), (31, 31), (-31, 0), (0, 0), (31, 0), (-31, 31), (0, 31) and (31, 31). Then the one with the smallest error is chosen to be the base for the next round. In the next round, the range is dropped by half to 16. This process is repeated until a single block remains. The total comparisons required in the quick search is about 8 x 5 + 1 = 40 (1% of the full search). The quick search is able to find over 90% of matches found in a full search.

Based on the current Q value, an error level threshold will be set to classify each block as a moved block, a new block or a no-change block at step **170**. If the error level of the block is higher than the threshold, the block is considered to be new, otherwise, it is classified as moved. A no-change block is a special case of the moved block with offset (0,0).

If the Q value is set to a high quality level, then a comparatively high percentage of blocks will be considered new, and a great deal of bandwidth will be consumed. If the Q value is low, a higher percentage of blocks will be considered "moved blocks", so much less bandwidth will be required.

As shown in **Figure 4B**:

1. if the block error level is low relative to Q, then the current block will be considered a moved block, and the vector for the block will be set to the offset (x, y) determined by the motion estimation analysis, at step **172**;
2. if the block error level is high relative to Q, then the current block will be considered a new block, and the vector for the block will be set out of range at step **174**. This out of range value will be recognized by the decompression software as representing a new block; and
3. if the block error level is close to, or equal to zero, then the current block will be considered an unchanged block, and the vector for the block will be set to (0, 0) at step **176**.

As shown, the steps of **164 - 176** are repeated until all of the blocks in the frame have been considered. When step **164** detects that this has been completed, the frame will now be defined by a set of motion estimation vectors. These motion estimation data are compiled into a motion estimation table which is considered at step **178** of **Figure 4C**.

At step **178**, the routine determines whether it would be more efficient to record all of the block changes in the frame, or whether there are so many that it would be more efficient to simply save the entire frame as a JPEG (Joint Photographic Experts Group) image. The JPEG standard is a high-quality compression standard for still pictures, and is well known in the art.

In the preferred embodiment of the invention, the decision point was taken to be 70%; that is, if 70% or more of the blocks in the frame are new or moved, then the routine branches to step **180** where the entire frame is saved as a JPEG image. Clearly, other decision points could used.

The size of this JPEG image is then compared with the TBR at step **182**, and if the JPEG image is small enough, control passes to step **194**. If the image is too large, then remedial measures are taken at step **184**. A number of things can be done to reduce the size of the full-frame JPEG image, including the following:

1. full frames can be scaled down, for example, to half-frame resolution as indicated at step **184** in **Figure 4C**; or

2. full frames can be compressed with lower quality. Most JPEG generating software has a variable quality setting, which can be linked to the Q value used by the routine of the invention. The quality of course, can be increased if the bitrate is lower than the TBR, or decreased if the bitrate is higher than the TBR.

After the remedial measures are effected, control passes to step **190**.

If the number of new and moved blocks is determined to be small at step **178**, then control passes to step **186** where the motion estimation table is compressed using "Huffman coding". Huffman coding is "lossless" in that no detail or information is lost in compressing and decompressing.

Briefly, Huffman coding is performed by categorizing data points by the likelihood that they will occur. Then, the most common points are assigned shorter codes, and the less likely codes are assigned longer codes.

For example, given six data points A through F, with probabilities as shown below, one could generate a Huffman assignment table as shown:

Point	Probability	Huffman Code	Bit Length
A	0.3	00	2
B	0.3	01	2
C	0.13	100	3
D	0.12	101	3
E	0.1	110	3
F	0.05	111	3

In straight binary coding, a 3-bit word would be required to encoding the six data points A through F. With the Huffman coding, however, the more common data points have 2-bit words, and the longer ones, 3-bit words. Considering the probabilities as noted above, the average word will have 2.34 bits, which gives 2.34/3 = 78% compression, with no loss of data.

As noted above, the motion estimation is recorded as an offset vector (x, y). All (x, y) pairs are sorted and counted, and popular pairs will be assigned a code in the table. Unpopular pairs will be assigned to one single code and will store the offset (x, y) uncompressed. Note again that a vector of (0, 0) means there as no change from the history frame to the new frame, and that an out of range pair is used for new blocks.

The invention uses predetermined Huffman coding tables within the compressor and decompressor. Adaptive Huffman coding or dynamic Huffman

tables could be used, but generally, adaptive Huffman would consume too many CPU cycles to decompress, and dynamic Huffman tables would increase the amount of data to download.

A series of JPEG images are then generated at step **188**, corresponding to the blocks that are identified in the motion estimation table.

At step **190**, it is then determined whether the data for the current frame is less than the TBR. If the data for the frame still exceeds the TBR, in spite of dropping the quality level as described above, the frame is simply dropped at step **192**. Otherwise, the image data is copied to the compressed video data file at step **194**.

In the preferred embodiment, the compressed video data and the executable Java decompression code are stored separately. The executable Java decompression code is stored in a zip file containing the Java Applet byte code, and the compressed video data is stored in a separate file which will be streamed down to the client machine during playback.

A new history frame is then constructed at step **196**, simply by saving the current image in a buffer, so it can be compared with the next image when the routine is repeated. When the routine runs for the first time, the history frame will be null (i.e. the buffer will be empty), so when the first comparison is made to history frame at step **166**, there will be no matches.

Compressed audio data can also be added to the compressed video data file at this point. In the preferred embodiment, the audio and video are interleaved so that synchronization is implicit. For example, the data blocks could be stored as follows: V A A A V A A A V A A A V ("V" characters representing video data, and "A" characters representing audio data).

The number of audio blocks associated with each video frame depends on the audio block size and frame rate, and it varies between frames as it may round up or down to the nearest block. Given, for example, 12 frames per second, 200 sample audio blocks, 8000 samples per second, this will yield:

audio samples per frame = 8000 / 12 = 666.667

audio blocks per frame = 666.667 / 200 = 3.333

Thus, the interleaved blocks will be stored in this manner: V AAA V AAAA V AAA V AAA V AAAA V AAA V AAA V AAAA .... In other words, some of the A groups will have three blocks and some will have four.

Note that even if there is an empty video frame (dropped or no change), there may still be an audio block associated with it, which must be stored. In this case, null video blocks are inserted which have a size of 0 for the data portion. The decompression software recognizes these 0 blocks as null video blocks, thus ensuring synchronization.

Downloading and Executing an Video File

The compression technique of the invention may be resource intensive itself, but it results in compressed video which may be decompressed using simple operations. Thus, decompression can be performed on the end user's computer or other device. These simple operations can be executed quickly enough, that an executable file can be created which can run in a Java environment.

Java is a code interpretor which is almost universally supported by Web browsers for computers and similar devices. More important, it is platform independent, so that Java code and applets may be executed on any platform.

Because it is an interpretor, Java executes applets much slower than executable machine code. Thus, Java cannot execute the processing intensive decompression techniques used by MPEG and similar video standards, fast enough to provide real time, high quality video. Thus, MPEG players must be implemented using browser plugins and external machine code players. This raises the problems noted in the Background: that MPEG and similar video formats lack universality, and that end users must download and update multiple video players on an ongoing basis.

The JPEG imaging standard used in the invention does use some complex processing, but JPEG is supported by Java natively as one of its standard image formats. Thus, the code to produce the JPEG images is in machine code, and is inherent to Java.

Since the decompression routine of the invention can be applied in a Java environment, this allows Web sites, for example, to provide executable applets which can be downloaded and played by an end user with a Java-enabled browser or operating system. These executable applets can also be delivered to end users in other ways such as via Email or Banner Ads.

Using executable applets, the end user does not have to obtain application software or browser plugins to listen to the content. Thus, the end user does not have to address issues of compatibility with his platform and the format of the video

content being downloaded, inconvenience of obtaining upgrades, and possibly requiring many different software packages to address different video formats. In the preferred embodiment of the invention, decompression software is packaged with the video content so the end user simply downloads an executable file.

As noted above, there are two main methods in which the preferred embodiment of the invention would generally be implemented: as an icon on a Web page, which is executed when an end user **34** clicks upon it, or as an applet which executes when anyone visits the Web page. In either case, the method presented in the flow chart of **Figure 6** would be performed.

Execution of this method generally requires that the end user **34** have the following:

1. a device capable of connecting to the Internet;
2. a Java compatible browser, email reader or other Java compatible software application; and
3. a video display system, typically consisting of a video display terminal and a video card.

It is also necessary to have a Java applet residing on an Internet Web server for the end user **34** to download.

The routine begins at step **198** of **Figure 5**, after the end user **34** has initiated the downloading or execution of an applet in one of the manners noted above. The zipped decompression applet is then transmitted to the end user **34** and is unzipped for execution. An auto detection function is performed by the Java applet, as known in the art, to determined the available bandwidth capability of the end user **34**. The applet uses this information to determine which of the previously stored compressed video data files should be downloaded, providing the best quality for the bandwidth available.

The compressed video data is then "streamed" to the end user, using the undocumented but commonly used streaming facility in Java. "Streaming" is the process of beginning to play the content before all of the content has been downloaded. In the method of the invention, content can be decompressed and played as soon as the zipped decompression applet and the first sub-block of the compressed video data have been received; meanwhile, the balance of the compressed video content can be downloaded and decompressed in the background.

The decompression software simply buffers the compressed video content as it is received and decompressed. As the video card on the end user's computer requires data, it simply calls the buffer for data.

Decoding of the compressed video file is performed on a frame by frame basis, in a manner that is complementary to the compression routine described above.

Each frame of the compressed video file is considered separately at step **200**. To begin with, the motion estimate data for the frame is decoded using a predetermined Huffman table that was downloaded with the executable Java applet, at step **202**.

The JPEG image or images associated with the frame, are then decompressed at step **204**, using a standard JPEG engine as known in the art. If the frame was stored as a full-frame JPEG image, then there will only be one image to decompress and copy to the new frame.

If the frame was stored as an assembly of moved blocks, then these blocks are copied from the history frame to the new frame at step **206**, being moved in accordance with their decoded motion vectors.

Blocks which are unchanged from the history frame are then copied from the history frame at step **208**, to the new frame.

If the JPEG image had been scaled down at step **184**, it is now returned to full scale at step **210**, and is copied to the new frame.

The completed image is now ready for display, and is transferred to the video card for display on the end user's video interface at step **212**. This new frame is then buffered as the new history frame at step **214**, so that it can be used as the basis for the next frame (if required). Control then returns to step **200**, so that the next frame can be considered.

To summarize, the method of the invention provides a number of marketable advantages including the following:

1. compressed, high quality video can be streamed over the existing Internet using a standard 56.6 modem and telephone line, or higher speed connections;
2. asymmetric compression provides sufficiently fast decompression, making a platform independent, Java implementation possible;

3. no plugins or players are required on the end user's device, so there are no difficulties with system compatibility, having to obtain multiple media players, or having to purchase or upgrade software;
4. the decompression software can be run on any Java-enabled Web browser;
5. a regular Web server can be used - special servers are not required;
6. the method of the invention allows various sizes of compressed files to be created, so that the end user **34** can obtain an executable file that is optimal for the bandwidth of his network connection. Most compression systems are static, and the end user **34** has no choice of which video file to download;
7. the efficient compression of the preferred embodiment provides for an extremely small download; and
8. the software operator **42** does not have to perform any complex programming to compress his files or to generate executable applets which may be loaded onto his Web site. All programming code is generated automatically.

The invention is not limited by the nature of the Web page being transmitted. The invention could be used to insert simple banners into Web pages, or more sophisticated multimedia advertisements. As well, these advertisements could be sent along with real audio, real video, telephone over Internet, video conferencing over Internet, or other data and software applications.

While particular embodiments of the present invention have been shown and described, it is clear that changes and modifications may be made to such embodiments without departing from the true scope and spirit of the invention.

Portions of the invention could be implemented in part, in different applications.

As well, the invention offers a compression technique that is particularly effective in view of the current trade-off between available resources and video quality. It is expected that the bandwidth and speed of existing communication networks, and the available processing power on various computing platforms will continue to improve, thus the tradeoff curve will slowly shift. This will allow the method of the invention to be implemented with more resource intensive functions. For example, the preferred embodiment of the invention avoids certain resource-intensive techniques such as sub-pixel motion estimation and delta/error encoding. However, as processing speeds increase, it may become practical to employ such techniques.

The method steps of the invention need not be implemented as Java code, but may be embodiment in sets of executable machine code stored in a variety of formats such as object code or source code. Clearly, the executable machine code may be integrated with the code of other programs, implemented as subroutines, by external program calls or by other techniques as known in the art.

The embodiments of the invention may be executed by a computer processor or similar device programmed in the manner of method steps, or may be executed by an electronic system which is provided with means for executing these steps. Similarly, an electronic memory medium such computer diskettes, CD-Roms, Random Access Memory (RAM), Read Only Memory (ROM) or similar computer software storage media known in the art, may be programmed to execute such method steps. As well, electronic signals representing these method steps may also be transmitted via a communication network.

The invention could be applied to all manner of appliances having computer or processor control and communication capability, including computers, smart terminals, lap top computers, personal digital assistants, cellular telephones, Blue-tooth devices, Internet-ready telephones, televisions, television set top units, and automobiles. Such implementations would be clear to one skilled in the art, and do not take away from the invention.

WHAT IS CLAIMED IS:

1. A method of compressing video comprising the steps of:

digitally sampling an video signal;

setting the value of a quality parameter (Q) and the value of a targeted bit rate (TBR);

adjusting said value of said Q parameter so that the bit rate for a video frame of said video signal is maximized, but will not exceed said TBR; and

compressing said video frame in accordance with said value of said Q parameter.

2. The method of claim 1, wherein said step of adjusting comprises the steps of:

calculating a predicted bit rate (PBR) for a video frame of said video signal; and

responding to said PBR being greater than said TBR by reducing the value of Q for said video frame.

3. The method of claim 1, wherein said step of responding further comprises the step of:

responding to said PBR being less than said TBR by increasing the value of Q for said frame.

4. The method of claim 1, wherein said step of calculating a predicted bit rate (PBR) comprises the step of:

averaging the bit rate of the previous three seconds of compressed video data.

5. The method of claim 1, further comprising the step of:

responding to a frame being high contrast, by performing motion estimation on "blurred YUV" data

6. The method of claim 1, further comprising the step of: generating blurred YUV data by averaging adjacent pixels together; whereby motion estimation using blurred data will detect more matches without increasing the occurrence of false matches.

7. A method of compressing video comprising the steps of:

digitally sampling a video signal;

for each frame of said digitally sampled video signal:

converting said frame to YUV format;

dividing said YUV format frame into blocks;

for each block:

calculating a motion estimation vector between said frame and a previous frame;

compressing said motion estimation vector using Huffman compression; and

generating a JPEG image of said block; and

merging said Huffman compressed motion estimation vectors and said JPEG images with a Java decompression program.

8. A method of compressing video comprising the steps of:

digitally sampling a video signal;

establishing an available bitrate;

for each frame of said digitally sampled video signal:

converting said frame to YUV format;

calculating bit rate over the last three seconds;

dividing said YUV format frame into blocks;

for each block:

calculating a motion estimation vector between said frame and a previous frame;

classifying said block as moved, new or no-change, in response to said available bit-rate;

compressing said motion estimation vector using Huffman compression; and

generating a JPEG image of said block, the quality of said JPEG image being dependent on said bitrate; and

merging said Huffman compressed motion estimation vectors and said JPEG images with a Java decompression program.

9. The method of claim 2, where said step of coding comprises the step of Huffman coding said residual values.

10. The method of claim 1, wherein said step of digitally sampling further comprises the step of dividing said digital samples into blocks for said linear prediction operation.

11. The method of claim 9, further comprising the steps of:
sub-dividing each said block into sub-blocks;
searching past data to identify a prior sub-block which correlates to a current sub-block;
calculating a scaling factor which best matches said prior sub-block to said current sub-block;
recording the starting point of said prior sub-block and the scaling factor; and
subtracting said scaled prior sub-block from said current sub-block.

12. The method of claim 1, further comprising the step of:
wrapping said residual data in a Java applet, including a decompression routine.

13. A method of decompressing a compressed video file comprising the steps of:
receiving data corresponding to said compressed video file;
sorting said data into blocks; and
linearly expanding said data, reproducing said video file.

14. A system for executing the method of any one of claims 1 through 13.

15. A computer readable memory medium for storing software code executable to perform the method steps of any one of claims 1 through 13.

16. A carrier signal incorporating software code executable to perform the method steps of any one of claims 1 through 13.

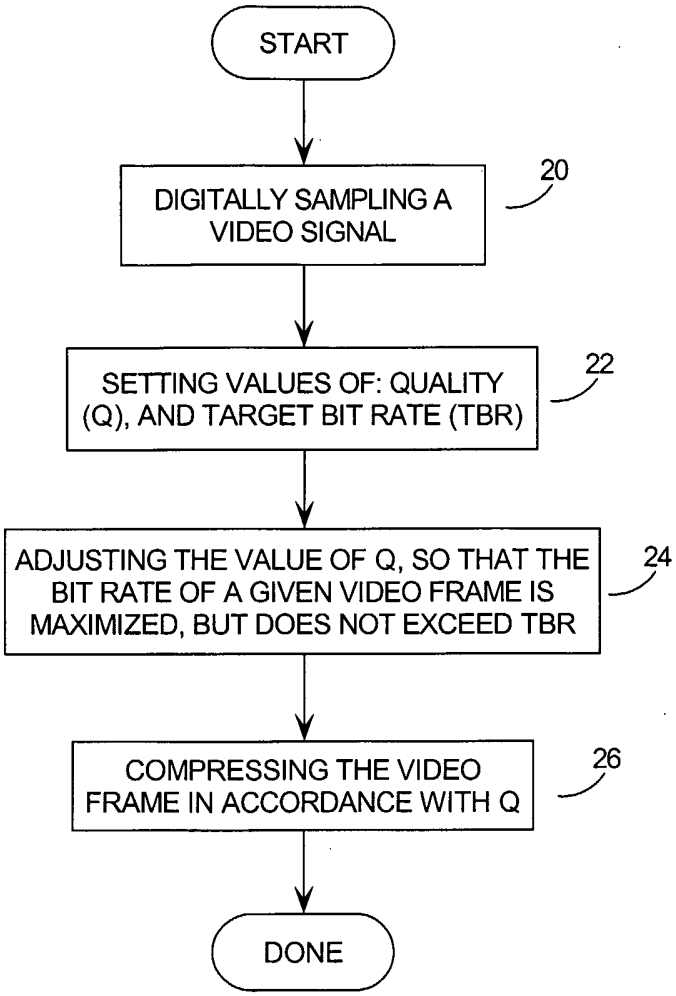

START
DIGITALLY SAMPLING A VIDEO SIGNAL
20
SETTING VALUES OF: QUALITY (Q), AND TARGET BIT RATE (TBR)
22
ADJUSTING THE VALUE OF Q, SO THAT THE BIT RATE OF A GIVEN VIDEO FRAME IS MAXIMIZED, BUT DOES NOT EXCEED TBR
24
COMPRESSING THE VIDEO FRAME IN ACCORDANCE WITH Q
26
DONE

FIGURE 1

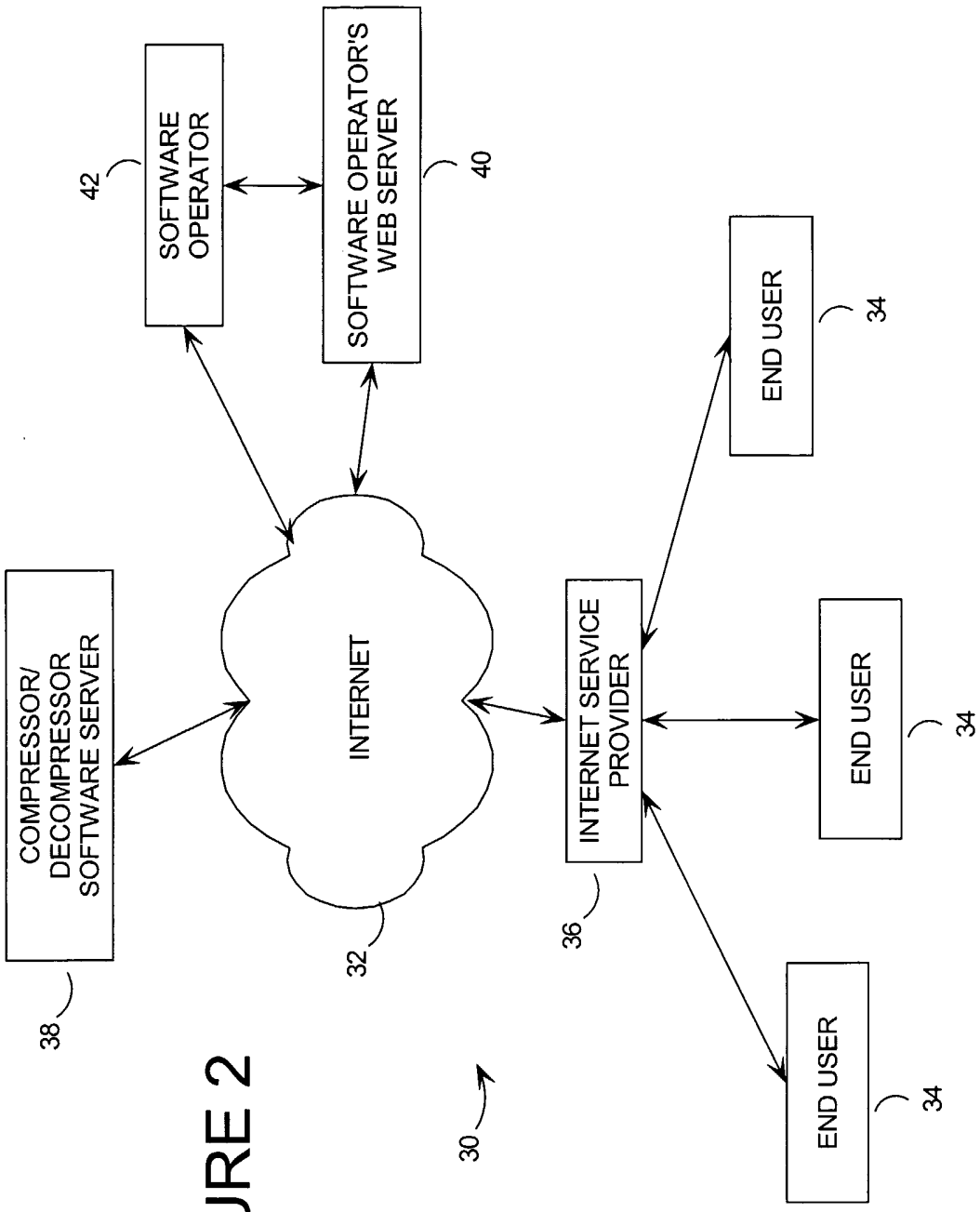

SOFTWARE OPERATOR
42
SOFTWARE OPERATOR'S WEB SERVER
40
END USER
34
COMPRESSOR/ DECOMPRESSOR SOFTWARE SERVER
38
INTERNET
32
INTERNET SERVICE PROVIDER
36
END USER
34
END USER
34
30

FIGURE 2

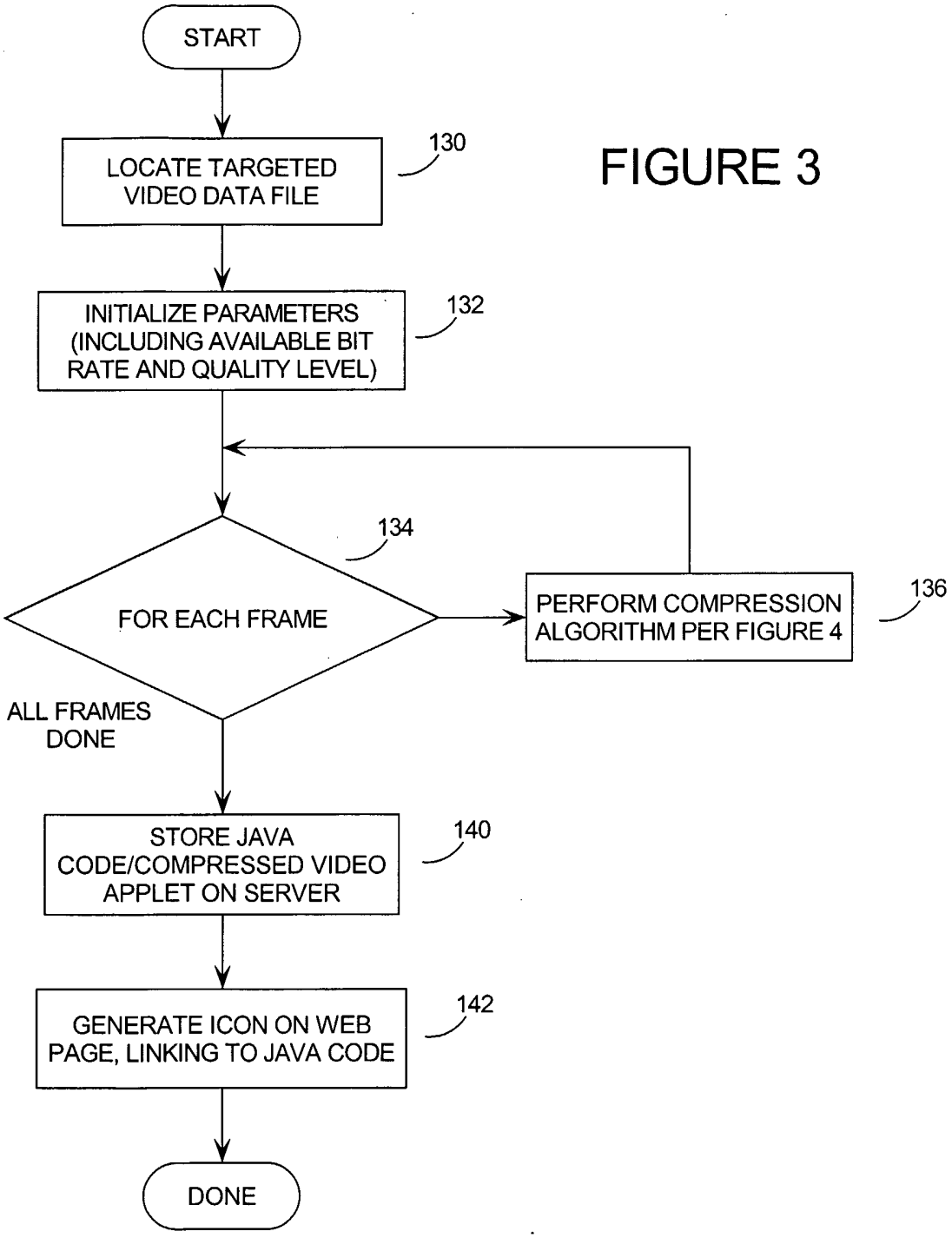
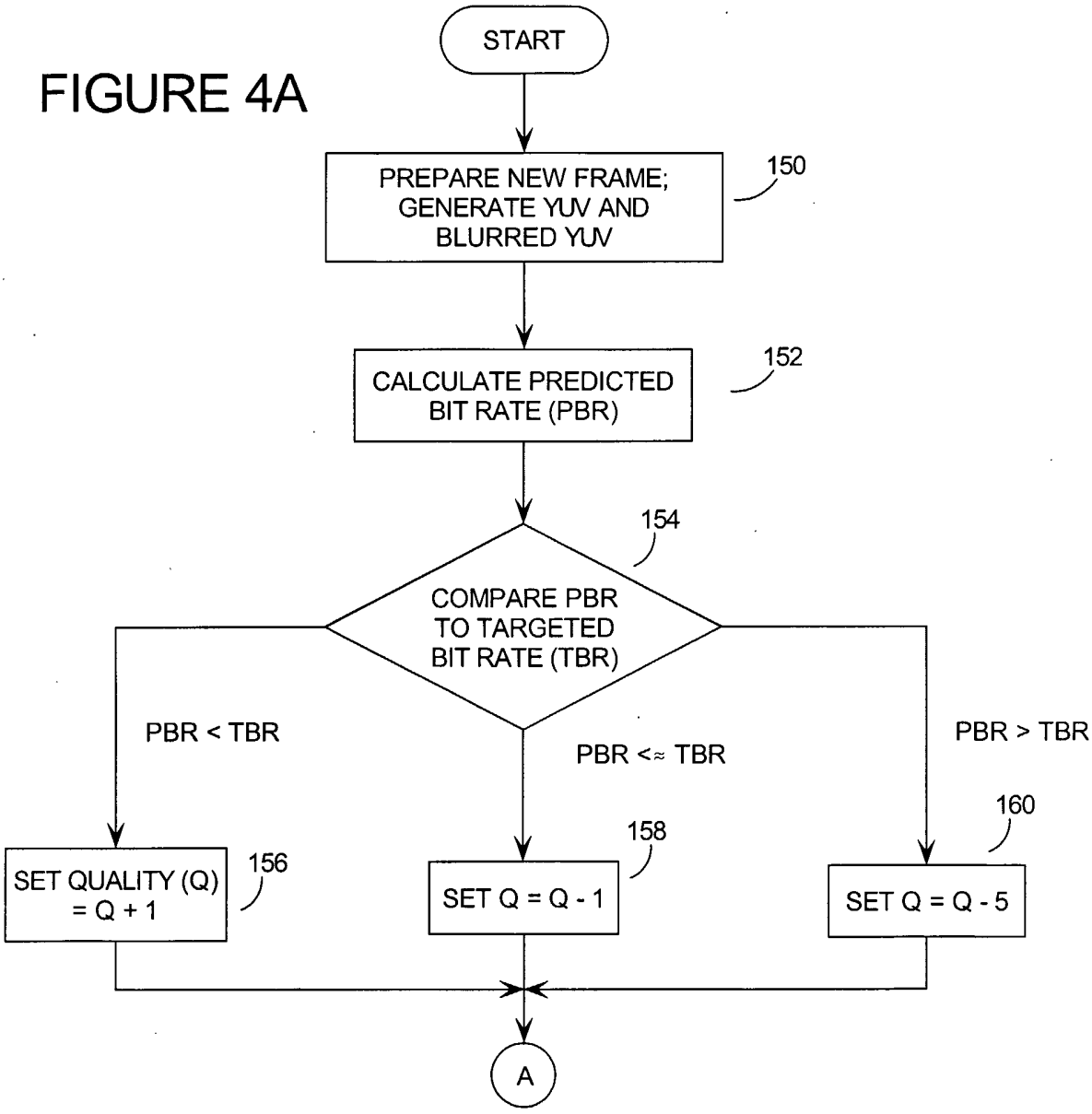

START
LOCATE TARGETED VIDEO DATA FILE
130
INITIALIZE PARAMETERS (INCLUDING AVAILABLE BIT RATE AND QUALITY LEVEL)
132
134
FOR EACH FRAME
PERFORM COMPRESSION ALGORITHM PER FIGURE 4
136
ALL FRAMES DONE
STORE JAVA CODE/COMPRESSED VIDEO APPLET ON SERVER
140
GENERATE ICON ON WEB PAGE, LINKING TO JAVA CODE
142
DONE

FIGURE 3

FIGURE 4A


START
PREPARE NEW FRAME; GENERATE YUV AND BLURRED YUV
150
CALCULATE PREDICTED BIT RATE (PBR)
152
154
COMPARE PBR TO TARGETED BIT RATE (TBR)
PBR < TBR
PBR <≈ TBR
PBR > TBR
SET QUALITY (Q) = Q + 1
156
SET Q = Q - 1
158
SET Q = Q - 5
160
A

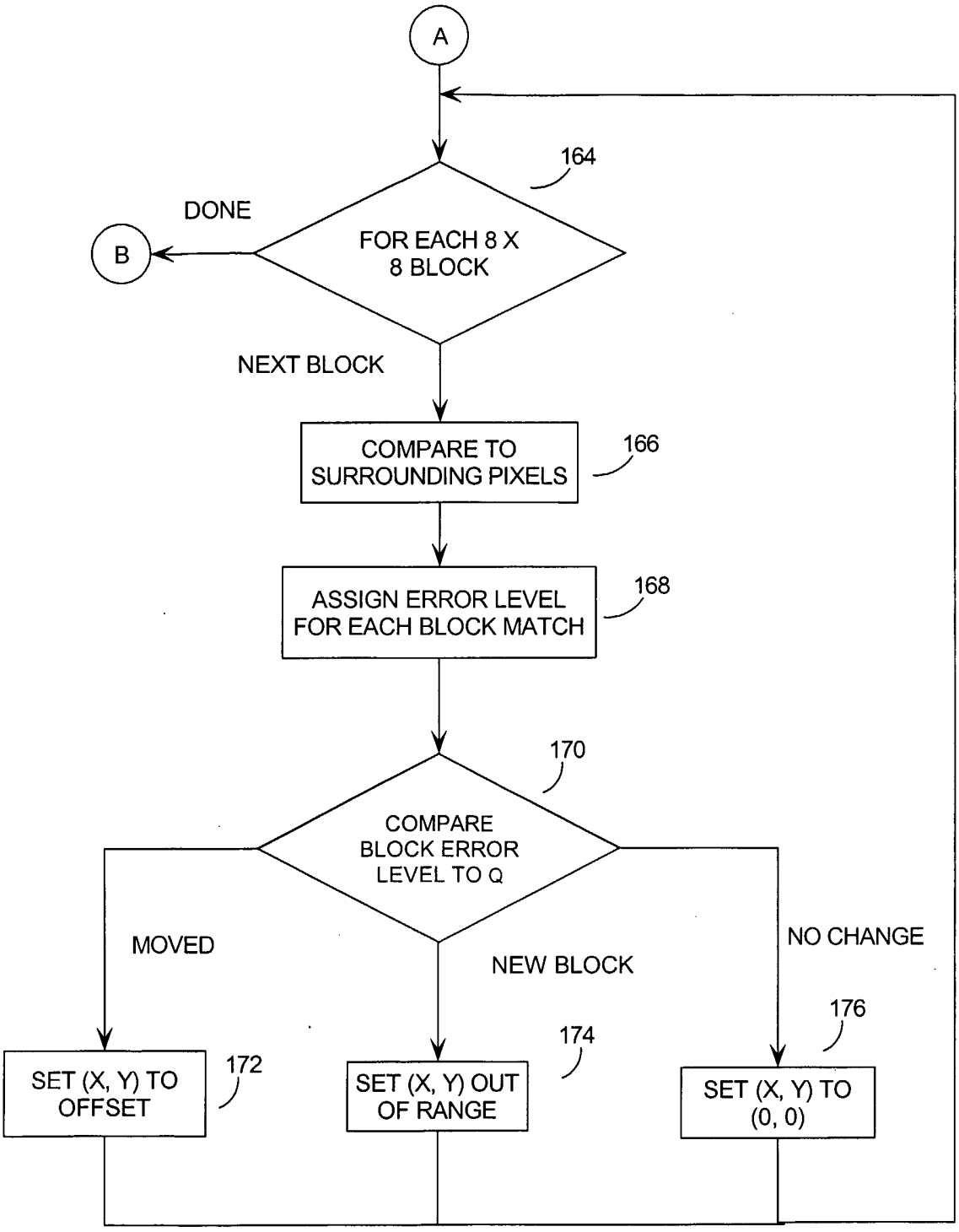

A
164
DONE
B
FOR EACH 8 X 8 BLOCK
NEXT BLOCK
COMPARE TO SURROUNDING PIXELS
166
ASSIGN ERROR LEVEL FOR EACH BLOCK MATCH
168
170
COMPARE BLOCK ERROR LEVEL TO Q
MOVED
NEW BLOCK
NO CHANGE
176
174
172
SET (X, Y) TO OFFSET
SET (X, Y) OUT OF RANGE
SET (X, Y) TO (0, 0)

FIGURE 4B

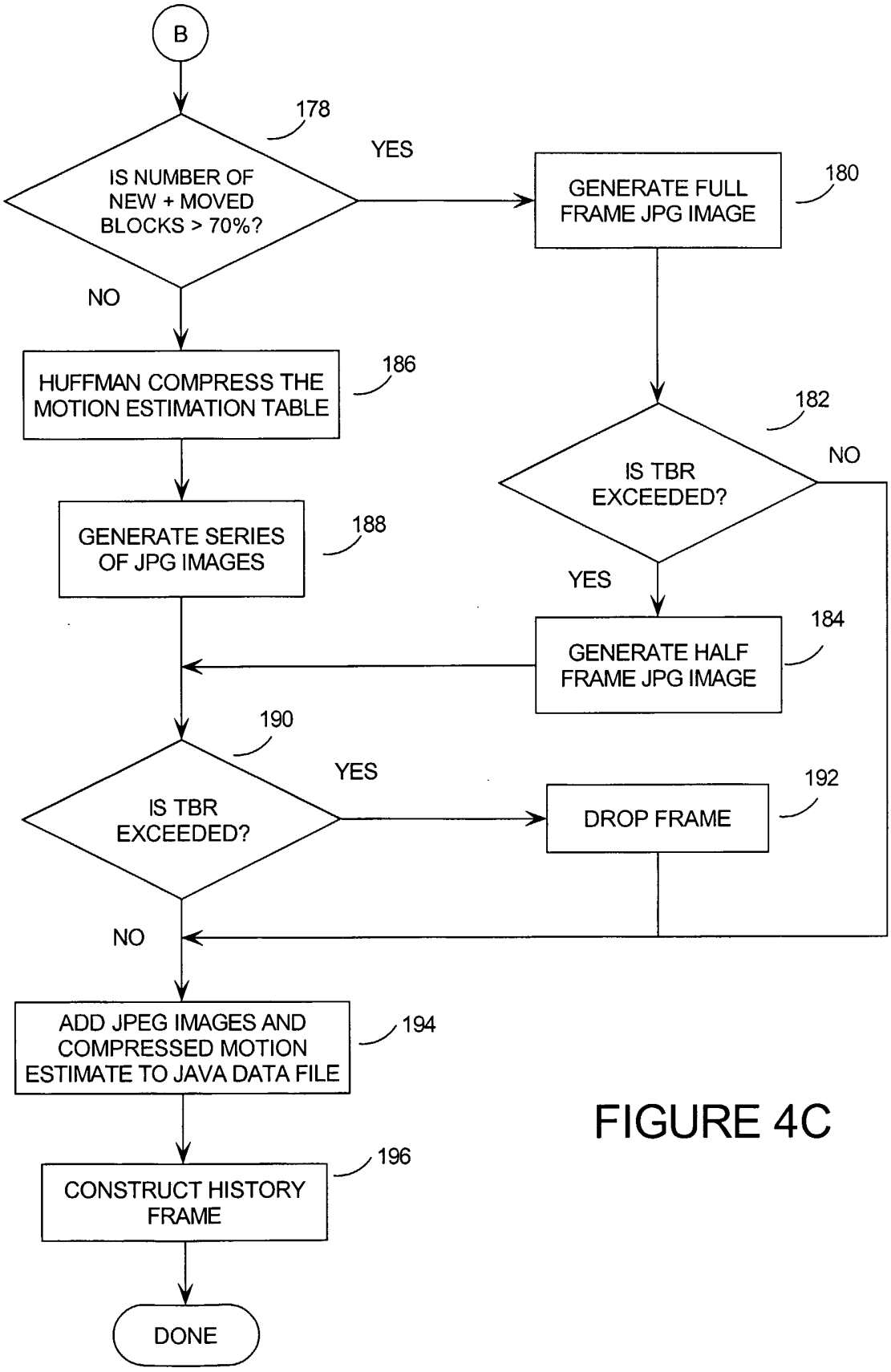
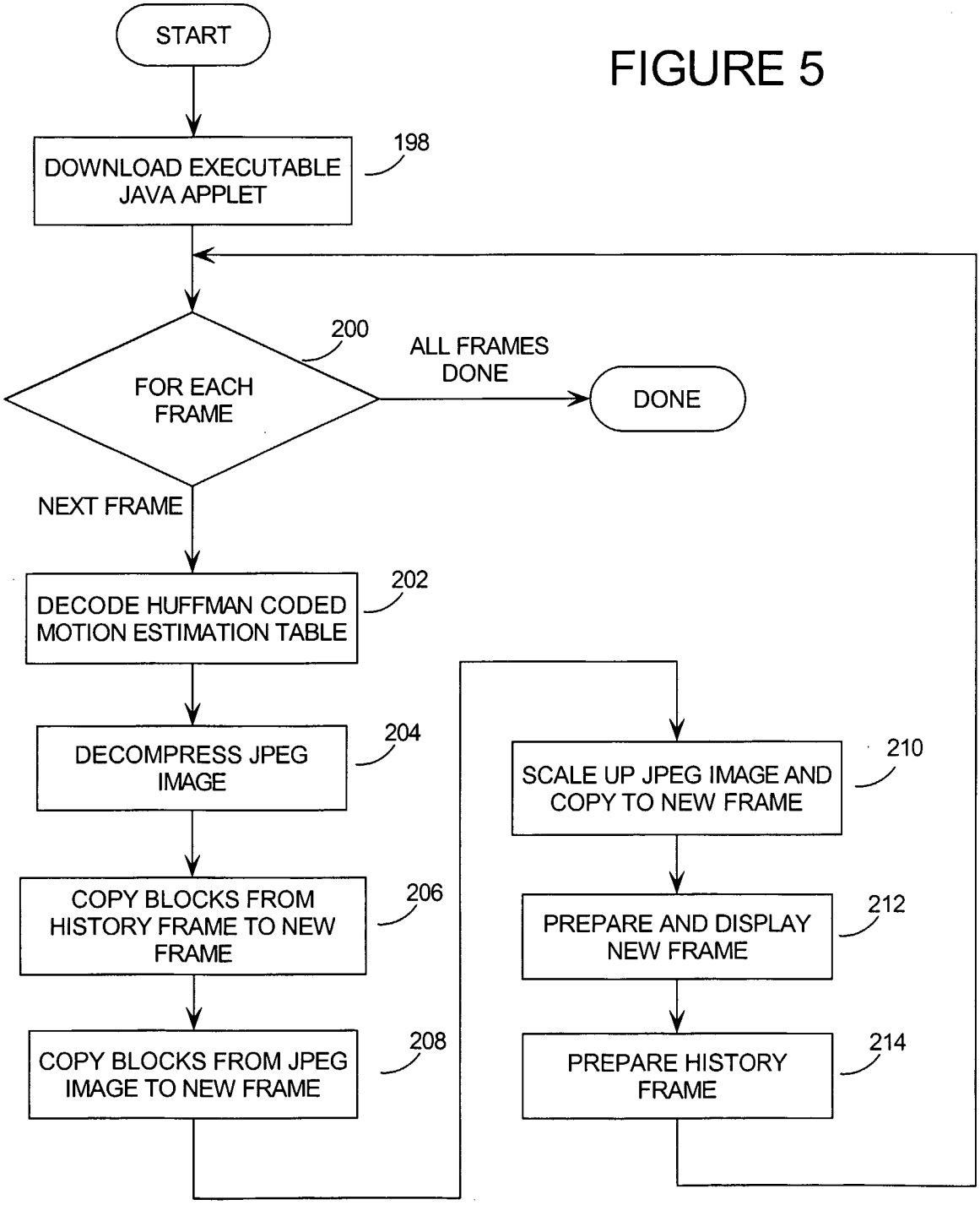

B
178
IS NUMBER OF NEW + MOVED BLOCKS > 70%?
YES
NO
GENERATE FULL FRAME JPG IMAGE
180
HUFFMAN COMPRESS THE MOTION ESTIMATION TABLE
186
182
IS TBR EXCEEDED?
NO
YES
GENERATE SERIES OF JPG IMAGES
188
GENERATE HALF FRAME JPG IMAGE
184
190
IS TBR EXCEEDED?
YES
DROP FRAME
192
NO
ADD JPEG IMAGES AND COMPRESSED MOTION ESTIMATE TO JAVA DATA FILE
194
CONSTRUCT HISTORY FRAME
196
DONE

FIGURE 4C

FIGURE 5


START
DOWNLOAD EXECUTABLE JAVA APPLET
198
FOR EACH FRAME
200
ALL FRAMES DONE
DONE
NEXT FRAME
DECODE HUFFMAN CODED MOTION ESTIMATION TABLE
202
DECOMPRESS JPEG IMAGE
204
COPY BLOCKS FROM HISTORY FRAME TO NEW FRAME
206
COPY BLOCKS FROM JPEG IMAGE TO NEW FRAME
208
SCALE UP JPEG IMAGE AND COPY TO NEW FRAME
210
PREPARE AND DISPLAY NEW FRAME
212
PREPARE HISTORY FRAME
214