IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

UNILOC 2017 LLC, Plaintiff, v. BITMOVIN, INC., Defendant.	C.A. No. 19-cv-179-CFC JURY TRIAL DEMANDED

SECOND AMENDED COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff Uniloc 2017 LLC ("Uniloc"), by and through the undersigned counsel, hereby files this Second Amended Complaint and makes the following allegations of patent infringement relating to U.S. Patent Nos. 6,628,712 (the "'712 patent"), 6,895,118 (the "'118 patent"), 6,519,005 (the "'005 patent") and 6,470,345 (the "'345 patent") (collectively "the Asserted Patents") against Defendant Bitmovin, Inc. ("Bitmovin") and alleges as follows upon actual knowledge with respect to itself and its own acts, and upon information and belief as to all other matters.

NATURE OF THE ACTION

1. This is an action for patent infringement. Uniloc alleges that Bitmovin has infringed and/or is infringing one or more of the '712 patent, the '118 patent, the '005 patent and the '345 patent, copies of which are attached as Exhibits A-D, respectively.

2. Uniloc alleges that Bitmovin directly infringes and/or has infringed the Asserted Patents by making, using, offering for sale, selling, and/or importing various products and services that: (1) dynamically switch and transcode program video and advertisement videos, (2) perform a method of coding a digital image comprising macroblocks in a binary data stream, (3)

perform a method for motion coding an uncompressed (pixel level) digital video data stream perform a method for providing content via a computer network and a computer system and (4) perform a method for replacing substrings in file and directory pathnames with tokens in a computer-implemented file system, such as the Bitmovin DASH compatible video player. Uniloc seeks damages and other relief for Bitmovin's infringement of the Asserted Patents.

THE PARTIES

3. Uniloc 2017 LLC is a Delaware corporation having places of business at 1209 Orange Street, Wilmington, Delaware 19801 and 620 Newport Center Drive, Newport Beach, California 92660.

4. Upon information and belief, Bitmovin is a Delaware corporation with a place of business at 301 Howard Street, Suite 1800, San Francisco, California 94015. Bitmovin may be served through its registered agent at The Company Corporation, 251 Little Falls Drive, Wilmington, Delaware 19808.

JURISDICTION AND VENUE

5. This action for patent infringement arises under the Patent Laws of the United States, 35 U.S.C. § 1 et. seq. This Court has original jurisdiction under 28 U.S.C. §§ 1331 and 1338.

6. This Court has both general and specific personal jurisdiction over Bitmovin because Bitmovin is a Delaware corporation that has committed acts within this District giving rise to this action and has established minimum contacts with this forum such that the exercise of jurisdiction over Bitmovin would not offend traditional notions of fair play and substantial justice. Bitmovin, directly and through subsidiaries and intermediaries (including distributors, retailers, franchisees and others), has committed and continues to commit acts of infringement in

this District by, among other things, making, using, testing, selling, importing, and/or offering for sale products that infringe the Asserted Patents.

7. Venue is proper in this District and division under 28 U.S.C. §§1391(b)-(d) and 1400(b) because Bitmovin is incorporated in this District, transacts business in this District and has committed and continues to commit acts of direct infringement in this District.

<u>COUNT I: INFRINGEMENT OF THE ’712 PATENT</u>

8. The allegations of paragraphs 1-7 of this First Amended Complaint are incorporated by reference as though fully set forth herein.

9. Uniloc owns by assignment the entire right, title, and interest in the ’712 patent.

10. The ’712 patent, titled “Seamless Switching of MPEG Video Streams,” issued on September 30, 2003. A copy of the ’712 patent is attached as Exhibit A. The priorty date for the ’712 patent is November 23, 1999. The inventions of the ’712 patent were developed by an inventor at Koninklijke Philips Electronics N.V.

11. Pursuant to 35 U.S.C. § 282, the ’712 patent is presumed valid.

12. Claim 4 of the ’712 patent reads as follows:

> 4. A method of switching from a first compressed data input stream to a second compressed data input stream, resulting in a compressed data output stream, said method of switching comprising the steps of:
>
> buffering, in which the data contained in the first and the second input stream are stored,
>
> controlling the storage of the input streams during the buffering step in order to switch, at a switch request, from the first input stream to the second input stream,
>
> transcoding the stream provided by the control step, the transcoding includes controlling occupancy of a buffer by feedback to DCT coefficient quantization in order to provide the output stream in a seamless way.

13. The invention of claim 4 of the ’712 patent concerns a novel method for

switching from a first compressed data input stream to a second compressed data input stream, resulting in a compressed data output stream. ’712 patent at 1:6-9. Such an invention is useful in switching and editing MPEG compressed video signals. ’712 patent at 1:10-11.

14. At the time of invention of the ’712 patent, encoding/decoding systems included a method of switching from a first encoded video sequence to a second one. ’712 patent at 1:15-19. In order to avoid underflow or overflow of the decoded buffer, transcoding of the input streams is used to shift the temporal position of the switching point and to obtain at the output of the transcoders, streams containing an identical entry point and the same decoder buffer characteristics. *Id.* at 1:19-24. This prior art method has several major drawbacks. According to the background art, the output bit rate of each transcoder is equal to its input bit rate, which makes the switching method not very flexible. *Id.* at 1:15-28. Finally, the solution of the background art is rather complex and costly to implement as the switching device needs two transcoders. *Id.* at 1:32-35.

15. As demonstrated below, the claimed invention of claim 4 of the ’712 patent provides a technological solution to the problem faced by the inventors—transcoding the stream provided by the controlling of two input streams where the transcoding includes controlling the occupancy of a buffer by feedback to DCT coefficient quantization in order to provide the output stream in a seamless way. This technological solution of claim 4 of the ’712 patent provides an improved method of switching between encoded video streams that is “both flexible and easy to implement” and overcomes the disadvantages of the prior art. *Id.* at 1:38-40. For example, the solution of the ’712 patent allows switching from a first compressed data stream encoded at a bit rate R1 to a second compressed data stream encoded at a bit rate R2, the output stream resulting from the switch being encoded again, using the transcoding system, at a bit rate R where R may

be different from R1 and R2. *Id.* at 1:52-59. Thus, the patented solution has greater flexibility than the prior art and its "implementation will be less complex and less expensive" than the prior art in addition to being more flexible. *Id.* at 1:39-40, 1:52-59, 2:9-10, 2:33.

16. A person of ordinary skill in the art reading the '712 patent and its claims would understand that the patent's disclosure and claims are drawn to solving a specific, technical problem arising in the field of video compression. In particular, the present invention relates to the technical problem involved in switching from a first compressed data input stream to a second compressed data input stream, resulting in a compressed data output stream, and is applicable, for example, to switching and editing MPEG compressed video signals. *Id.* at 1:6-12.

17. As detailed in the specification, the invention of claim 4 of the '712 patent provides a technological solution to the specific technological problems faced by the inventor that existed at the time of the invention. First the specification describes the prior art and the drawbacks associated with the prior art:

> International patent application WO 99/05870 describes a method and device of the above kind. This patent application relates, in encoding/decoding systems, to an improved method of switching from a first encoded video sequence to a second one. In order to avoid underflow or overflow of the decoded buffer, a transcoding of the input streams is used to shift the temporal position of the switching point and to obtain at the output of the transcoders, streams containing an identical entry point and the same decoder buffer characteristics.
>
> The previously described method has several major drawbacks. According to the background art, the output bit rate of each transcoder is equal to its input bit rate, which makes the switching method not very flexible. Moreover, said method implies that the first picture of the second video sequence just after the switch will be an Intra-coded (I) picture.
>
> Finally, the solution of the background art is rather complex and costly to implement as the switching device needs two transcoders.

'712 patent at 1:15-35.

18. In light of the drawbacks with the prior art, the inventor of the ’712 patent claimed a new method where transcoding of the output stream is provided by the controlling of two input streams where the transcoding includes controlling the occupancy of a buffer by feedback to DCT coefficient quantization in order to provide the output stream in a seamless way:

> To prevent overflow or underflow of this buffer, a regulation REG is performed; the buffer occupancy is controlled by a feedback to the DCT coefficient quantization. When switching from a video sequence encoded at a bit rate R**1** to another one that has been separately encoded at a bit rate R**2**, the respective decoder buffer delays at the switching point do not match. The role of the transcoder is to compensate the difference between these buffer delays in order to provide the output stream OS in a seamless way. Furthermore, the encoded bit rate R of the output stream can be chosen by the user.

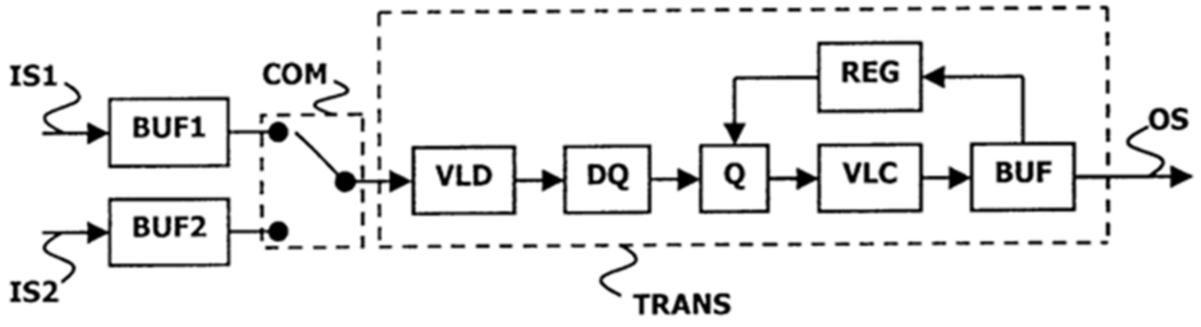


FIG. 2

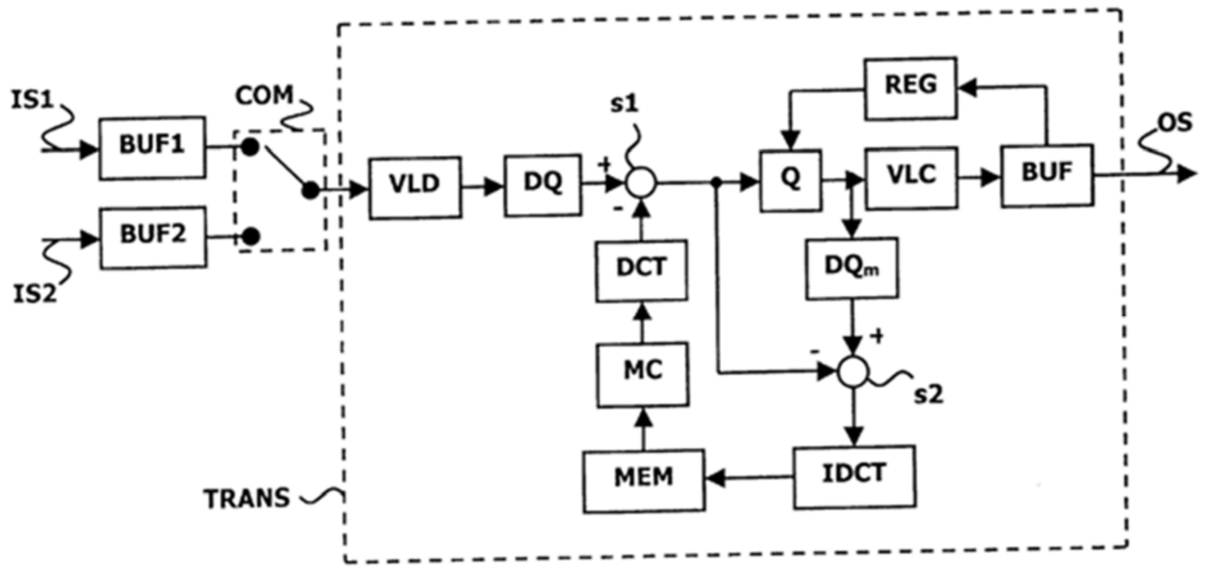


FIG. 3

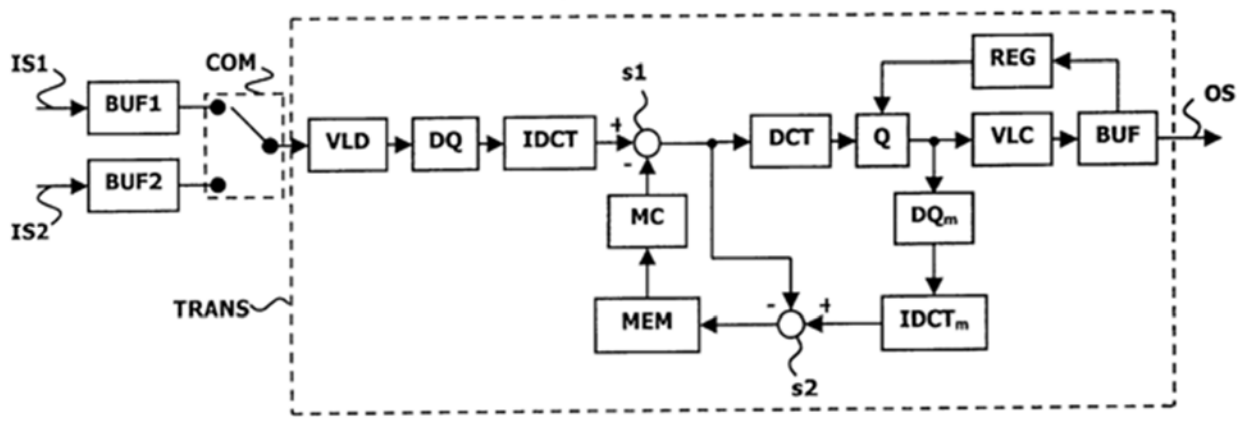


FIG. 4

’712 patent at 4:15-25, Figs. 2-4.

19. The claimed invention of claim 4 of the ’712 patent improves the functionality of switching from a first compressed data input stream to a second compressed data input stream, resulting in a compressed data output stream. ’712 patent at 1:5-2:37; 2:66-4:32. The claimed invention of claim 4 of the ’712 patent also was not well-understood, routine or conventional at

the time of invention. Rather, the claimed invention was a departure from the conventional way of switching from a first encoded video sequence to a second one.

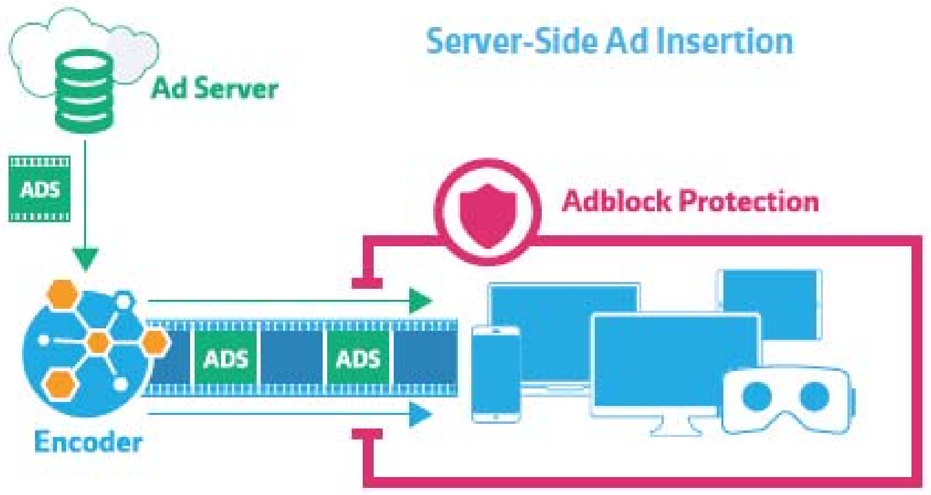
20. In light of the foregoing, a person of ordinary skill in the art would understand that the claimed subject matter of the ’712 patent presents advancements in the field of image compression. A person of ordinary skill in the art would understand that claim 4 of the ’712 patent is directed to a method of transcoding a stream provided by the controlling of two input streams where the transcoding includes controlling the occupancy of a buffer by feedback to DCT coefficient quantization in order to provide the output stream in a seamless way. Moreover, a person of ordinary skill in the art would understand that claim 4 of the ’712 patent contains the inventive concept of transcoding a stream provided by the controlling of two input streams where the transcoding includes controlling the occupancy of a buffer by feedback to DCT coefficient quantization in order to provide the output stream in a seamless way.

21. Upon information and belief, Bitmovin has directly infringed at least claim 4 of the ’712 patent by making, using, testing, selling, offering for sale, importing and/or licensing in the United States without authority products and services that dynamically switch and transcode program videos and advertisement videos (collectively “the ’712 Accused Infringing Devices”) in an exemplary manner as described below.

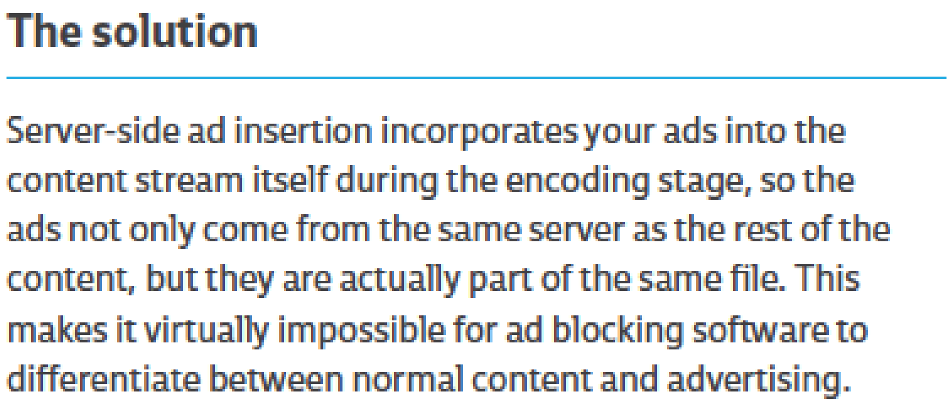
22. The ’712 Accused Infringing Devices, including a Server-Side Ad Insertion (SSAI) algorithm, practice the method of switching from a first compressed data input stream to a second compressed data input stream, resulting in a compressed data output stream.

23. The ’712 Accused Infringing Devices implement a Server-Side Ad Insertion (SSAI) algorithm that switches from the programming video to the ad video at the beginning of

an ad break and from the ad video back to the programing video at the end of an ad break. The output video is a compressed video data stream encoded in, for example, the H.264 standard.



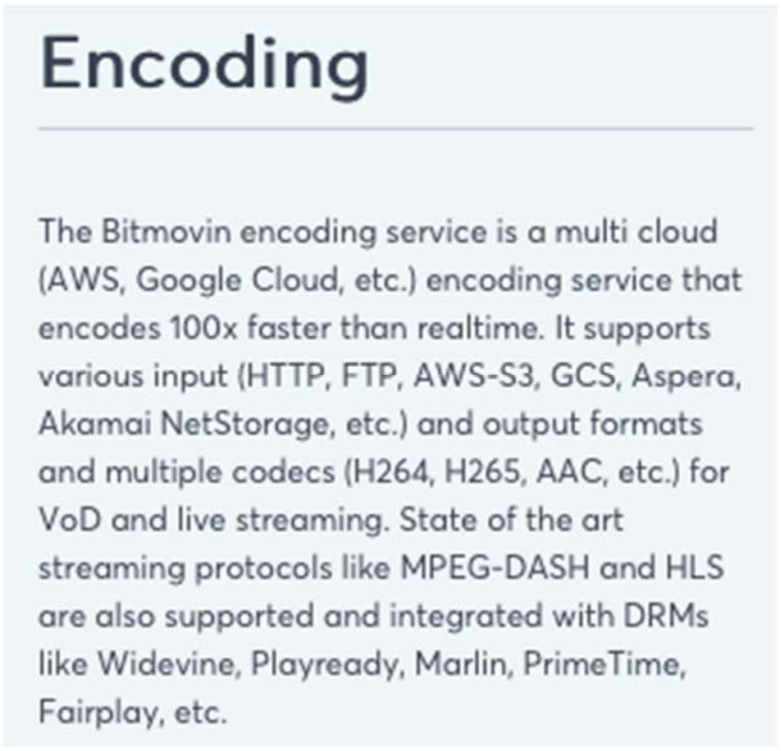
Source: https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.



The solution

Server-side ad insertion incorporates your ads into the content stream itself during the encoding stage, so the ads not only come from the same server as the rest of the content, but they are actually part of the same file. This makes it virtually impossible for ad blocking software to differentiate between normal content and advertising.

Source: https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.

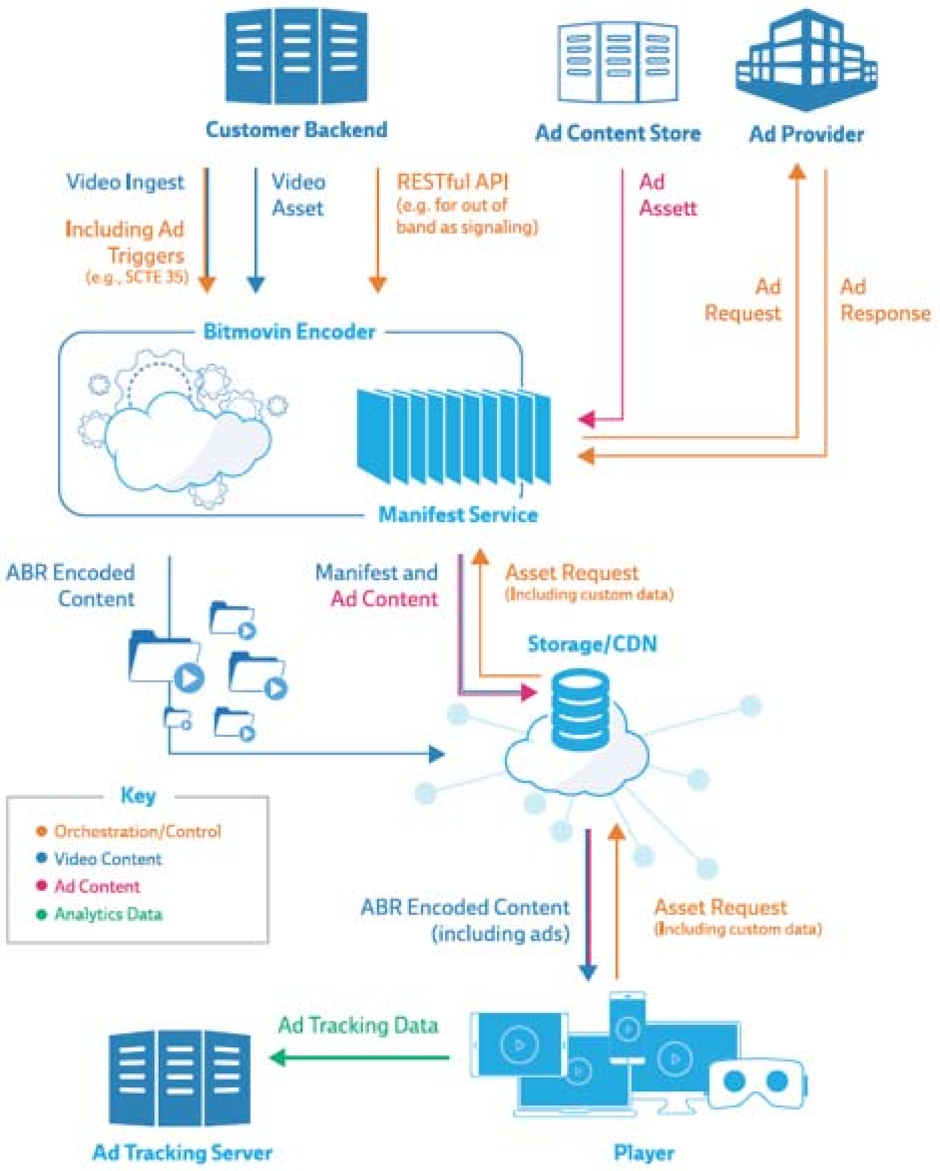


Encoding

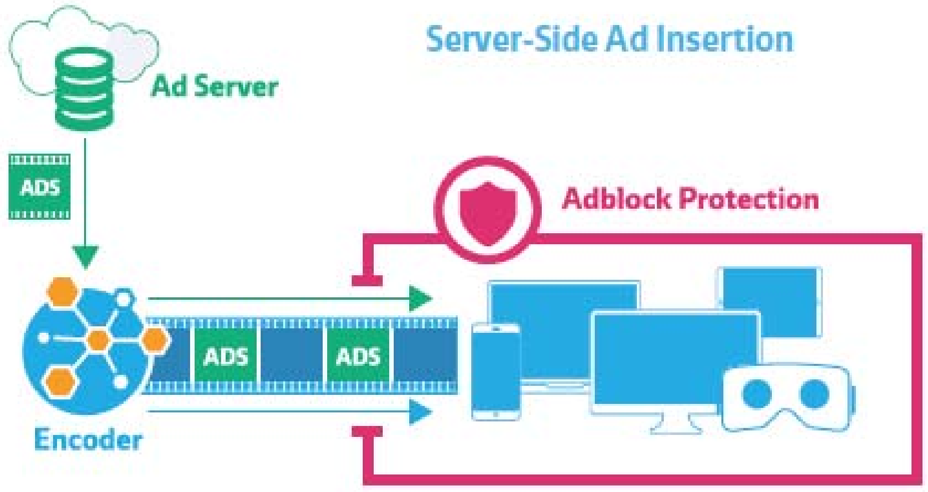
The Bitmovin encoding service is a multi cloud (AWS, Google Cloud, etc.) encoding service that encodes 100x faster than realtime. It supports various input (HTTP, FTP, AWS-S3, GCS, Aspera, Akamai NetStorage, etc.) and output formats and multiple codecs (H264, H265, AAC, etc.) for VoD and live streaming. State of the art streaming protocols like MPEG-DASH and HLS are also supported and integrated with DRMs like Widevine, Playready, Marlin, PrimeTime, Fairplay, etc.

Source: https://bitmovin.com/docs/encoding/api-reference#/reference/encoding, last accessed Nov. 29, 2018.

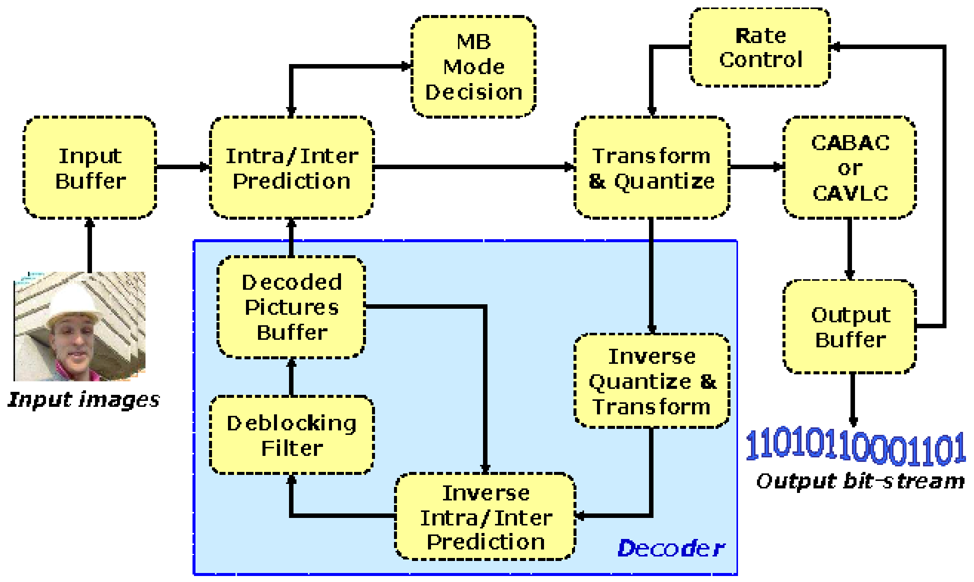
24. The ’712 Accused Infringing Devices buffer and store the data contained in the first and second input streams. The programming videos, both live and on demand, and the ad videos are buffered, in which they are also stored before they are encoded by Bitmovin’s encoding products.



Source: https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.

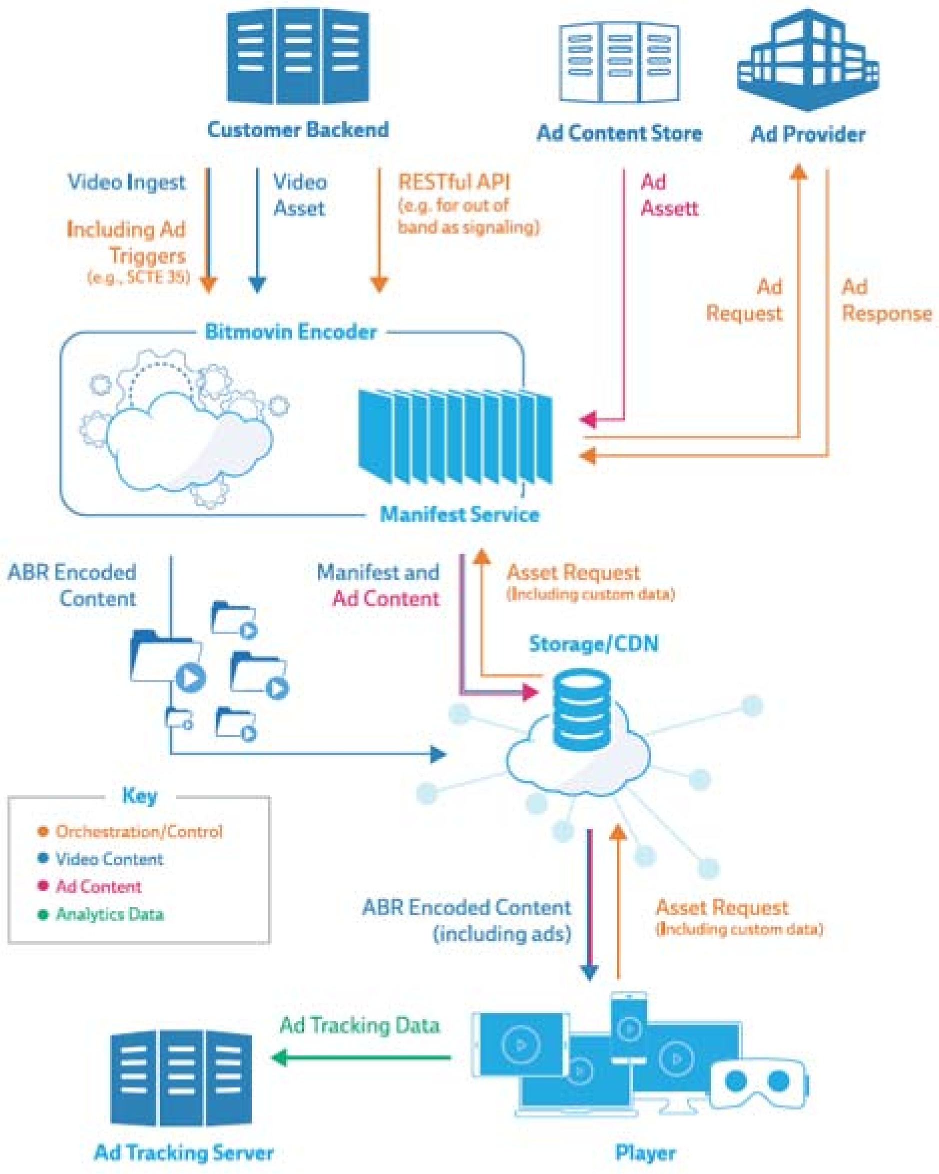


Source: https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.

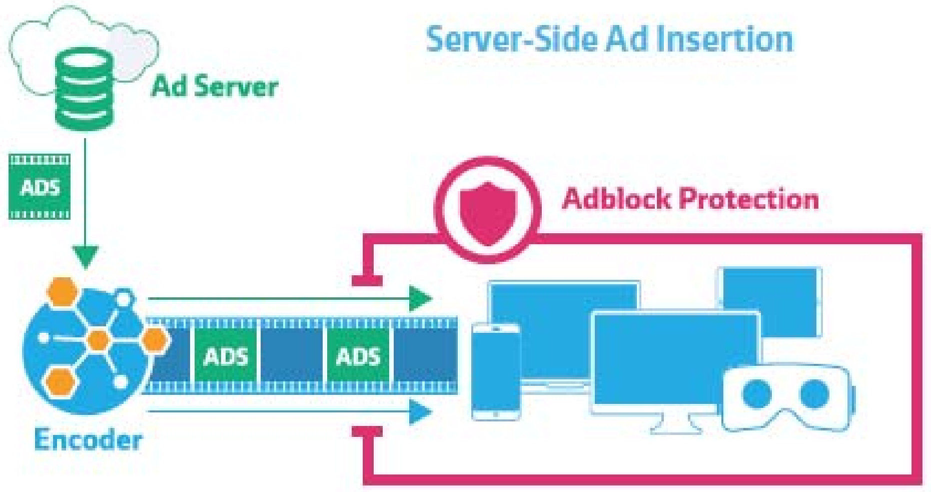


Source: https://ieeexplore.ieee.org/document/7080395, last accessed Nov. 29, 2018.

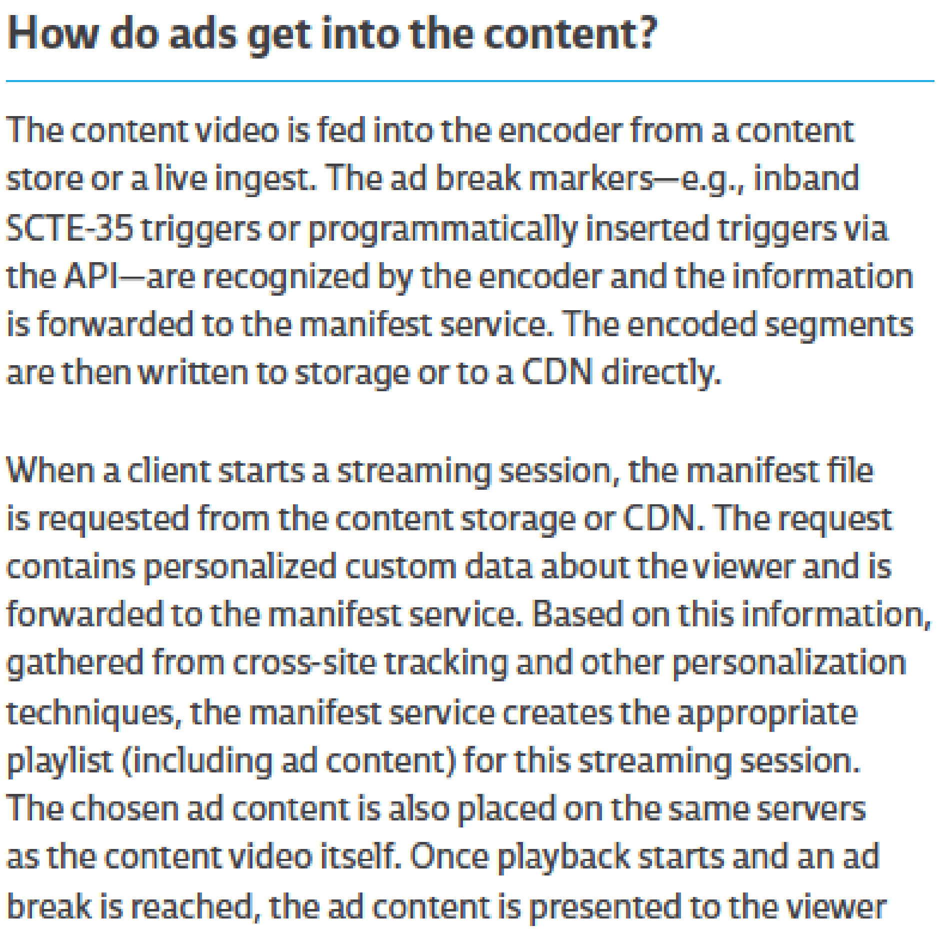
25. The ’712 Accused Infringing Devices, including a Server-Side Ad Insertion (SSAI) algorithm, control the storage of the input streams in the buffer system in order to switch, at a switch request, from the first input stream to the second input stream for its server-side dynamic ad insertion or ad stitching.



Source: https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.



Source: https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.

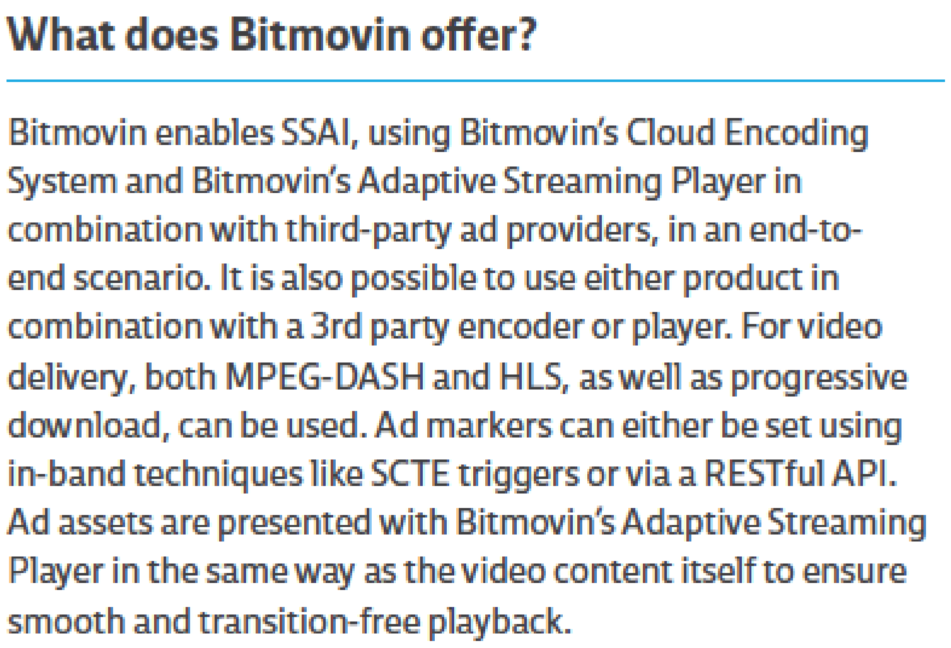


How do ads get into the content?

The content video is fed into the encoder from a content store or a live ingest. The ad break markers—e.g., inband SCTE-35 triggers or programmatically inserted triggers via the API—are recognized by the encoder and the information is forwarded to the manifest service. The encoded segments are then written to storage or to a CDN directly.

When a client starts a streaming session, the manifest file is requested from the content storage or CDN. The request contains personalized custom data about the viewer and is forwarded to the manifest service. Based on this information, gathered from cross-site tracking and other personalization techniques, the manifest service creates the appropriate playlist (including ad content) for this streaming session. The chosen ad content is also placed on the same servers as the content video itself. Once playback starts and an ad break is reached, the ad content is presented to the viewer

Source: https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.

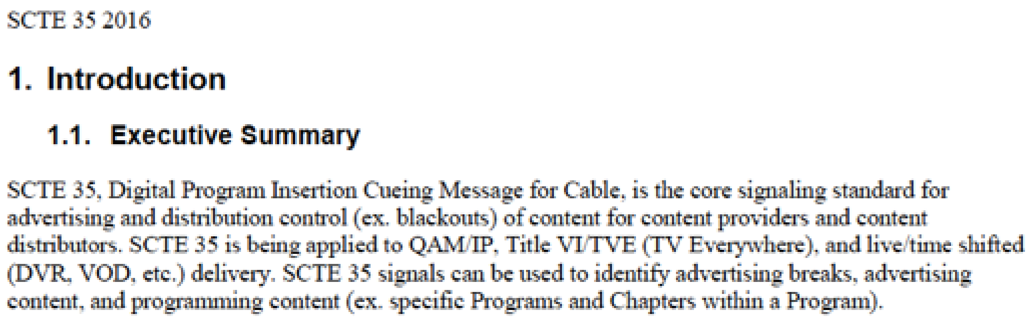


What does Bitmovin offer?

Bitmovin enables SSAI, using Bitmovin's Cloud Encoding System and Bitmovin's Adaptive Streaming Player in combination with third-party ad providers, in an end-to-end scenario. It is also possible to use either product in combination with a 3rd party encoder or player. For video delivery, both MPEG-DASH and HLS, as well as progressive download, can be used. Ad markers can either be set using in-band techniques like SCTE triggers or via a RESTful API. Ad assets are presented with Bitmovin's Adaptive Streaming Player in the same way as the video content itself to ensure smooth and transition-free playback.

Source: https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.

26. The switch request in the '712 Accused Infringing Devices, including a Server-Side Ad Insertion (SSAI) algorithm use the Society of Cable Telecommunications Engineers (SCTE) triggers for identifying an impending ad break. The Society of Cable Telecommunications Engineers standard 35 defines a family of markers (or triggers), such #EXT-X-SCTE35, #OATCLS-SCTE35, #ASSET, #CUE-OUT, #CUE-OUT-CONT, and #CUE-IN that are associated with switching between different video streams.



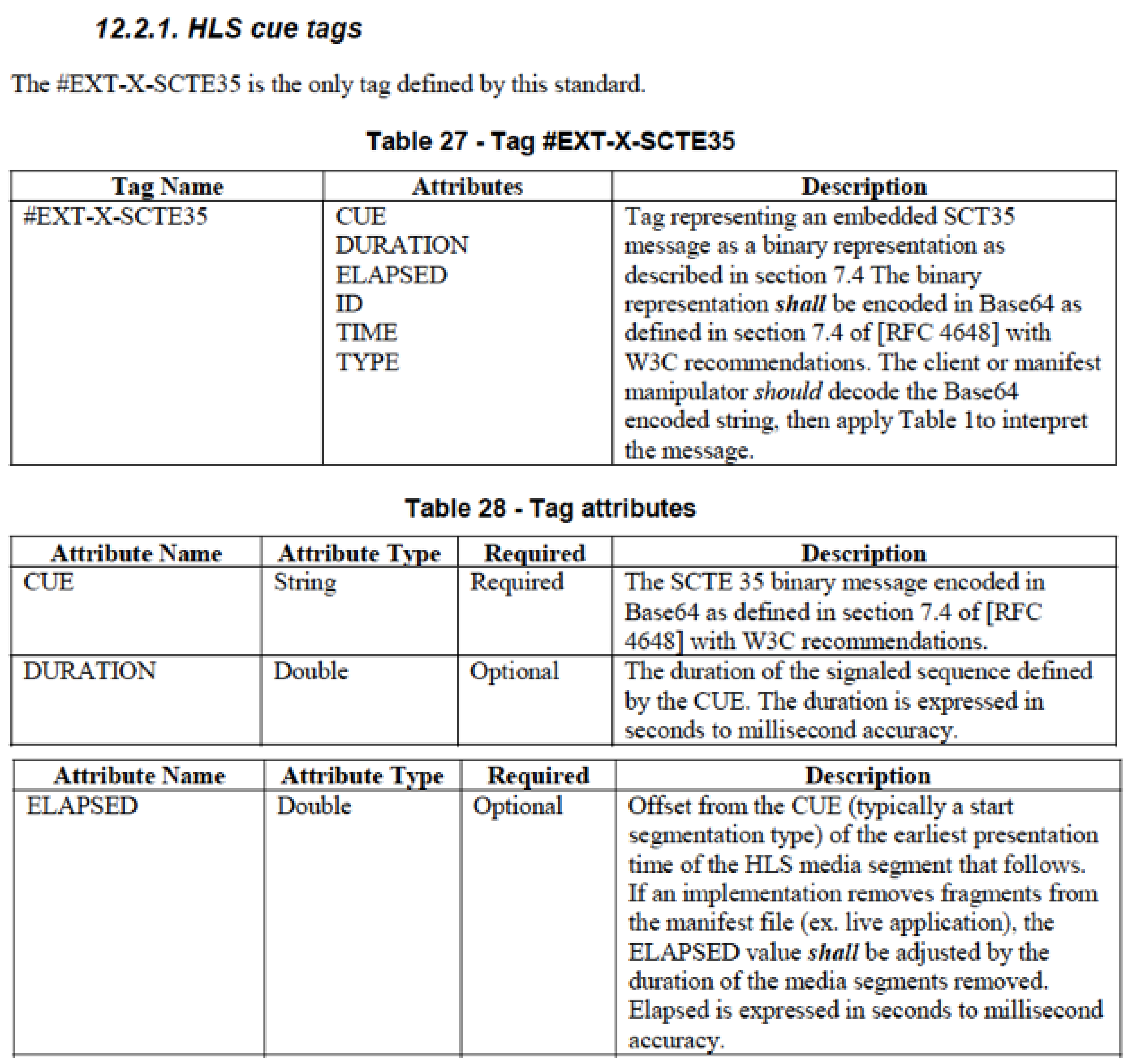
SCTE 35 2016

1. Introduction

1.1. Executive Summary

SCTE 35, Digital Program Insertion Cueing Message for Cable, is the core signaling standard for advertising and distribution control (ex. blackouts) of content for content providers and content distributors. SCTE 35 is being applied to QAM/IP, Title VI/TVE (TV Everywhere), and live/time shifted (DVR, VOD, etc.) delivery. SCTE 35 signals can be used to identify advertising breaks, advertising content, and programming content (ex. specific Programs and Chapters within a Program).

Source: https://www.scte.org/SCTEDocs/Standards/SCTE%2035%202016.pdf, Page 7, last accessed Oct. 1, 2018.



12.2.1. HLS cue tags

The #EXT-X-SCTE35 is the only tag defined by this standard.

Table 27 - Tag #EXT-X-SCTE35

Tag Name	Attributes	Description
#EXT-X-SCTE35	CUE DURATION ELAPSED ID TIME TYPE	Tag representing an embedded SCT35 message as a binary representation as described in section 7.4 The binary representation ***shall*** be encoded in Base64 as defined in section 7.4 of [RFC 4648] with W3C recommendations. The client or manifest manipulator *should* decode the Base64 encoded string, then apply Table 1to interpret the message.

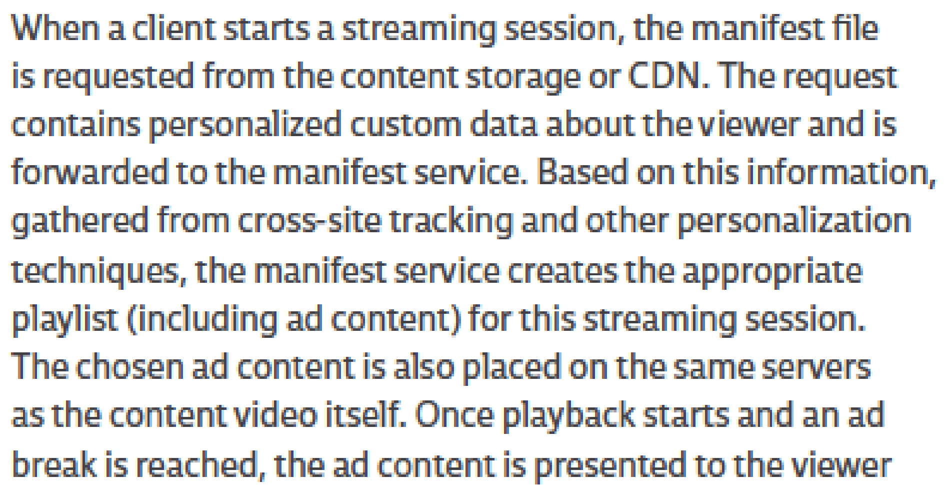
Table 28 - Tag attributes

Attribute Name	Attribute Type	Required	Description
CUE	String	Required	The SCTE 35 binary message encoded in Base64 as defined in section 7.4 of [RFC 4648] with W3C recommendations.
DURATION	Double	Optional	The duration of the signaled sequence defined by the CUE. The duration is expressed in seconds to millisecond accuracy.
ELAPSED	Double	Optional	Offset from the CUE (typically a start segmentation type) of the earliest presentation time of the HLS media segment that follows. If an implementation removes fragments from the manifest file (ex. live application), the ELAPSED value ***shall*** be adjusted by the duration of the media segments removed. Elapsed is expressed in seconds to millisecond accuracy.

Source: https://www.scte.org/SCTEDocs/Standards/SCTE%2035%202016.pdf, Pages 70-71, last accessed Oct. 1, 2018.

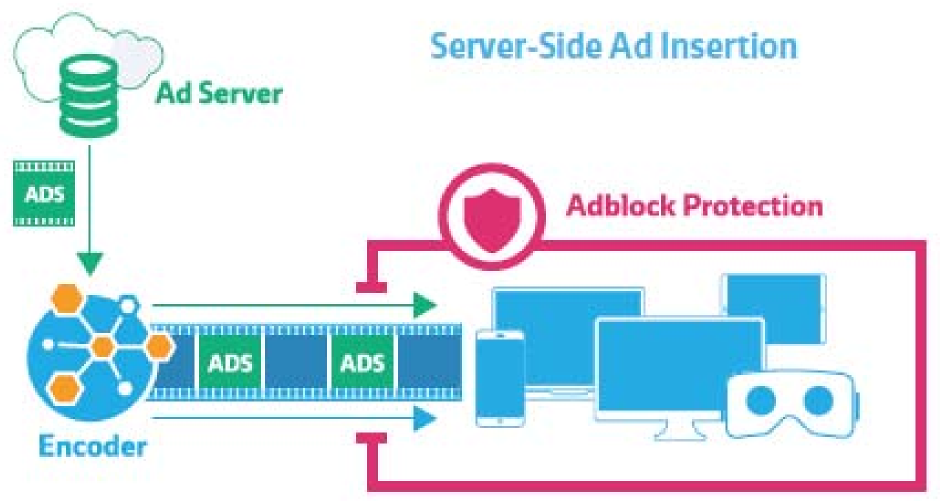
27. The ’712 Accused Infringing Devices, including a Server-Side Ad Insertion (SSAI) algorithm, provide a transcoding system (TS) including a quantization block and a buffer, wherein occupancy of the buffer in the transcoding system is controlled by feedback to the quantization block to provide the output stream in a seamless way from the output of the commutation device.

28. The ’712 Accused Infringing Devices, including a Server-Side Ad Insertion (SSAI) algorithm, transcode the compressed ad videos retrieved from 3rd party ad servers so that the ad break and ad content are “presented to the viewer in a smooth and transition-free manner.”



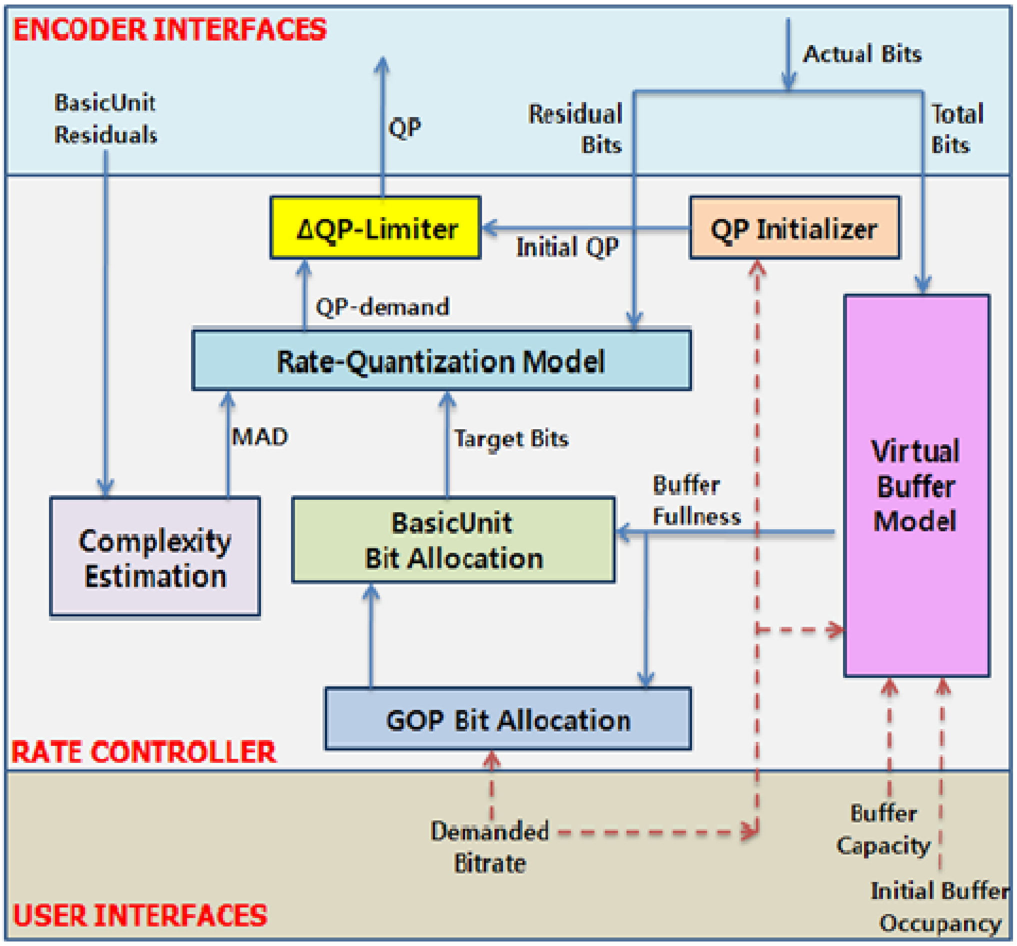
When a client starts a streaming session, the manifest file is requested from the content storage or CDN. The request contains personalized custom data about the viewer and is forwarded to the manifest service. Based on this information, gathered from cross-site tracking and other personalization techniques, the manifest service creates the appropriate playlist (including ad content) for this streaming session. The chosen ad content is also placed on the same servers as the content video itself. Once playback starts and an ad break is reached, the ad content is presented to the viewer

Source: https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.



Source: https://bitmovin.com/server-side-ad-insertion-datasheet/, last accessed Nov. 19, 2018.

29. The H.264 video codec supported in the '712 Accused Infringing Devices controls occupancy of the encoded bit stream buffer by feedback to DCT coefficient quantization as part of rate control and rate distortion optimization in the video encoders.



Source: https://www.researchgate.net/figure/Rate-control-structure-of-H264-AVC-JM-reference-model_fig1_260585793, last accessed Oct. 1, 2018.

30. Bitmovin has thus infringed at least claim 4 of the '712 patent by making, using, testing, selling, offering for sale, importing and/or licensing the '712 Accused Infringing Devices, and operating them such that all steps of at least claim 4 are performed.

31. Bitmovin has induced infringement of least claim 4 of the '712 patent since the filing of this action on January 30, 2019. *See, e.g.*, *DermaFocus LLC v. Ulthera, Inc.*, 201 F.

Supp. 3d 465, 470-472 (D. Del. Aug. 16, 2016); *Softwview LLC v. Apple Inc.*, 2012 WL 3061027, at *7 (D. Del. July 26, 2012); *Apeldyn Corp. v. Sony Corp.*, 852 F. Supp. 2d 568, 573-74 (D. Del. 2012). Bitmovin's customers are direct infringers of claim 4 of the '712 patent when customers use the '712 Accused Infringing Devices (*i.e.*, Bitmovin's Server-Side Ad Insertion (SSAI) algorithm) as described above in connection with Bitmovin's own direct infringement. Having knowledge of its own infringement, Bitmovin has, since the filing of the complaint, knowingly induced infringement and possessed the specific intent to encourage infringement of its customers by intentionally instructing its customers to infringe through videos, demonstrations, brochures and user guides, such as those located at https://bitmovin.com/encoding-service/; https://bitmovin.com/docs/encoding/tutorials; https://bitmovin.com/docs/encoding/faqs; https://bitmovin.com/docs/encoding/api-reference; https://bitmovin.com/docs/encoding/sdks; https://bitmovin.com/docs/encoding/releases; https://bitmovin.com/server-side-ad-insertion-datasheet/; and related domains and subdomains. Bitmovin is thereby liable for infringement of the '712 patent under 35 U.S.C. § 271 (b). *See, e.g.*, *DermaFocus*, 201 F. Supp. 3d at 471 ("Service of the original complaint in 2015, of course, gave defendant actual knowledge of the '559 patent. Defendant argues that, nevertheless, the FAC contains insufficient facts relating to whether defendant has the additional knowledge that third parties (its customers) are infringing the patent. (D.I. 13 at 5) Having determined, however, that plaintiff adequately pled direct infringement, and given the information contained in the FAC regarding defendant's promotional and educational materials (D.I. 11, exs. B, C and E), as well as use of the accused Ulthera System by a local physician, it is plausible to infer that defendant knew that the intended use of the Ulthera System (for which defendant's customers received instructions) was infringing. The court finds these allegations sufficient to plead induced infringement, that is, the FAC contains facts from which it is plausible to infer that defendant knew that its conduct would induce infringement by its customers, and had the specific intent to make it so.").

32. Bitmovin is also liable for contributory infringement of least claim 4 of the '712 patent since the filing of this action on January 30, 2019 for the same reasons it is liable for induced infringement and the following reasons. The portion of the '712 Accused Infringing Devices that performs the functionality of Bitmovin's Server-Side Ad Insertion (SSAI) algorithm in the manner described above (which is herein incorporated by reference) is a component of the '712 Accused Infringing Devices and is a material part of the invention of the '712 patent. Since the filing of the complaint, Bitmovin has knowledge that this component is especially adapted for infringement of the '712 patent based on Uniloc's infringement allegations and is not a staple article suitable for substantial non-infringing use of Bitmovin's Server-Side Ad Insertion (SSAI) algorithm and necessarily infringes when used in the manner described above. *DermaFocus*, 201 F. Supp. 3d at 471-72 ("With respect to contributory infringement, the FAC alleges that defendant: (1) had (at least post-suit) knowledge of the patent; (2) is selling its Ulthera System which is especially made for infringing use; (3) had knowledge of the infringing use; (4) the Ulthera System has no substantial non-infringing use: and (5) there is direct infringement. (D.I. 111 at ¶¶ 15, 16. Such allegations have passed muster under *Twombley*, *Iqbal*, and their progeny in the past."). Bitmovin is thereby liable for infringement of the '712 patent under 35 U.S.C. § 271(c).

33. Bitmovin's acts of direct and indirect infringement have caused damage to Uniloc, and Uniloc is entitled to recover damages sustained as a result of Bitmovin's wrongful acts in an amount subject to proof at trial.

COUNT II: INFRINGEMENT OF THE '118 PATENT

34. The allegations of paragraphs 1-7 of this First Amended Complaint are incorporated by reference as though fully set forth herein.

35. The '118 patent, titled "Method Of Coding Digital Image Based on Error Concealment," issued on May 17, 2005. A copy of the '118 patent is attached as Exhibit B. The priority date for the '118 patent is March 6, 2001. The inventions of the '118 patent were developed by inventors at Koninklijke Philips Electronics N.V.

36. Pursuant to 35 U.S.C. § 282, the '118 patent is presumed valid.

37. Claim 1 of the '118 patent addresses a technological problem indigenous to coding macroblocks in a binary digital stream where certain macroblocks have been excluded.

38. Claim 1 of the '118 patent reads as follows:

> 1. A method of coding a digital image comprising macroblocks in a binary data stream, the method comprising:
>
> an estimation step, for macroblocks, of a capacity to be reconstructed via an error concealment method,
>
> a decision step for macroblocks to be excluded from the coding, a decision to exclude a macroblock from coding being made on the basis of the capacity of such macroblock to be reconstructed,
>
> characterized in that it also includes a step of inserting a resynchronization marker into the binary data stream after the exclusion of one or more macroblocks.

39. The invention of claim 1 of the '118 patent concerns a novel method for digital coding of macroblocks within a data stream.

40. Just prior to the invention of the '118 patent, in June 1999, a then novel method for coding involved the exclusion of certain macroblocks in a digital image based upon the capacity of the macroblocks to be reconstructed via error concealment ("the June 1999

Method"). '118 patent at 1:14-21. In the June 1999 Method, the excluded macroblocks were replaced with "uncoded blocks with constant blocks, black blocks for example, subsequently detected by the receiver." '118 patent at 1:21-25. Alternatively, the June 1999 Method provided for allocating bits to communicate the address of the excluded blocks in interceded macroblocks that were not excluded. '118 patent at 1:26-32.

41. Both means of replacing the excluded blocks in the June 1999 Method suffered from significant drawbacks. For example, if constant blocks or black blocks were used as replacements for the excluded macroblocks there would be "graphical errors on most receivers." '118 patent at 1:62-67. Likewise, allocating bits to communicate the address of excluded blocks gave "rise to graphical 'lag' errors of image elements if macroblocks have been excluded." '118 patent at 1:56-62.

42. As demonstrated below, the claimed invention of claim 1 of the '118 patent provides a technological solution to the problem faced by the inventors— using resynchronization markers after the exclusion of a macroblock rather than replacing macroblocks with constant blocks, black blocks or bits allocated to communicate the address of the excluded blocks. This technological solution resulted in reduction in lag and graphical errors and improved bandwidth because of a reduction in the binary data stream.

43. As detailed in the specification, the invention of claim 1 of the '118 patent provides a technological solution to the specific technological problems faced by the inventors that existed at the time of the invention. First, the specification describes the June 1999 Method and the drawbacks associated with that method.

> A coding method of such type is known from the document "Geometric-Structure-Based Error Concealment with Novel Applications in Block-Based Low-Bit-Rate Coding" by W. Zeng and B. Liu in IEEE Transactions on Circuits and Systems For Video Technology, Vol. 9, No. 4, Jun. 1999.

> That document describes exclusions of blocks belonging to macroblocks, block combination, said macroblocks being capable of being intercoded or intracoded. That document proposes harmonizing this block exclusion with video coding standards, either, in a **first solution**, by replacing uncoded blocks with constant blocks, black blocks for example, subsequently detected by the receiver, or, in a **second solution**, by modifying the word that defines which blocks are coded within a macroblock, such modification taking place at the same time as a modification of the address words of the macroblocks when all the blocks in a macroblock are excluded. A certain number of bits are allocated to communicate the address of the excluded blocks in the interceded macroblocks.

’118 patent at 1:14-31 (emphasis added).

44. Both of these means of dealing with the excluded macroblocks in the June 1999 Method were disadvantageous and suffered from serious drawbacks that thwarted the purpose of excluding macroblocks (i.e., to further compress the data stream):

> In this case it is therefore impossible to change the addresses of the macroblocks or indicate which blocks are not coded, according to the **second solution** proposed in the document cited in the foregoing. All macroblocks are thus decoded and placed sequentially, giving rise to graphical “lag” errors of image elements if macroblocks have been excluded. The **first solution** proposed in the document cited involves detection by the decoder of the constant blocks replacing the excluded blocks. No provision for such detection is made in the MPEG-4 syntax, and this will cause graphical errors on most receivers.

’118 patent at 1:56-67 (emphasis added).

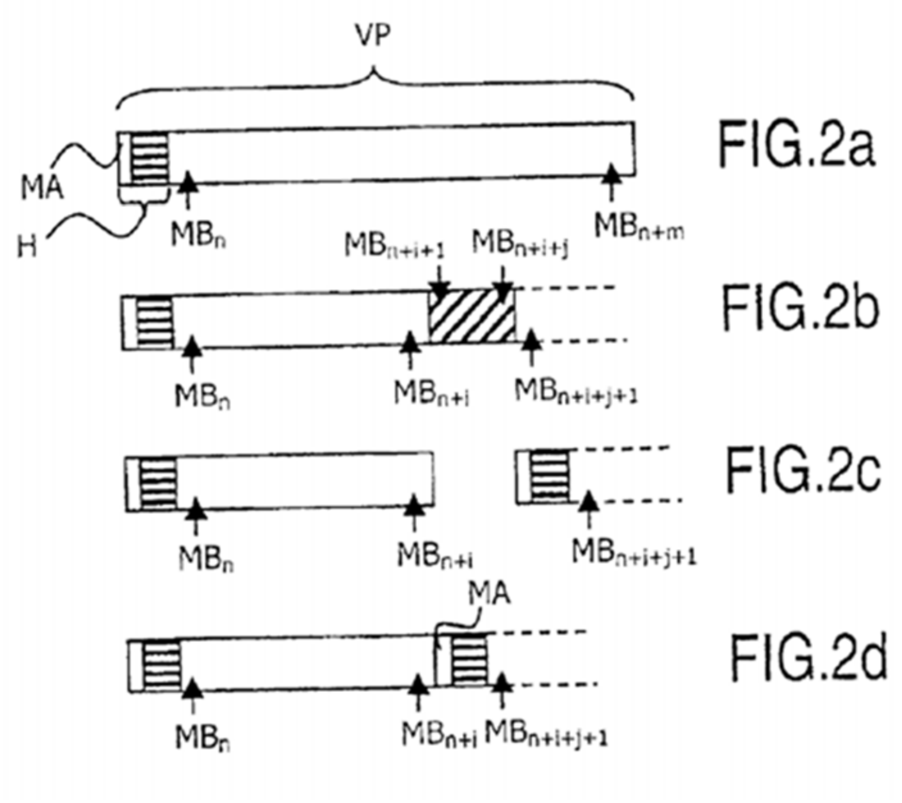
45. In light of the drawbacks with the June 1999 Method, the inventors of the ’118 patent claimed a new method where resynchronization markers included in header elements were used instead of constant blocks, black blocks and bits allocated to communicate the address of the excluded blocks:

> It is an object of the present invention to suggest a coding method that includes an exclusion of macroblocks having a certain capacity to be reconstructed from the coding compatible with coding standards which include point resynchronization means.

> Indeed, a coding method as defined in the introductory paragraph is characterized according to the invention in that it also includes a step of inserting a resynchronization marker into the binary data stream after the exclusion of one or more macroblocks.
>
> The resynchronization marker represents a certain number of bits in the data stream (at least between 17 and 23 bits). It is a further object of the present invention to reduce the binary data stream associated with the transmission of digital images by excluding macroblocks**.**

’118 patent at 2:1-15 (emphasis added).

46. The reduction in the data stream using the claimed method—as opposed to the June 1999 Method which added constant blocks, black blocks and other bits for excluded macroblocks—is depicted in Figure 2 and described in the specification:



> The resulting binary data stream in such case is shown in FIG. 2*d*. A resynchronization marker MA and the associated header element have been inserted in the stream at the point where the first one of the excluded macroblocks should have been, and before macroblock $MB_{n+i+j+1}$. Here, the reduction in the size of the binary data stream caused by the insertion of resynchronization marker MA and the associated header element is not zero according to FIG. **2**: the bloc representing excluded macroblocks MB_{n+i+1} to MB_{n+i+j} is larger than the size of the

> inserted header element.
> * * *
> Since the binary data stream includes coded data of a digital image comprising macroblocks, said binary data stream being such that macroblocks MB_{n+i+1} to MB_{n+i+j} are not coded in the binary data stream for at least one point in the binary data stream and since such uncoded macroblocks are capable of being reconstructed by an error concealment method, said binary data stream is thus characterized according to the invention in that a resynchronization marker MA is present in the binary data stream at the location in the binary data stream where the macroblocks are not coded.

’118 patent at 5:37-46.

47. The claimed invention of claim 1 of the ’118 patent improves the functionality of coding macroblocks in a binary digital stream where certain macroblocks have been excluded. The claimed invention of claim 1 of the ’118 patent also was not well-understood, routine or conventional at the time of invention. Rather, the claimed invention was a departure from the conventional way of performing coding macroblocks in a binary digital stream where certain macroblocks have been excluded.

48. A person of ordinary skill in the art reading claim 1 of the ’118 patent and the corresponding specification would understand that claim 1 improves the functionality of coding macroblocks in a binary digital stream where certain macroblocks have been excluded. This is because, as noted above, the June 1999 Method suffered from drawbacks including (1) lag errors; (2) graphical errors; and (3) no reduction in the size of the data stream because of the use of constant blocks, black blocks and allocating bits to communicate the address of the excluded blocks. A person of ordinary skill in the art would further understand that the claimed invention of claim 1 of the ’118 patent resolved these problems by using resynchronization markers in a way they had not been used before—as replacements for excluded blocks.

49. A person of ordinary skill in the art reading claim 1 of the ’118 patent and the corresponding specification would further understand that claim 1 of the ’118 patent represents a

departure from convention by (1) coding a data stream with excluded macroblocks in a way that is different from the recent June 1999 Method and (2) using resynchronization markers in a manner that had not been used before—as replacements for excluded macroblocks.

50. In light of the foregoing, a person of ordinary skill in the art reading the ’118 patent and its claims would understand that the patent’s disclosure and claims are drawn to solving a specific, technical problem arising in achieving more efficient video compression. Moreover, a person of ordinary skill in the art would understand that the claimed subject matter of the ’118 patent presents advancements in the field of digital image coding.

51. In light of the foregoing, a person of ordinary skill in the art would understand that claim 1 of the ’118 patent is directed to a method of coding macroblocks in a binary digital stream where certain macroblocks have been excluded. Moreover, a person of ordinary skill in the art would understand that claim 1 of the ’118 patent contains the inventive concept of using resynchronization markers after the exclusion of a macroblock rather than replacing macroblocks with constant blocks, black blocks or bits allocated to communicate the address of the excluded blocks.

52. Upon information and belief, Bitmovin makes, uses, offers for sale, and/or sells in the United States and/or imports into the United States products and services such as H.264 encoders that practice a method for coding a digital image comprising macroblocks in a binary data stream (collectively the “’118 Accused Infringing Devices”).

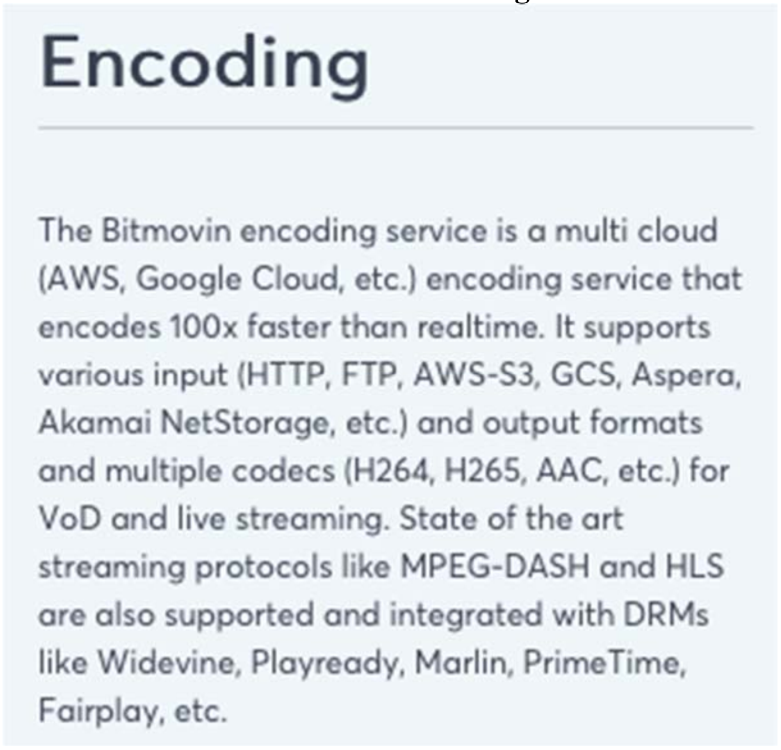
53. Upon information and belief, the ’118 Accused Infringing Devices infringe at least claim 1 in the exemplary manner described below.

54. The ’118 Accused Infringing Devices use H.264 (AVC) streams for coding video data (digital images) including macroblocks embedded in a binary stream.

55. H.264 is a widely used video compression format with decoder support on web browsers, TVs and other consumer devices. Moreover, H.264 codes digital images comprising macroblock streams.

56. The ’118 Accused Infringing Devices receive input video streams which are then encoded and/or transcoded using at least the H.264 standard. This is a widely used video compression format with decoder support on web browsers, TVs and other consumer devices. Moreover, H.264 uses motion compressor and estimator for motion coding video streams.

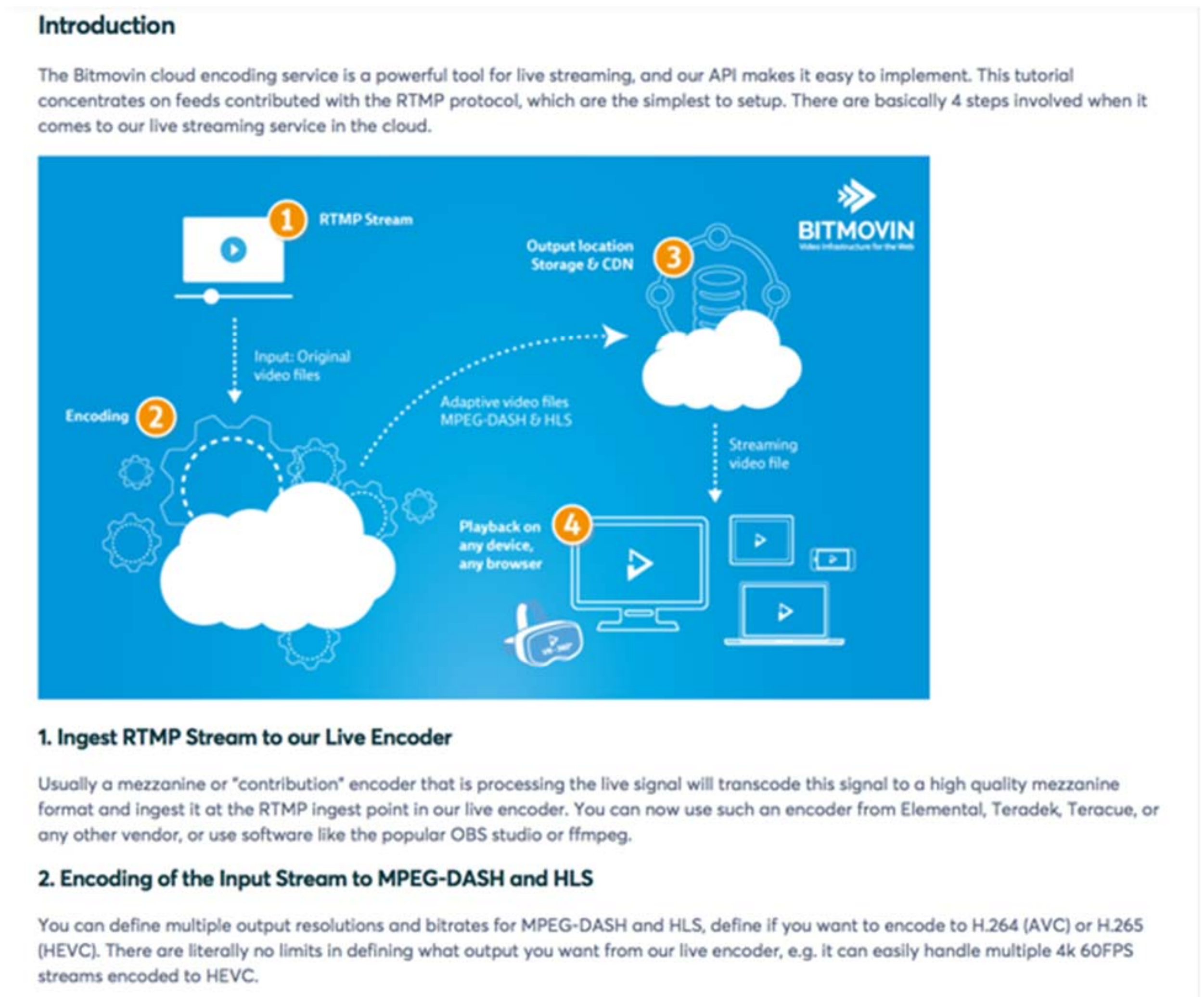
Bitmovin encodes video streams using H.264 encoders



Encoding

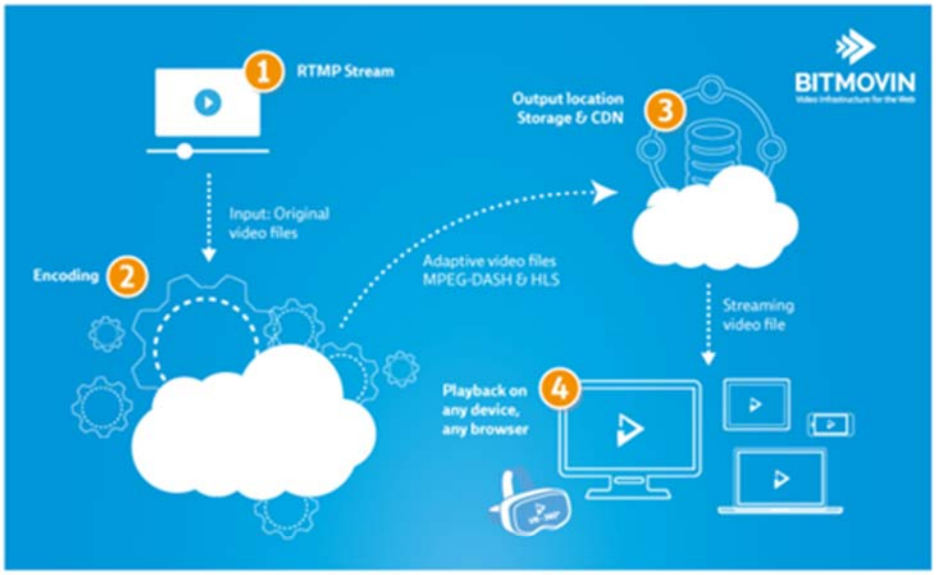
The Bitmovin encoding service is a multi cloud (AWS, Google Cloud, etc.) encoding service that encodes 100x faster than realtime. It supports various input (HTTP, FTP, AWS-S3, GCS, Aspera, Akamai NetStorage, etc.) and output formats and multiple codecs (H264, H265, AAC, etc.) for VoD and live streaming. State of the art streaming protocols like MPEG-DASH and HLS are also supported and integrated with DRMs like Widevine, Playready, Marlin, PrimeTime, Fairplay, etc.

Source: https://bitmovin.com/docs/encoding/api-reference#/reference/encoding, last accessed Nov. 29, 2018.



Introduction

The Bitmovin cloud encoding service is a powerful tool for live streaming, and our API makes it easy to implement. This tutorial concentrates on feeds contributed with the RTMP protocol, which are the simplest to setup. There are basically 4 steps involved when it comes to our live streaming service in the cloud.


1 RTMP Stream
Output location Storage & CDN 3
BITMOVIN
Input: Original video files
Encoding 2
Adaptive video files MPEG-DASH & HLS
Streaming video file
Playback on any device, any browser 4

1. Ingest RTMP Stream to our Live Encoder

Usually a mezzanine or "contribution" encoder that is processing the live signal will transcode this signal to a high quality mezzanine format and ingest it at the RTMP ingest point in our live encoder. You can now use such an encoder from Elemental, Teradek, Teracue, or any other vendor, or use software like the popular OBS studio or ffmpeg.

2. Encoding of the Input Stream to MPEG-DASH and HLS

You can define multiple output resolutions and bitrates for MPEG-DASH and HLS, define if you want to encode to H.264 (AVC) or H.265 (HEVC). There are literally no limits in defining what output you want from our live encoder, e.g. it can easily handle multiple 4k 60FPS streams encoded to HEVC.

Source: https://bitmovin.com/docs/encoding/quickstarts/create-a-live-encoding-from-an-rtmp-stream

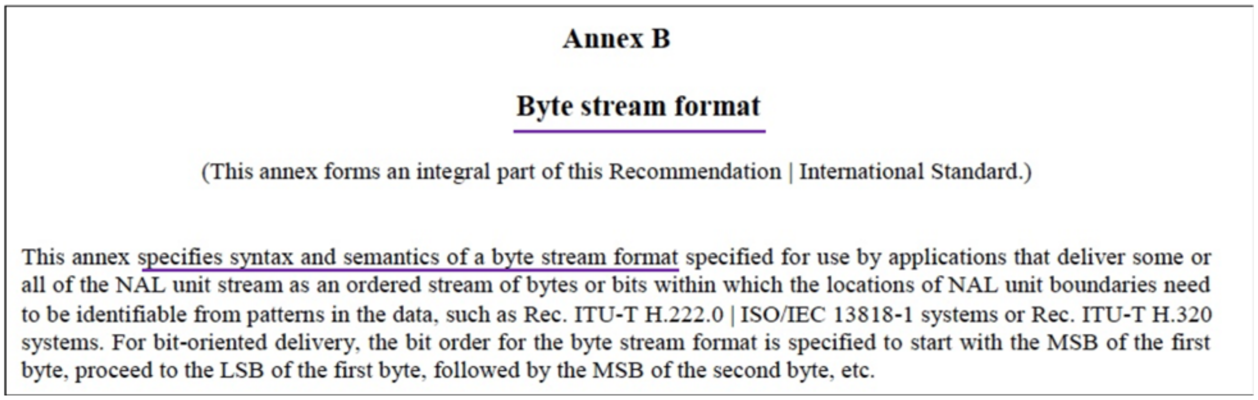
This Recommendation | International Standard was developed in response to the growing need for higher compression of moving pictures for various applications such as videoconferencing, digital storage media, television broadcasting, internet streaming, and communication. It is also designed to enable the use of the coded video representation in a flexible manner for a wide variety of network environments. The use of this Recommendation | International Standard allows motion video to be manipulated as a form of computer data and to be stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcasting channels.

Source: https://www.itu.int/rec/T-REC-H.264-201704-I/en , p. i

As in previous video coding Recommendations and International Standards, a macroblock, consisting of a 16x16 block of luma samples and two corresponding blocks of chroma samples, is used as the basic processing unit of the video decoding process.

A macroblock can be further partitioned for inter prediction. The selection of the size of inter prediction partitions is a result of a trade-off between the coding gain provided by using motion compensation with smaller blocks and the quantity

Source: https://www.itu.int/rec/T-REC-H.264-201704-I/en, section 0.6.3



Annex B

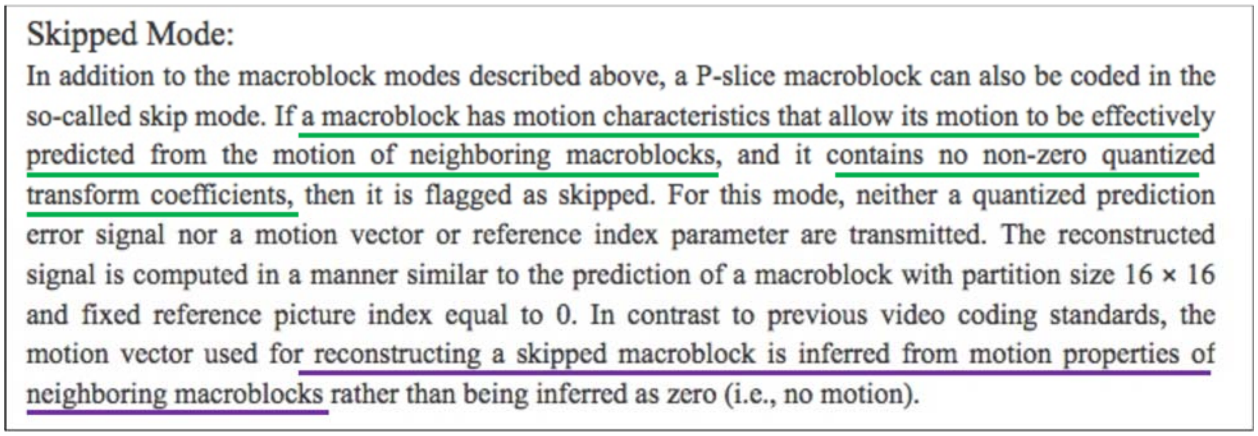
Byte stream format

(This annex forms an integral part of this Recommendation | International Standard.)

This annex specifies syntax and semantics of a byte stream format specified for use by applications that deliver some or all of the NAL unit stream as an ordered stream of bytes or bits within which the locations of NAL unit boundaries need to be identifiable from patterns in the data, such as Rec. ITU-T H.222.0 | ISO/IEC 13818-1 systems or Rec. ITU-T H.320 systems. For bit-oriented delivery, the bit order for the byte stream format is specified to start with the MSB of the first byte, proceed to the LSB of the first byte, followed by the MSB of the second byte, etc.

Source: https://www.itu.int/rec/T-REC-H.264-201704-I/en, Annex B

57. H.264 coding in the '118 Accused Infringing Devices supports skipped macroblocks. Before a macroblock is coded, an estimation is made of whether that macroblock can be reconstructed with an error concealment method by examining its motion characteristics, and checking to see that the resulting prediction contains no non-zero (i.e. all zero) quantized transform coefficients. This estimation provides an indication of the capacity for the macroblock to be reconstructed from properties of neighboring macroblocks, allowing the missing block to be concealed by inferring its properties.

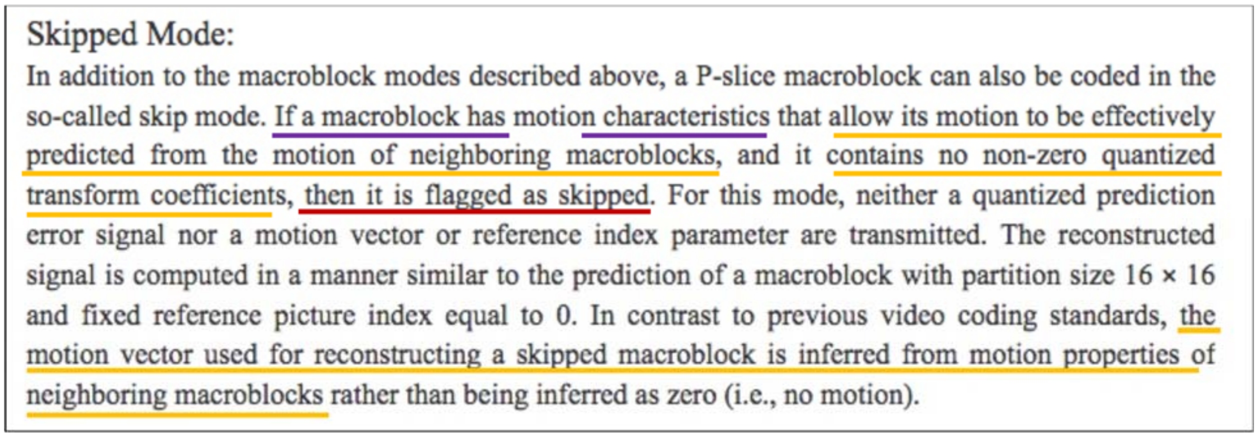


Skipped Mode:

In addition to the macroblock modes described above, a P-slice macroblock can also be coded in the so-called skip mode. If a macroblock has motion characteristics that allow its motion to be effectively predicted from the motion of neighboring macroblocks, and it contains no non-zero quantized transform coefficients, then it is flagged as skipped. For this mode, neither a quantized prediction error signal nor a motion vector or reference index parameter are transmitted. The reconstructed signal is computed in a manner similar to the prediction of a macroblock with partition size 16 × 16 and fixed reference picture index equal to 0. In contrast to previous video coding standards, the motion vector used for reconstructing a skipped macroblock is inferred from motion properties of neighboring macroblocks rather than being inferred as zero (i.e., no motion).

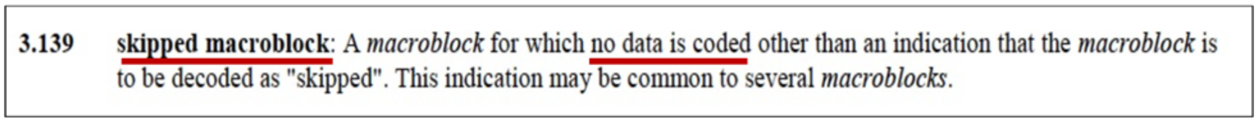
Source: http://mrutyunjayahiremath.blogspot.com/2010/09/h264-inter-predn.html

58. H.264 encoders in the ’118 Accused Infringing Devices perform a decision step to determine if a macroblock should be excluded from coding (skipped), with the decision to exclude made on the basis of its capacity to be reconstructing by inferring its motion properties from neighboring macroblocks, and based on all zero quantized transform coefficients.

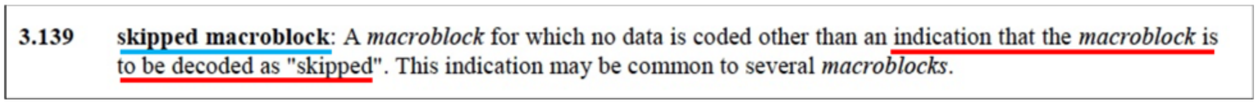

Skipped Mode:
In addition to the macroblock modes described above, a P-slice macroblock can also be coded in the so-called skip mode. If a macroblock has motion characteristics that allow its motion to be effectively predicted from the motion of neighboring macroblocks, and it contains no non-zero quantized transform coefficients, then it is flagged as skipped. For this mode, neither a quantized prediction error signal nor a motion vector or reference index parameter are transmitted. The reconstructed signal is computed in a manner similar to the prediction of a macroblock with partition size 16 × 16 and fixed reference picture index equal to 0. In contrast to previous video coding standards, the motion vector used for reconstructing a skipped macroblock is inferred from motion properties of neighboring macroblocks rather than being inferred as zero (i.e., no motion).

Source: http://mrutyunjayahiremath.blogspot.com/2010/09/h264-inter-predn.html

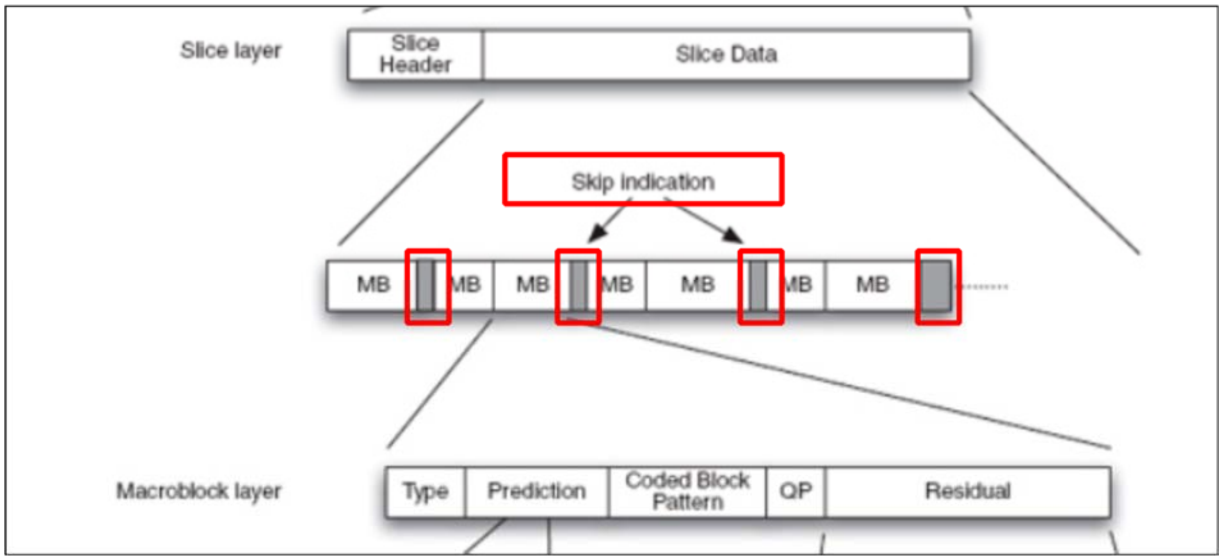
59. Skipped macroblocks are communicated with a mb_skip_flag = 1 (resynchronization marker at the point where the macroblocks are not coded (skipped)) in the binary data stream.


3.139 **skipped macroblock**: A *macroblock* for which no data is coded other than an indication that the *macroblock* is to be decoded as "skipped". This indication may be common to several *macroblocks*.

Source: https://www.itu.int/rec/T-REC-H.264-201704-I/en, p13


3.139 **skipped macroblock**: A *macroblock* for which no data is coded other than an indication that the *macroblock* is to be decoded as "skipped". This indication may be common to several *macroblocks*.

Source: https://www.itu.int/rec/T-REC-H.264-201704-I/en, p13



Source: https://www.safaribooksonline.com/library/view/the-h264 advanced/9780470516928/ch05.html#macroblock_layer

60. Bitmovin has thus infringed at least claim 1 of the '118 patent by making, using, testing, selling, offering for sale, importing and/or licensing the '118 Accused Infringing Devices, and operating them such that all steps of at least claim 1 are performed.

61. Bitmovin's acts of direct infringement have caused damage to Uniloc, and Uniloc is entitled to recover damages sustained as a result of Bitmovin's wrongful acts in an amount subject to proof at trial.

COUNT III: INFRINGEMENT OF THE '005 PATENT

62. The allegations of paragraphs 1-7 of this First Amended Complaint are incorporated by reference as though fully set forth herein.

63. The '005 patent, titled "Method of Concurrent Multiple-Mode Motion Estimation For Digital Video," issued on February 11, 2003. A copy of the '005 patent is attached as Exhibit C. The priority date for '005 patent is April 30, 1999. The inventions of the '005 patent were developed by inventors at Koninklijke Philips Electronics N.V.

64. Pursuant to 35 U.S.C. § 282, the '005 patent is presumed valid.

65. Claim 1 of the ’005 patent addresses a technological problem indigenous to motion coding in uncompressed digital video streams.

66. Claim 1 of the ’005 patent reads as follows:

> 1. A method for motion coding an uncompressed digital video data stream, including the steps of:
>
> comparing pixels of a first pixel array in a picture currently being coded with pixels of a plurality of second pixel arrays in at least one reference picture and concurrently performing motion estimation for each of a plurality of different prediction modes in order to determine which of the prediction modes is an optimum prediction mode;
>
> determining which of the second pixel arrays constitutes a best match with respect to the first pixel array for the optimum prediction mode; and,
>
> generating a motion vector for the first pixel array in response to the determining step.

67. The invention of claim 1 of the ’005 patent concerns “digital video compression” and, more particularly, “a motion estimation method and search engine for a digital video encoder that is simpler, faster, and less expensive than the presently available technology permits, and that permits concurrent motion estimation using multiple prediction modes.” ’005 patent at 1:6-11.

68. Data compression is the encoding of data using fewer “bits” than the original representation. Data compression is useful because it reduces the resources required to store and transmit data, and allows for faster retrieval and transmission of video data.

69. In the context of digital video with which the ’005 patent is concerned, a video codec is electronic circuitry or software that compresses and/or decompresses digital video for storage and/or transmission. Video codecs refer to video encoders and decoders.

70. Prior to digital video, video was typically stored as an analog signal on magnetic tape. Then, around the time of the development of compact discs (CDs), it became more feasible

to store and convey video in digital form. However, a large amount of storage and communications bandwidth was needed to record and convey raw video. Thus, what was needed was a method to reduce the amount of data used to represent the raw video. Accordingly, numerous engineers and many companies worked to develop solutions for compressing digital video data.

71. "Practical digital video compression started with the ITU H.261 standard in 1990." *A Brief History of Video Coding*, ARC International, Marco Jacobs and Jonah Probell (2007). Numerous other video compression standards thereafter were created and evolved. The innovation in digital video compression continues to this day.

72. In April 1999, at the time of the invention of claim 1 of the '005 patent, "different compression algorithms ha[d] been developed for digitally encoding video and audio information (hereinafter referred to generically as the 'digital video data stream') in order to minimize the bandwidth required to transmit this digital video data stream for a given picture quality." '005 patent at 1:11-17.

73. At the time of the invention of claim 1 of the '005 patent, the "most widely accepted international standards [for compression of digital video for motion pictures and television were] proposed by the Moving Pictures Expert Group (MPEG)." '005 patent at 1:20-24. Two such standards that existed at the time of the invention were MPEG-1 and MPEG-2.

74. In accordance with MPEG-1 and MPEG-2—and other compression standards for digital video—the video stream is "encoded/compressed . . . using a compression technique generally known as 'motion coding.'" '005 patent at 1:40-44. More particularly, rather than transmitting each video frame in its entirety, the standards at the time used motion estimation for

only those parts of sequential pictures that varied due to motion, where possible. ’005 patent at 1:45-48.

75. In general, the picture elements or “pixels” within a block of a picture are specified relative to those of a previously transmitted reference or “anchor” picture using differential or “residual” video, as well as so-called “motion vectors” that specify the location of an array (e.g., 16-by-16) of pixels or “macroblock” within the current picture relative to its original location within the anchor picture. ’005 patent at 1:48-55. A macroblock is a unit in image and video compression that typically consists of 16x16 samples of pixels. A motion vector is used to represent a macroblock in a picture based on the position of that same or similar macroblock in another picture (known as the reference picture).

76. At the time of the invention, there were various “prediction modes” that could be used for each macroblock that was to be encoded. ’005 patent at 3:7-11. Prediction modes are techniques for predicting image pixels or groups of pixels, and examples of prediction modes in MPEG include frame and field prediction modes. ’005 patent at 4:64-67. Moreover, at that time, motion coding allowed for the use of different prediction modes within the same frame, but required one prediction mode to be specified for a macroblock in advance of performing the motion estimation that results in a motion vector. ’005 patent at 3:12-15. Given that there are multiple prediction modes, the optimum prediction mode could not be known prior to encoding unless multiple motion estimations were performed on each macroblock sequentially. ’005 patent at 3:15-20. Then, after determining the optimum prediction mode based on multiple and sequential motion estimations, the optimal prediction mode would be selected and only then would the motion estimation that results in the generation of a motion vector occur.

77. In this prior art method, numerous and sequential motion estimations would have to run to find the optimal prediction mode. Only after these sequential motion estimations have been run and the optimal prediction mode selected could the motion estimation that results in the motion vector for the macroblock be carried out. Because "motion estimation usually consists of an exhaustive search procedure in which all 256 pixels of the two corresponding macroblocks are compared, and which is repeated for a large number of macroblocks," having to sequentially run numerous motion estimations to find the optimal prediction mode and only then performing the motion estimation using the optimal prediction mode to generate the motion vector is very computationally intensive, complex, inefficient, lengthy and cost ineffective. '005 patent at 3:20-43.

78. As demonstrated below, the claimed invention of claim 1 of the '005 patent provides a technological solution to the problem faced by the inventors, namely concurrently determining the optimal prediction mode while performing motion estimation along with generating the motion vector more simply, faster and in a less expensive way.

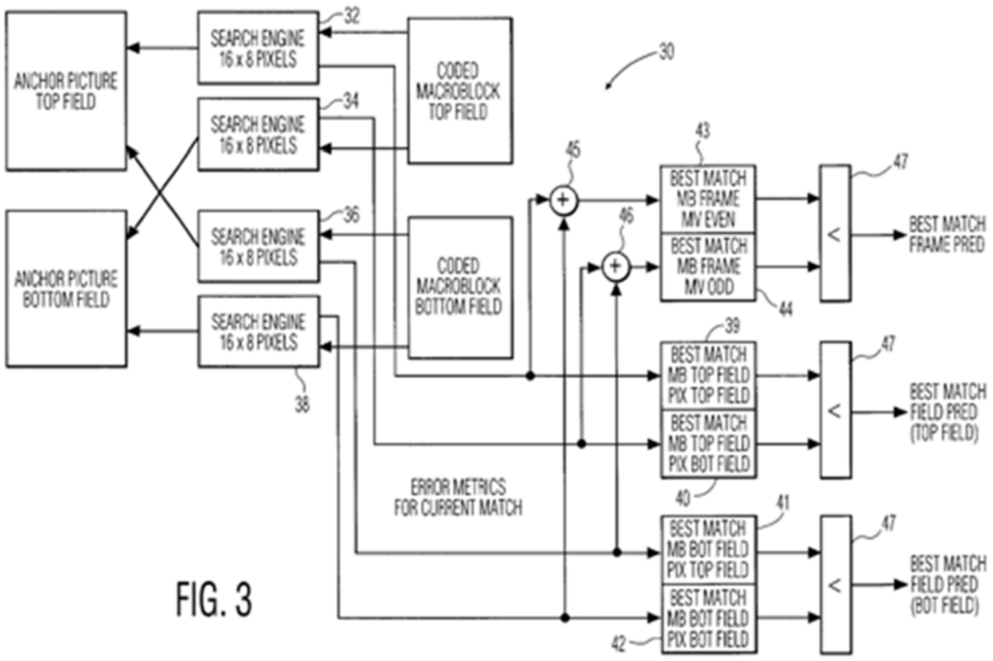
79. As detailed in the specification, the invention of claim 1 of the '005 patent provides a technological solution to the problems faced by the inventors.

> Based on the above and foregoing, it can be appreciated that there presently exists a need in the art that overcomes the disadvantages and shortcomings of the presently available technology. <u>The present invention fulfills this need in the art by performing motion coding of an uncompressed digital video sequence in such a manner that the prediction mode for each individual macroblock is determined as part of the motion estimation process, along with the actual motion vector(s), and need not be specified in advance; only the type of picture currently being coded need be known.</u> Since the latter must be determined at a higher level of video coding than the macroblock layer, this method makes possible a much more efficient, as well as optimal, degree of video compression than would otherwise be possible using conventional methods of motion estimation. Further, <u>the present invention provides a novel scheme for concurrently searching for the optimum macroblock match within the appropriate anchor picture according to each of a plurality of motion prediction</u>

> modes during the same search operation for the given macroblock, without the need for a separate search to be performed on the same macroblock for each such mode. Since this search procedure is the single most complex and expensive aspect of motion estimation, in both time and hardware, such a method as the present invention will clearly result in a more efficient video image coding and compression than would otherwise be possible given the aforementioned practical limitations of the presently available technology.

’005 patent at 3:40-67 (emphasis added).

80. The technological solution of claim 1 of the ’005 patent is further shown in Figure 3 which visually depicts a motion estimation process for concurrently performing motion estimation for frame prediction mode and field prediction modes for frame pictures:



ANCHOR PICTURE TOP FIELD
ANCHOR PICTURE BOTTOM FIELD
SEARCH ENGINE 16 x 8 PIXELS
32
SEARCH ENGINE 16 x 8 PIXELS
34
SEARCH ENGINE 16 x 8 PIXELS
36
SEARCH ENGINE 16 x 8 PIXELS
38
CODED MACROBLOCK TOP FIELD
CODED MACROBLOCK BOTTOM FIELD
30
45
46
43
BEST MATCH MB FRAME MV EVEN
BEST MATCH MB FRAME MV ODD
44
47
BEST MATCH FRAME PRED
39
BEST MATCH MB TOP FIELD PIX TOP FIELD
BEST MATCH MB TOP FIELD PIX BOT FIELD
40
47
BEST MATCH FIELD PRED (TOP FIELD)
ERROR METRICS FOR CURRENT MATCH
41
BEST MATCH MB BOT FIELD PIX TOP FIELD
BEST MATCH MB BOT FIELD PIX BOT FIELD
42
47
BEST MATCH FIELD PRED (BOT FIELD)

FIG. 3

81. Claim 1 of the ’005 patent improves the functionality of motion coding in video compression by performing the concurrent determination of the optimal prediction mode while performing motion estimation along with generating the motion vector. The claimed invention of claim 1 of the ’005 patent also was not well-understood, routine or conventional at the time of

the invention. Rather, as set forth below, the claimed invention was a departure from the conventional ways of performing motion coding in video compression.

> Based on the above and foregoing, it can be appreciated that there presently exists a need in the art that overcomes the disadvantages and shortcomings of the presently available technology. The present invention fulfills this need in the art by performing motion coding of an uncompressed digital video sequence in such a manner that the prediction mode for each individual macroblock is determined as part of the motion estimation process, along with the actual motion vector(s), and need not be specified in advance; only the type of picture currently being coded need be known. Since the latter must be determined at a higher level of video coding than the macroblock layer, this method makes possible a much more efficient, as well as optimal, degree of video compression than would otherwise be possible using conventional methods of motion estimation. Further, the present invention provides a novel scheme for concurrently searching for the optimum macroblock match within the appropriate anchor picture according to each of a plurality of motion prediction modes during the same search operation for the given macroblock, without the need for a separate search to be performed on the same macroblock for each such mode. Since this search procedure is the single most complex and expensive aspect of motion estimation, in both time and hardware, such a method as the present invention will clearly result in a more efficient video image coding and compression than would otherwise be possible given the aforementioned practical limitations of the presently available technology.

’005 patent at 3:40-67 (emphasis added).

> The present invention relates generally to digital video compression, and, more particularly, to a motion estimation method and search engine for a digital video encoder that is simpler, faster, and less expensive than the presently available technology permits, and that permits concurrent motion estimation using multiple prediction modes.

’005 patent at 1:7-11 (emphasis added).

> In either case, the methods and architectures of the present invention result in a means of significantly improving the video compression efficiency and, hence, the resulting picture quality, without the need for either greater hardware costs or higher computational complexity.

’005 patent at 14:62-67 (emphasis added).

> In all known motion estimation methods, the prediction mode must be specified for every macroblock before the motion estimation, with its constituent search, is performed. However, in accordance with the present invention, in one of its

> aspects, the motion estimation may be performed, in a frame picture, forth both frame and field prediction modes simultaneously, during the same search for the anchor picture.

’005 patent at 8:6-13 (emphasis added).

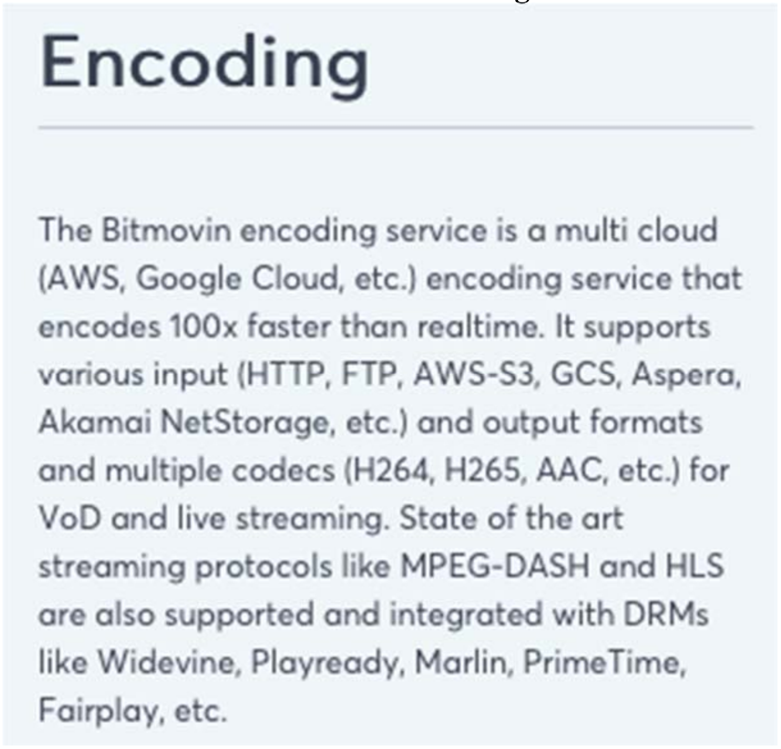
82. In light of the foregoing, and the general knowledge of a person of ordinary skill in the art, a person of ordinary skill in the art reading the ’005 patent and its claims would understand that the patent’s disclosure and claims are drawn to solving a specific, technical problem arising in the field of digital video compression. Moreover, a person of ordinary skill in the art would understand that the claimed subject matter of the ’005 patent presents advancements in the field of digital video compression, and more particularly to a motion estimation method and search engine for a digital video encoder that is simpler, faster, and less expensive than prior art technology, and that permits concurrent motion estimation using multiple prediction modes. A person of ordinary skill in the art would understand that claim 1 of the ’005 patent is directed to a method for motion coding an uncompressed digital video data stream, which provides concurrent motion estimation using multiple prediction modes along with the generation of motion vectors. Moreover, a person of ordinary skill in the art would understand that claim 1 of the ’005 patent contains that corresponding inventive concept.

83. Upon information and belief, Bitmovin makes, uses, offers for sale, and/or sells in the United States and/or imports into the United States products and services such as its H.264 encoders that practice a method for motion coding an uncompressed digital video data stream (collectively the “’005 Accused Infringing Devices”).

84. Upon information and belief, the ’005 Accused Infringing Devices infringe at least claim 1 in the exemplary manner described below.

85. The ’005 Accused Infringing Devices provide a method for motion coding an uncompressed (pixel level) digital video data stream. The ’005 Accused Infringing Devices receive input video streams which are then encoded and/or transcoded using at least the H.264 (AVC) standard. The H.264 standard is a widely used video compression format with decoder support on web browsers, TVs and other consumer devices. Moreover, H.264 uses motion compressor and estimator for motion coding video streams.

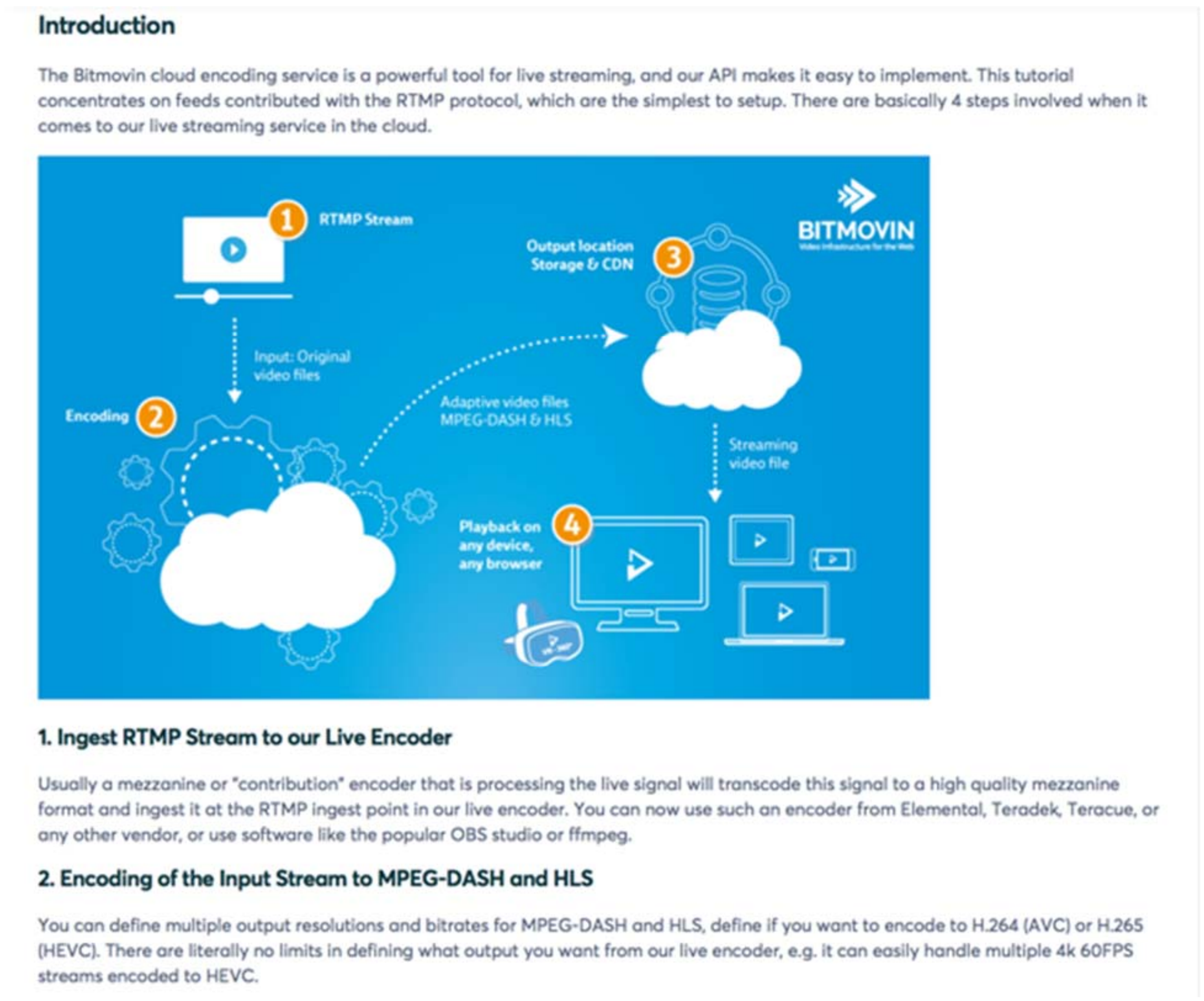
Bitmovin encodes video streams using H.264 encoders



Encoding

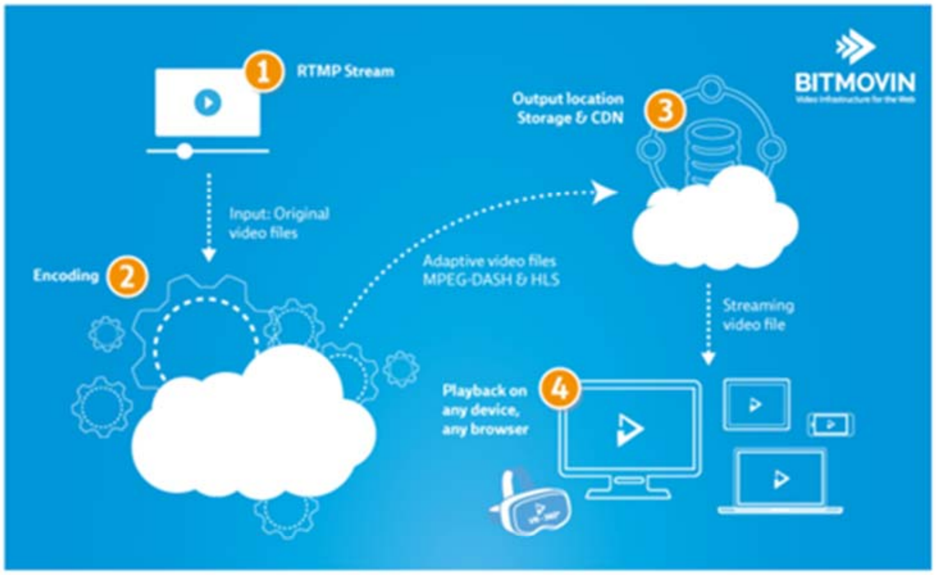
The Bitmovin encoding service is a multi cloud (AWS, Google Cloud, etc.) encoding service that encodes 100x faster than realtime. It supports various input (HTTP, FTP, AWS-S3, GCS, Aspera, Akamai NetStorage, etc.) and output formats and multiple codecs (H264, H265, AAC, etc.) for VoD and live streaming. State of the art streaming protocols like MPEG-DASH and HLS are also supported and integrated with DRMs like Widevine, Playready, Marlin, PrimeTime, Fairplay, etc.

Source: https://bitmovin.com/docs/encoding/api-reference#/reference/encoding, last accessed Nov. 29, 2018.



Introduction

The Bitmovin cloud encoding service is a powerful tool for live streaming, and our API makes it easy to implement. This tutorial concentrates on feeds contributed with the RTMP protocol, which are the simplest to setup. There are basically 4 steps involved when it comes to our live streaming service in the cloud.


BITMOVIN
1 RTMP Stream
Input: Original video files
Encoding 2
Adaptive video files MPEG-DASH & HLS
Output location Storage & CDN 3
Streaming video file
Playback on any device, any browser 4

1. Ingest RTMP Stream to our Live Encoder

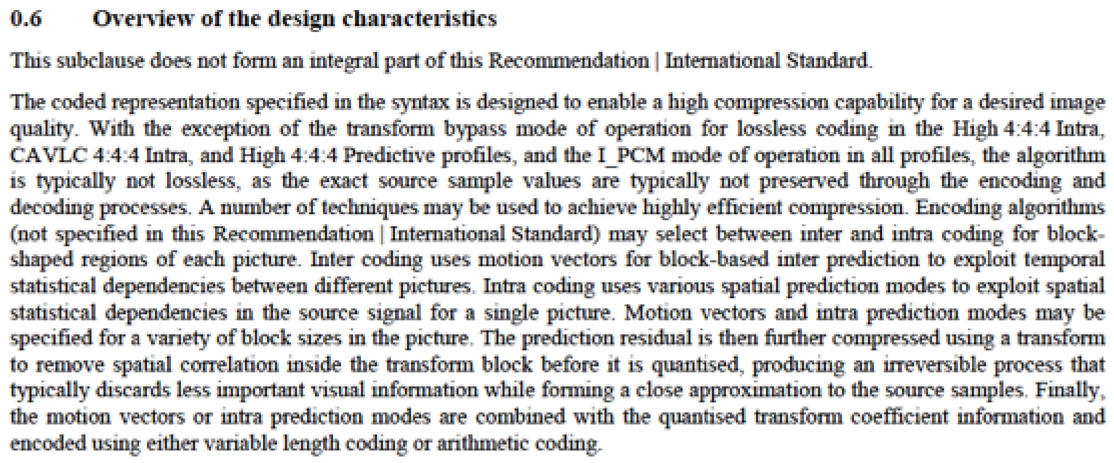
Usually a mezzanine or "contribution" encoder that is processing the live signal will transcode this signal to a high quality mezzanine format and ingest it at the RTMP ingest point in our live encoder. You can now use such an encoder from Elemental, Teradek, Teracue, or any other vendor, or use software like the popular OBS studio or ffmpeg.

2. Encoding of the Input Stream to MPEG-DASH and HLS

You can define multiple output resolutions and bitrates for MPEG-DASH and HLS, define if you want to encode to H.264 (AVC) or H.265 (HEVC). There are literally no limits in defining what output you want from our live encoder, e.g. it can easily handle multiple 4k 60FPS streams encoded to HEVC.

Source: https://bitmovin.com/docs/encoding/quickstarts/create-a-live-encoding-from-an-rtmp-stream

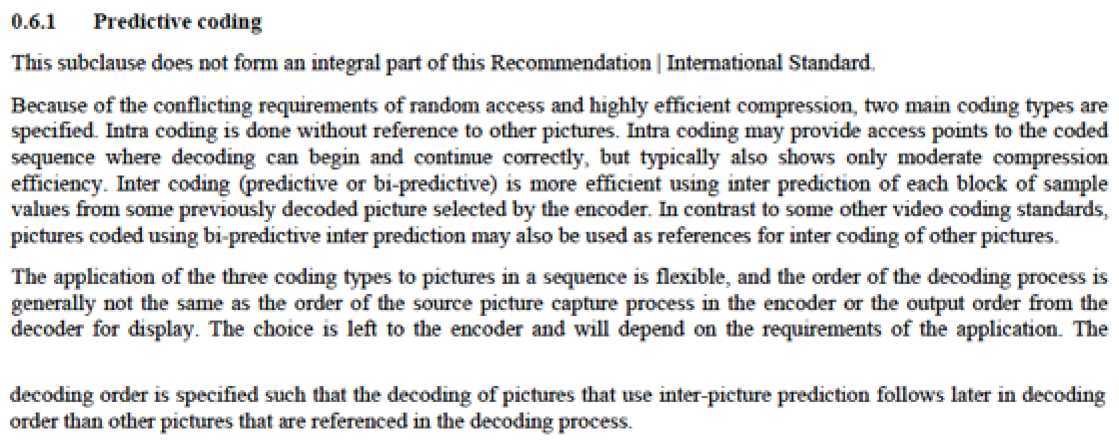
H.264 Uses Predictive Coding



0.6 Overview of the design characteristics

This subclause does not form an integral part of this Recommendation | International Standard.

The coded representation specified in the syntax is designed to enable a high compression capability for a desired image quality. With the exception of the transform bypass mode of operation for lossless coding in the High 4:4:4 Intra, CAVLC 4:4:4 Intra, and High 4:4:4 Predictive profiles, and the I_PCM mode of operation in all profiles, the algorithm is typically not lossless, as the exact source sample values are typically not preserved through the encoding and decoding processes. A number of techniques may be used to achieve highly efficient compression. Encoding algorithms (not specified in this Recommendation | International Standard) may select between inter and intra coding for block-shaped regions of each picture. Inter coding uses motion vectors for block-based inter prediction to exploit temporal statistical dependencies between different pictures. Intra coding uses various spatial prediction modes to exploit spatial statistical dependencies in the source signal for a single picture. Motion vectors and intra prediction modes may be specified for a variety of block sizes in the picture. The prediction residual is then further compressed using a transform to remove spatial correlation inside the transform block before it is quantised, producing an irreversible process that typically discards less important visual information while forming a close approximation to the source samples. Finally, the motion vectors or intra prediction modes are combined with the quantised transform coefficient information and encoded using either variable length coding or arithmetic coding.



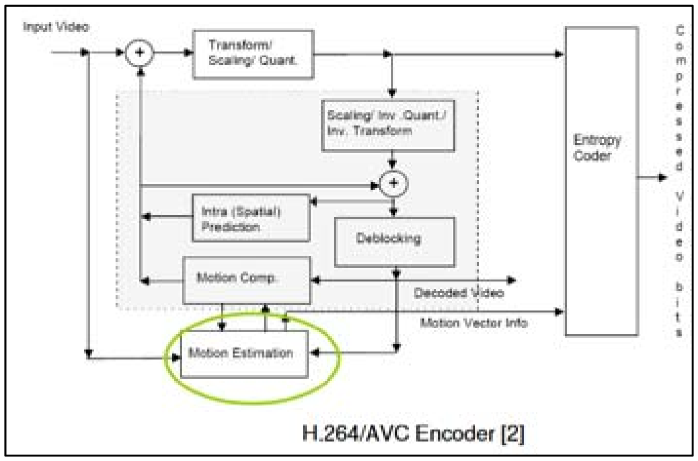
0.6.1 Predictive coding

This subclause does not form an integral part of this Recommendation | International Standard.

Because of the conflicting requirements of random access and highly efficient compression, two main coding types are specified. Intra coding is done without reference to other pictures. Intra coding may provide access points to the coded sequence where decoding can begin and continue correctly, but typically also shows only moderate compression efficiency. Inter coding (predictive or bi-predictive) is more efficient using inter prediction of each block of sample values from some previously decoded picture selected by the encoder. In contrast to some other video coding standards, pictures coded using bi-predictive inter prediction may also be used as references for inter coding of other pictures.

The application of the three coding types to pictures in a sequence is flexible, and the order of the decoding process is generally not the same as the order of the source picture capture process in the encoder or the output order from the decoder for display. The choice is left to the encoder and will depend on the requirements of the application. The decoding order is specified such that the decoding of pictures that use inter-picture prediction follows later in decoding order than other pictures that are referenced in the decoding process.

Source: H.264 Standard (03-2010) at pp. 3-4

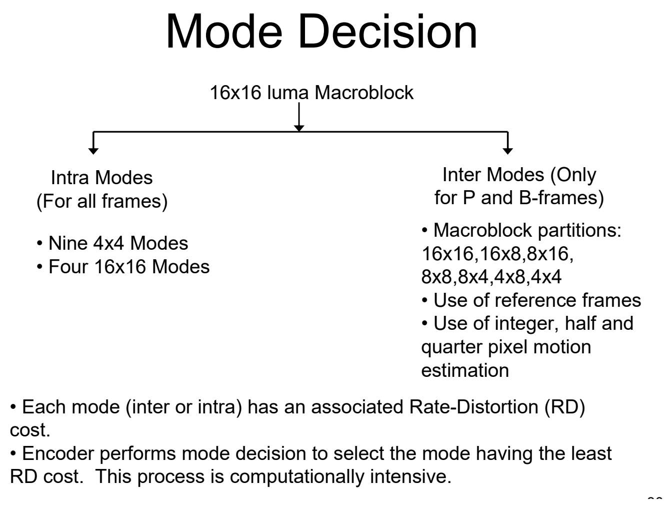


Source: https://courses.cs.washington.edu/courses/csep590a/07au/lectures/rahullarge.pdf

86. The '005 Accused Infringing Devices provide a method for comparing pixels of a first pixel array (e.g., a macroblock) in a picture currently being coded with pixels of a plurality of second pixel arrays in at least one reference picture and concurrently performing motion estimation for each of a plurality of different prediction modes in order to determine which of the prediction modes is an optimum prediction mode.

87. H.264 uses different motion estimation modes in inter-frame prediction. These modes are commonly referred to as inter-frame prediction modes, or inter modes. Each inter mode involves partitioning the current macroblock into a different combination of sub blocks,

and selecting the optimum motion vector for the current macroblock based on the partition. The inter-frame prediction modes, or inter modes, can be further categorized by the number and position of the reference frames, as well as the choice of integer pixel, half pixel and quarter pixel values in motion estimation. The Bitmovin H.264 encoders concurrently perform motion estimation of a macroblock for all inter-modes and select the most optimum prediction mode with least rate distortion cost.


Mode Decision

16x16 luma Macroblock
Intra Modes (For all frames)
Inter Modes (Only for P and B-frames)

- Nine 4x4 Modes
- Four 16x16 Modes

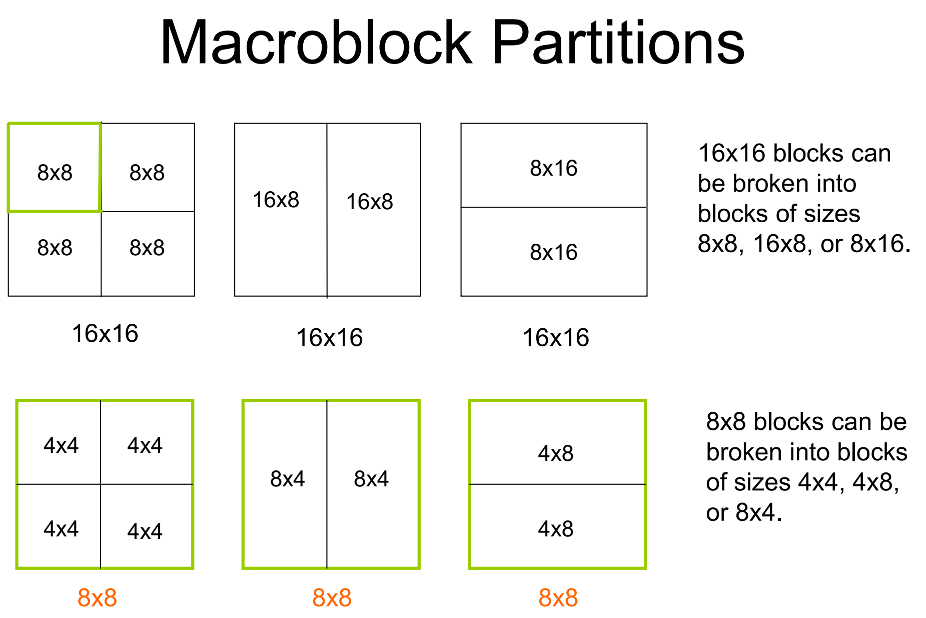
- Macroblock partitions: 16x16,16x8,8x16, 8x8,8x4,4x8,4x4
- Use of reference frames
- Use of integer, half and quarter pixel motion estimation

- Each mode (inter or intra) has an associated Rate-Distortion (RD) cost.
- Encoder performs mode decision to select the mode having the least RD cost. This process is computationally intensive.

Source: https://courses.cs.washington.edu/courses/csep590a/07au/lectures/rahullarge.pdf, p. 30

88. H.264 provides a hierarchical way to partition a macroblock, with the available partitions shown in the following two figures. An exemplary inter-frame prediction mode, or inter mode, can be for a macroblock to be partitioned to encompass a 16x8 sub block on the left, and two 8x8 sub blocks on the right.

Macroblock partitions for inter-frame prediction modes


Macroblock Partitions
8x8
8x8
8x8
8x8
16x16
16x8
16x8
16x16
8x16
8x16
16x16
16x16 blocks can be broken into blocks of sizes 8x8, 16x8, or 8x16.
4x4
4x4
4x4
4x4
8x8
8x4
8x4
8x8
4x8
4x8
8x8
8x8 blocks can be broken into blocks of sizes 4x4, 4x8, or 8x4.

Source: https://courses.cs.washington.edu/courses/csep590a/07au/lectures/rahullarge.pdf, p. 4

H.264 provides macroblock partitions for inter-frame prediction modes

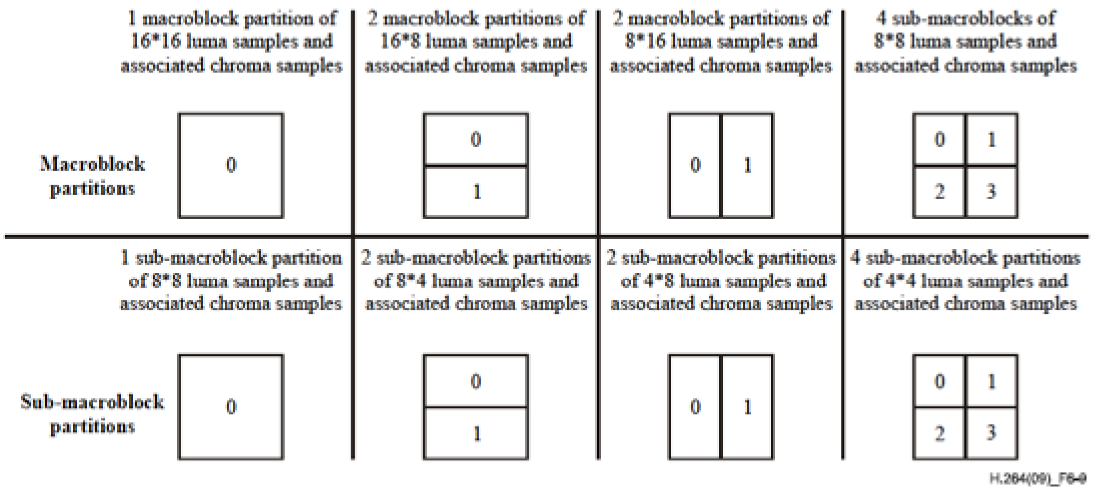
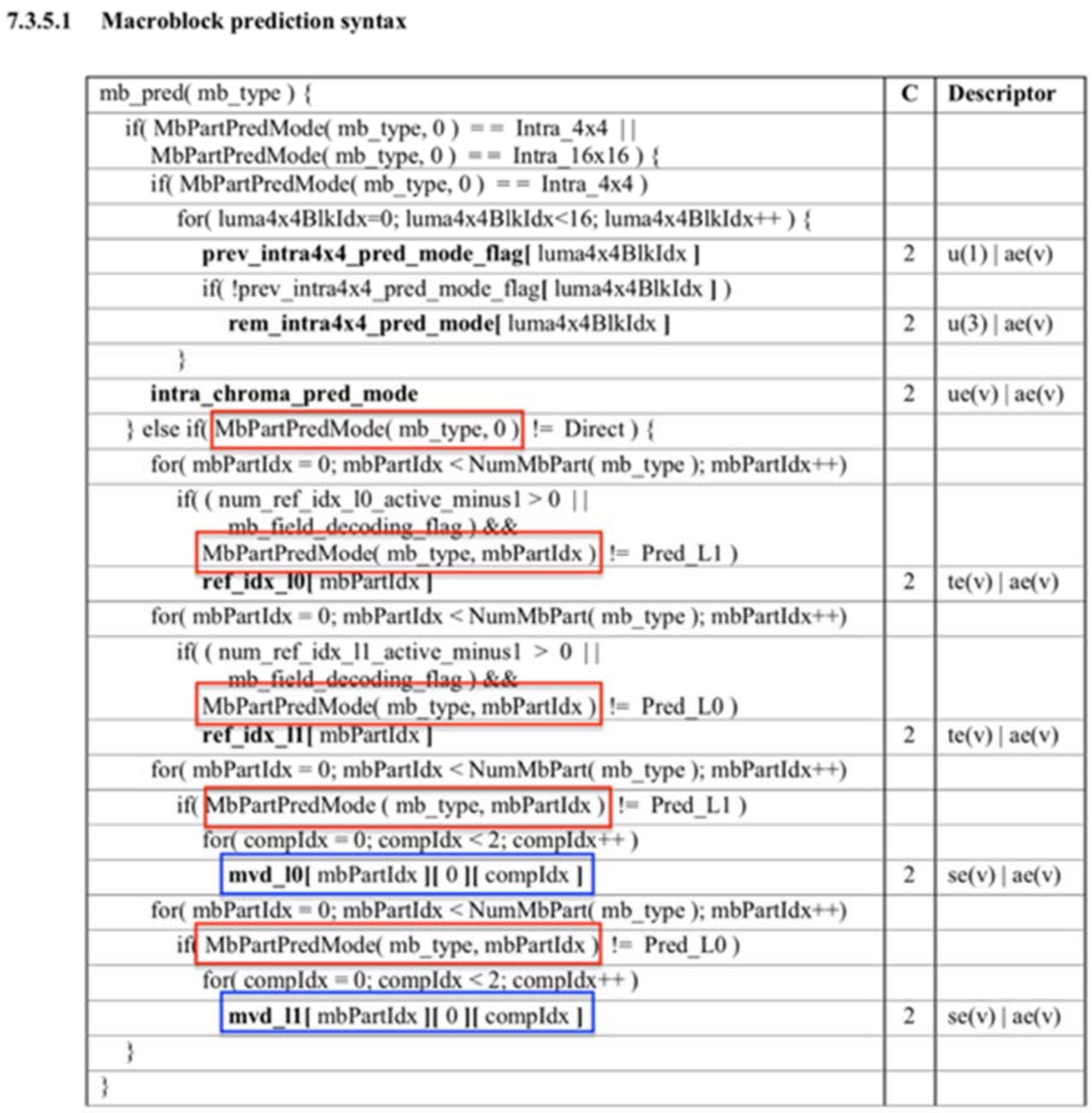

1 macroblock partition of 16*16 luma samples and associated chroma samples
2 macroblock partitions of 16*8 luma samples and associated chroma samples
2 macroblock partitions of 8*16 luma samples and associated chroma samples
4 sub-macroblocks of 8*8 luma samples and associated chroma samples
Macroblock partitions
0
0
1
0
1
0
1
2
3
1 sub-macroblock partition of 8*8 luma samples and associated chroma samples
2 sub-macroblock partitions of 8*4 luma samples and associated chroma samples
2 sub-macroblock partitions of 4*8 luma samples and associated chroma samples
4 sub-macroblock partitions of 4*4 luma samples and associated chroma samples
Sub-macroblock partitions
0
0
1
0
1
0
1
2
3
H.264(09)_F6-9

Figure 6-9 – Macroblock partitions, sub-macroblock partitions, macroblock partition scans, and sub-macroblock partition scans

Source: H.264 Standard (03-2010) at p. 26

89. The optimum prediction mode as chosen for the current macroblock is embedded in the compressed bit stream of H.264, as shown in the following two syntaxes.

Macroblock prediction syntax in H.264

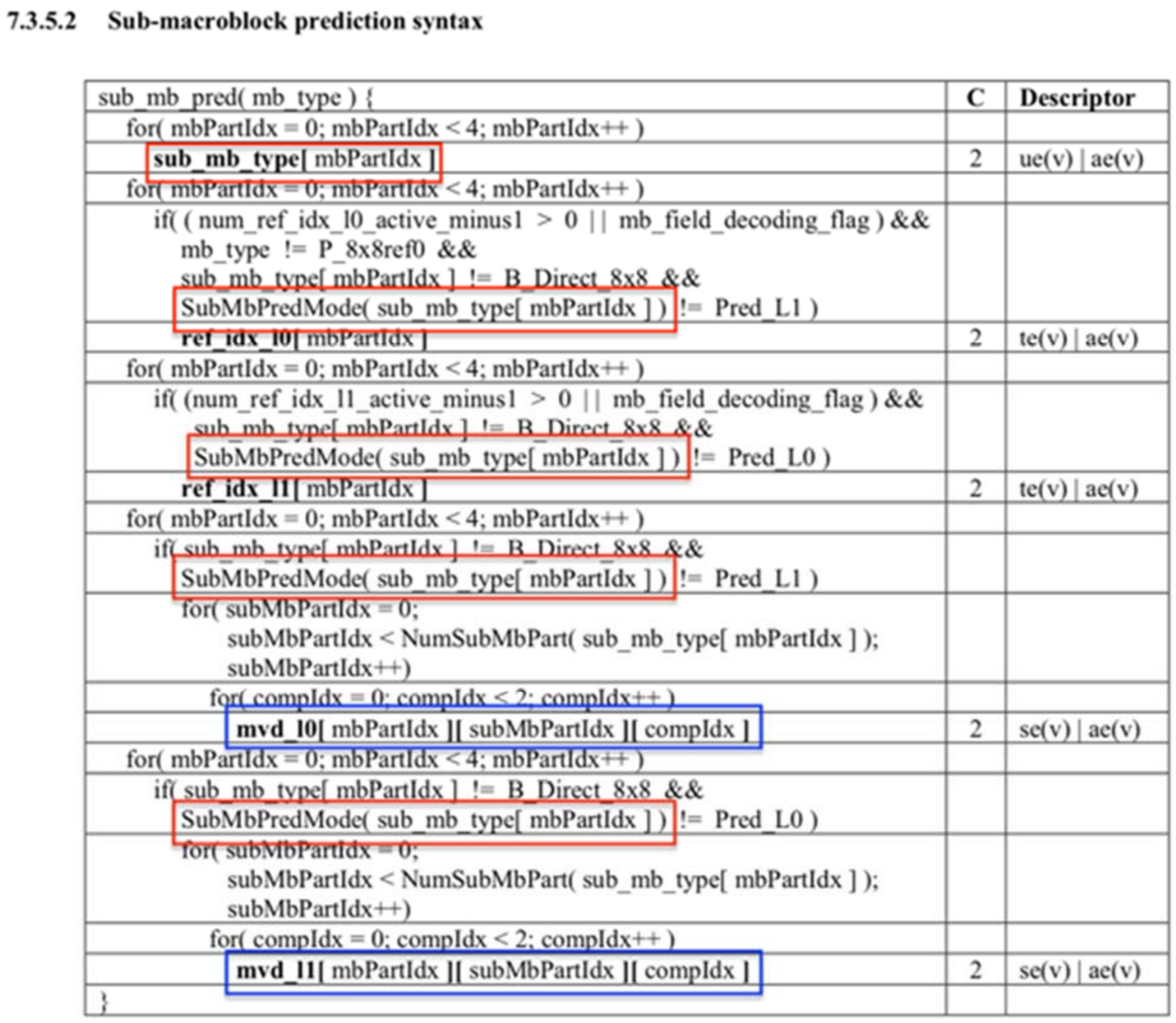


7.3.5.1 Macroblock prediction syntax

mb_pred(mb_type) {	C	Descriptor
if(MbPartPredMode(mb_type, 0) = = Intra_4x4 || MbPartPredMode(mb_type, 0) = = Intra_16x16) {		
if(MbPartPredMode(mb_type, 0) = = Intra_4x4)		
for(luma4x4BlkIdx=0; luma4x4BlkIdx<16; luma4x4BlkIdx++) {		
prev_intra4x4_pred_mode_flag[luma4x4BlkIdx **]**	2	u(1) | ae(v)
if(!prev_intra4x4_pred_mode_flag[luma4x4BlkIdx])		
rem_intra4x4_pred_mode[luma4x4BlkIdx **]**	2	u(3) | ae(v)
}		
intra_chroma_pred_mode	2	ue(v) | ae(v)
} else if(MbPartPredMode(mb_type, 0) != Direct) {		
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if((num_ref_idx_l0_active_minus1 > 0 || mb_field_decoding_flag) && MbPartPredMode(mb_type, mbPartIdx) != Pred_L1)		
ref_idx_l0[mbPartIdx **]**	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if((num_ref_idx_l1_active_minus1 > 0 || mb_field_decoding_flag) && MbPartPredMode(mb_type, mbPartIdx) != Pred_L0)		
ref_idx_l1[mbPartIdx **]**	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(MbPartPredMode (mb_type, mbPartIdx) != Pred_L1)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx **][** 0 **][** compIdx **]**	2	se(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(MbPartPredMode(mb_type, mbPartIdx) != Pred_L0)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx **][** 0 **][** compIdx **]**	2	se(v) | ae(v)
}		
}		

Source: H.264 Standard (03-2010) at p. 57

Sub-macroblock prediction syntax in H.264



7.3.5.2 Sub-macroblock prediction syntax

sub_mb_pred(mb_type) {	C	Descriptor
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
sub_mb_type[mbPartIdx]	2	ue(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if((num_ref_idx_l0_active_minus1 > 0 || mb_field_decoding_flag) && mb_type != P_8x8ref0 && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
ref_idx_l0[mbPartIdx]	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if((num_ref_idx_l1_active_minus1 > 0 || mb_field_decoding_flag) && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
ref_idx_l1[mbPartIdx]	2	te(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx][subMbPartIdx][compIdx]	2	se(v) | ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx][subMbPartIdx][compIdx]	2	se(v) | ae(v)
}		

Source: H.264 Standard (03-2010) at p. 58

90. The ’005 Accused Infringing Devices provide a method for determining which of the second pixel arrays (e.g., macroblock) constitutes a best match with respect to the first pixel array (e.g., macroblock) for the optimum prediction mode.

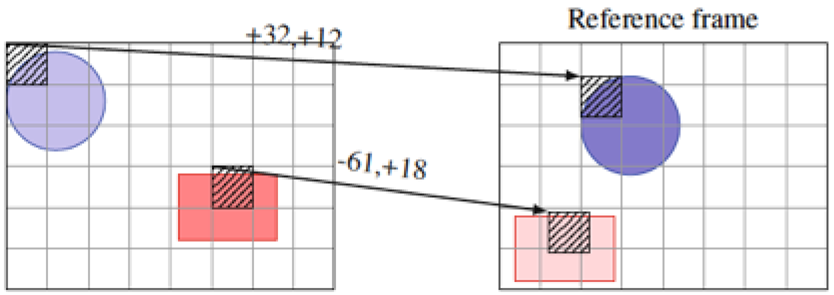
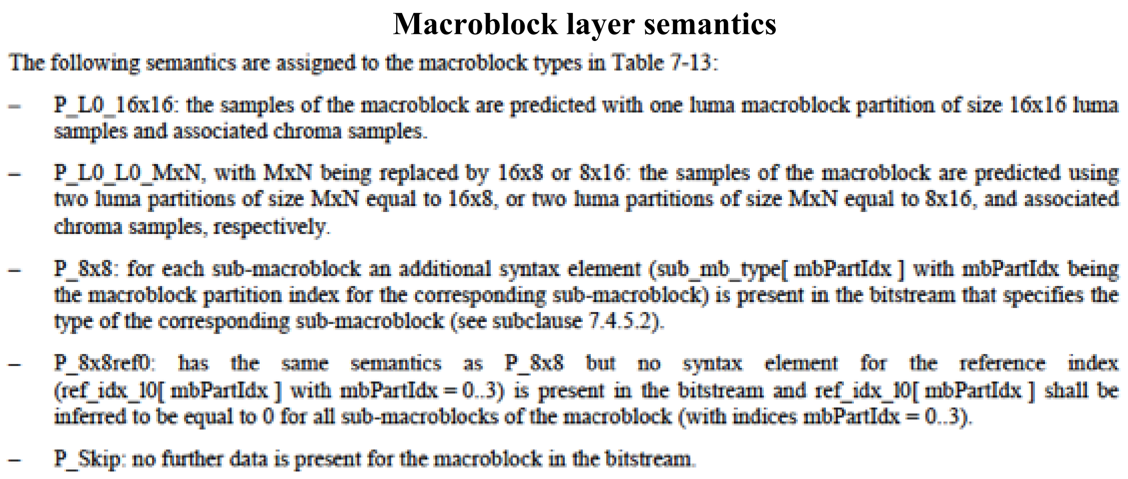


Fig. 2.4: Motion estimation. For each MB the best matching block in the reference frame is found. The encoder codes the differences (errors) between the MBs and their best matching blocks. Arrows indicate motion vectors and are labeled by the vector coordinates. In this example the shapes are identical but their colors are slightly larger/darker.

Source: B. Juurlink et al., Scalable Parallel Programming Applied to H.264, Chapter 2: Understanding the Application: An Overview of the H.264 Standard, p. 12

91. For example, the encoder performs mode decision to select the most optimum prediction mode with least rate distortion cost.



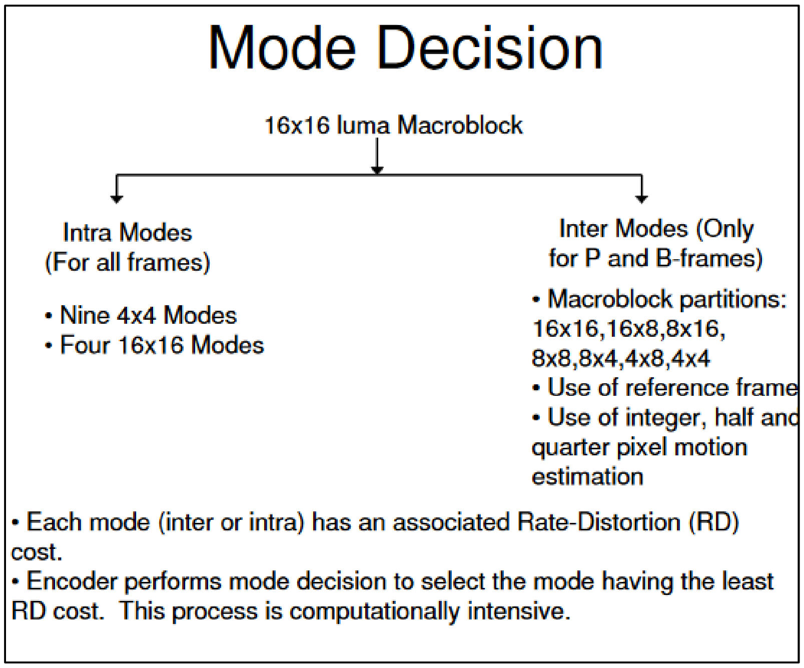
Macroblock layer semantics

The following semantics are assigned to the macroblock types in Table 7-13:

- P_L0_16x16: the samples of the macroblock are predicted with one luma macroblock partition of size 16x16 luma samples and associated chroma samples.
- P_L0_L0_MxN, with MxN being replaced by 16x8 or 8x16: the samples of the macroblock are predicted using two luma partitions of size MxN equal to 16x8, or two luma partitions of size MxN equal to 8x16, and associated chroma samples, respectively.
- P_8x8: for each sub-macroblock an additional syntax element (sub_mb_type[mbPartIdx] with mbPartIdx being the macroblock partition index for the corresponding sub-macroblock) is present in the bitstream that specifies the type of the corresponding sub-macroblock (see subclause 7.4.5.2).
- P_8x8ref0: has the same semantics as P_8x8 but no syntax element for the reference index (ref_idx_l0[mbPartIdx] with mbPartIdx = 0..3) is present in the bitstream and ref_idx_l0[mbPartIdx] shall be inferred to be equal to 0 for all sub-macroblocks of the macroblock (with indices mbPartIdx = 0..3).
- P_Skip: no further data is present for the macroblock in the bitstream.

Source: H.264 Standard (03-2010), p. 100

Mode Decision



Mode Decision

16x16 luma Macroblock

Intra Modes
(For all frames)

• Nine 4x4 Modes
• Four 16x16 Modes

Inter Modes (Only
for P and B-frames)

• Macroblock partitions:
16x16,16x8,8x16,
8x8,8x4,4x8,4x4
• Use of reference frame
• Use of integer, half and
quarter pixel motion
estimation

• Each mode (inter or intra) has an associated Rate-Distortion (RD) cost.
• Encoder performs mode decision to select the mode having the least RD cost. This process is computationally intensive.

Source: https://courses.cs.washington.edu/courses/csep590a/07au/lectures/rahullarge.pdf, p. 30

92. The '005 Accused Infringing Devices provide a method for generating a motion vector for the first pixel array in response to the determining step. The encoder calculates the appropriate motion vectors and other data elements represented in the video data stream.

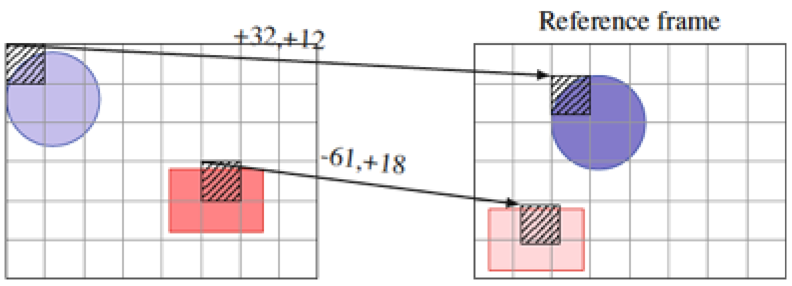
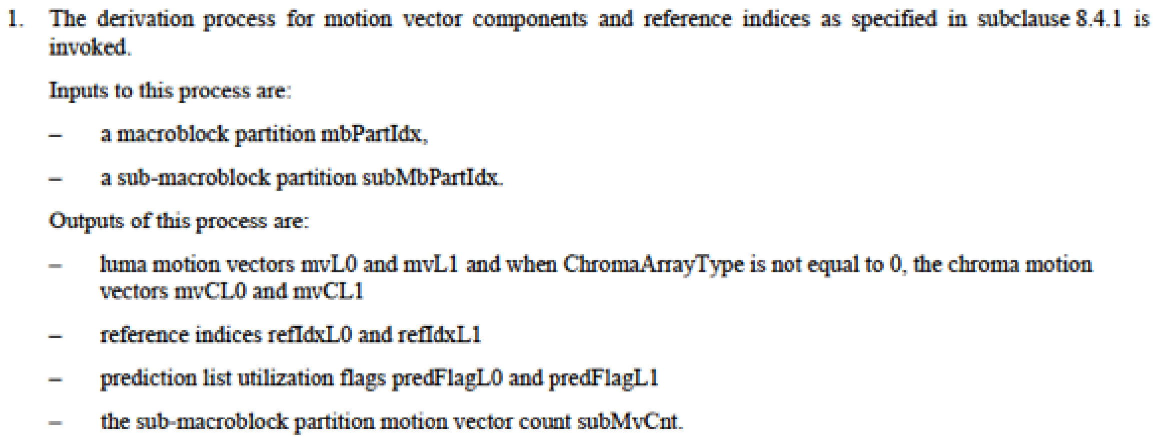


Fig. 2.4: Motion estimation. For each MB the best matching block in the reference frame is found. The encoder codes the differences (errors) between the MBs and their best matching blocks. Arrows indicate motion vectors and are labeled by the vector coordinates. In this example the shapes are identical but their colors are slightly larger/darker.

Source: B. Juurlink et al., Scalable Parallel Programming Applied to H.264, Chapter 2: Understanding the Application: An Overview of the H.264 Standard, p. 12

Motion Vector Derivation is described below



1. The derivation process for motion vector components and reference indices as specified in subclause 8.4.1 is invoked.

   Inputs to this process are:

   – a macroblock partition mbPartIdx,

   – a sub-macroblock partition subMbPartIdx.

   Outputs of this process are:

   – luma motion vectors mvL0 and mvL1 and when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1

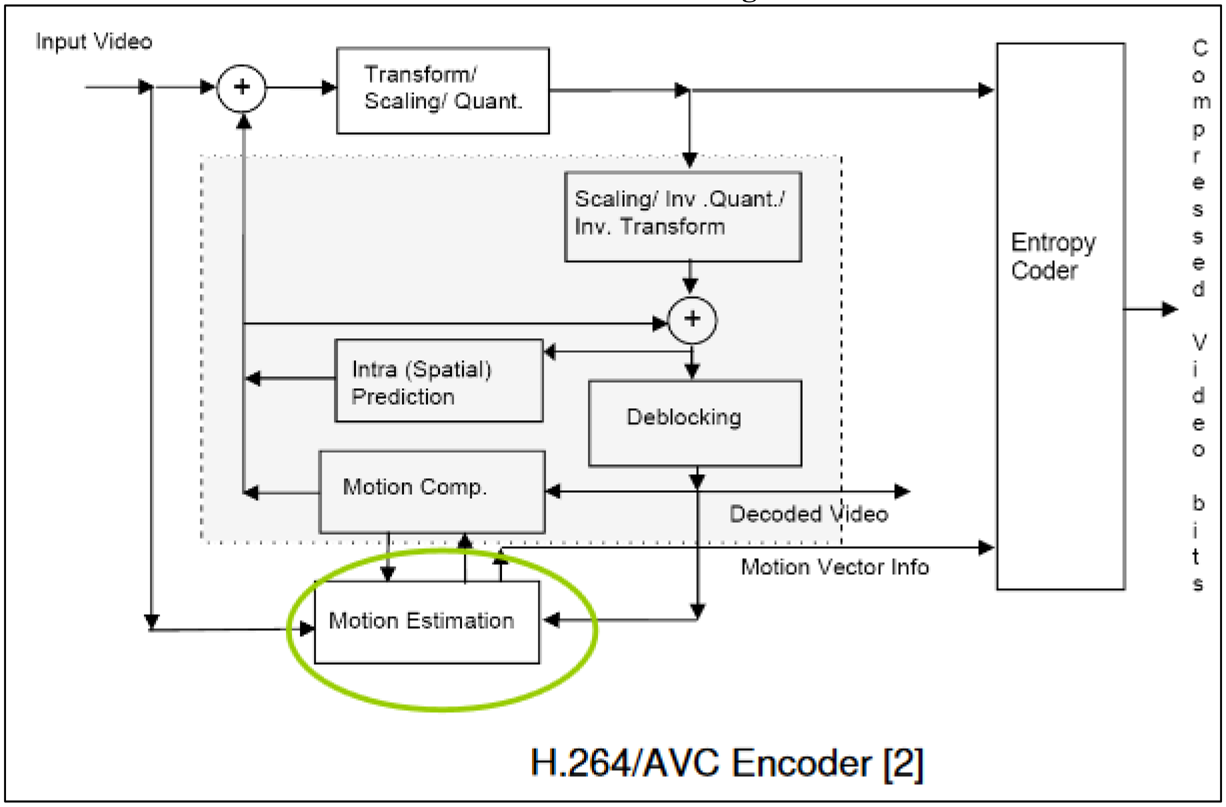
   – reference indices refIdxL0 and refIdxL1

   – prediction list utilization flags predFlagL0 and predFlagL1

   – the sub-macroblock partition motion vector count subMvCnt.

Source: H.264 Standard (03-2010), p. 151

H.264 Encoder Block Diagram



Source: https://courses.cs.washington.edu/courses/csep590a/07au/lectures/rahullarge.pdf, p. 2

93. Bitmovin has thus infringed at least claim 1 of the ’005 patent by making, using, testing, selling, offering for sale, importing and/or licensing the ’005 Accused Infringing Devices, and operating them such that all steps of at least claim 1 are performed.

94. Bitmovin’s acts of direct infringement have caused damage to Uniloc, and Uniloc is entitled to recover damages sustained as a result of Bitmovin’s wrongful acts in an amount subject to proof at trial.

COUNT IV: INFRINGEMENT OF THE ’345 PATENT

95. The allegations of paragraphs 1-7 of this First Amended Complaint are incorporated by reference as though fully set forth herein.

96. Uniloc owns by assignment the entire right, title, and interest in the ’345 patent.

97. The ’345 patent, is titled “Replacement of Substrings in File/Directory Pathnames With Numeric Tokens.” issued on October 22, 2002. A copy of the ’345 patent is attached as Exhibit D. The priority date for the ’345 patent is January 4, 2000. The inventions of the ’345 patent were developed by IBM.

98. Pursuant to 35 U.S.C. § 282, the ’345 patent is presumed valid.

99. Claim 1 of the ’345 patent addresses a technological problem indigenous to data processing systems and file systems in a networked environment—specifically in the computer science field of canonicalization. https://en.wikipedia.org/wiki/Canonicalization.

100. Claim 1 of the ’345 patent reads as follows:

> 1. A method for replacing substrings in file and directory pathnames with tokens in a computer-implemented file system, comprising the acts of:
>
> reading a name string to be converted into a list of tokens;
>
> canonicalizing a current working directory and the name string to form a pathname containing a plurality of substrings;

> parsing the pathname and replacing each substring with an associated token; and
>
> validating the parsed pathname containing the list of tokens.

101. The invention of claim 1 of the '345 patent concerns a novel method for canonicalization where substrings are replaced in file and directory pathnames with tokens in the computer-implemented file system.

102. At the time of invention of the '345 patent, in the field of data processing systems and file systems in a networked environment, canonicalization was a task used in file systems to identify file system resources, such as files, directories or other types of resources. '345 patent at 1:11-19. Another important task at the time is the semantic validation of a path, made up of the root, intermediate directories, and file or directory specification. *Id.* at 1:20-22. All intermediate directories must be valid for a pathname to refer to a valid file system resource. *Id.* at 1:22-27. Canonicalization and validation are often intertwined in a single function or set of functions. *Id.* at 1:28-29. The combination of these two functions can effect some savings by being more efficient. *Id.* at 1:34-35. If the current working directory for a given process is taken to be always valid, then validation of a path can start with the partial information specified by the user of the file system. *Id.* at 1:35-40. However useful this method of combining these two functions can be, the two tasks must always be considered separately, or severe penalties could occur. *Id.* at 1:41-44.

103. As demonstrated below, the claimed invention of claim 1 of the '345 patent provides a technological solution to the problem faced by the inventors—replacing substrings in file and directory pathnames with tokens in a computer-implemented file system by parsing pathnames and replacing each substring with an associated token and validating the parsed pathname containing a list of tokens. This technological solution resulted in a significant and

substantial improvement in the performance of storage of strings as well as in the performance of comparing substrings and savings in the amount of storage needed to implement a file system as only one copy need be kept of any substring. ’345 patent at 2:24-41.

104. As detailed in the specification, in designing a file system that is structured on a client/server split, where the client portion keeps track of a current working directory and therefore has to perform the canonicalization, the path validation can often only be efficiently done by the server. *Id.* at 1:52-56. The inventors discovered that in most cases even where there is no client/server split, it is advantageous to separate canonicalization from validation and perform these two operations in a close sequence, but not interleaving validation of intermediate path information with a forming of a canonical name. *Id.* at 1:56-62. This results in a simpler implementation and superior performance, especially in a network environment. *Id.* at 1:62-63.

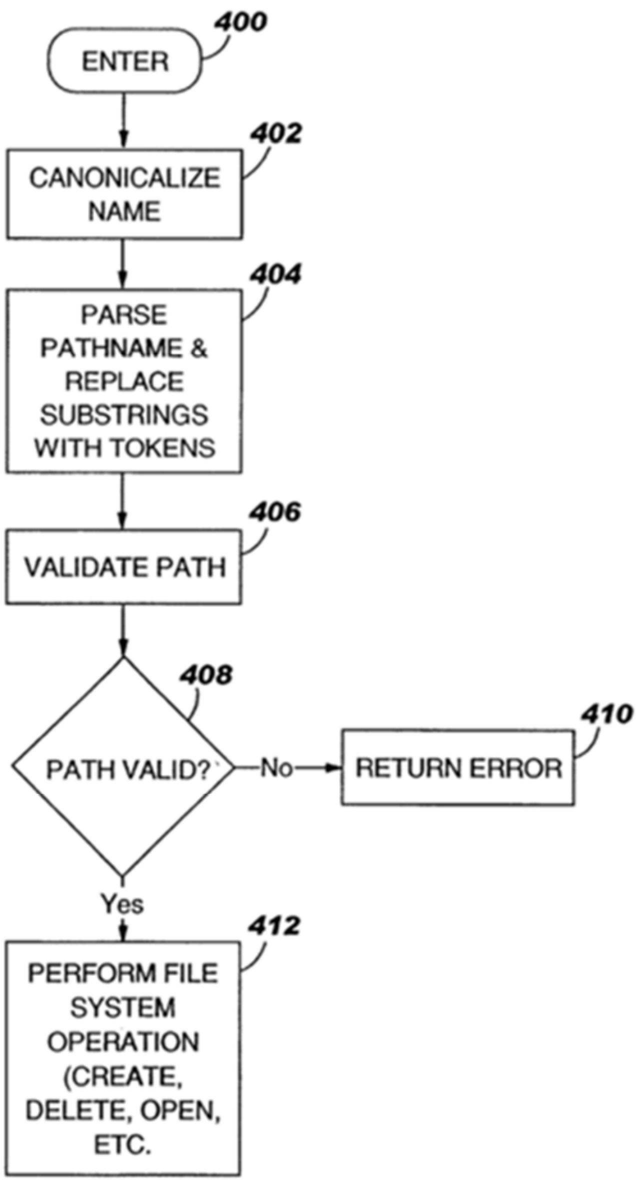
105. In dealing with file/directory pathnames, the number of sometimes quite lengthy strings poses a significant problem, especially when these are broken into substrings which then are constantly compared to other substrings. *Id.* at 2:24-27. According to the invention of the ’345 patent, parsing the strings into their semantically correct substrings and replacing those substrings with unique numeric tokens provides a significant improvement in the storage of the strings as well as better performance in comparing those substrings. *Id.* at 2:27-31. Since each substring (e.g., a subdirectory, filename or extension) is replaced with a numeric value, these numeric values can be arithmetically compared (e.g., is a ==b) instead of string compared (i.e., are all characters the same, what about uppercase vs. lowercase, etc.). *Id.* at 2:32-36. This represents a substantial improvement in performance. *Id.* at 2:36-38. In addition, by keeping a string dictionary, which the token uniquely indexes, only one copy is kept of any substring. *Id.*

at 2:38-39. This too can represent a substantial savings in the amount of storage needed to implement a file system. *Id.* at 2:40-41.

106. The foregoing is set forth in Figures 4-7 and the accompanying text:

> FIG. 4 illustrates a high-level flowchart of the token replacement process of the present invention. The process starts in entry block 400 in which the current working directory and filename (e.g., current-work-dir=.backslash.dir1.backslash.dir2; name=filename) are input to the canonicalization process as indicated by logic block 402. This action results in the canonical form such as pathname=.backslash.dir1.backslash.dir2.backslash.filename. This is followed in logic block 404 with parsing of the pathname and replacement of substrings with tokens. The substrings in this small example are "dir1", "dir2", and "filename". The result of this action are tokens t1, t2, and t3. The validation of the path is the next act in the process as indicated by logic block 406. From this act the process continues in decision block 408 with a determination of the validity of the path. If the path is found to be invalid an error is returned as indicated by termination block 410. Otherwise, the path is found to be valid and a file system operation is performed as indicated in logic block 412.

FIG. 4

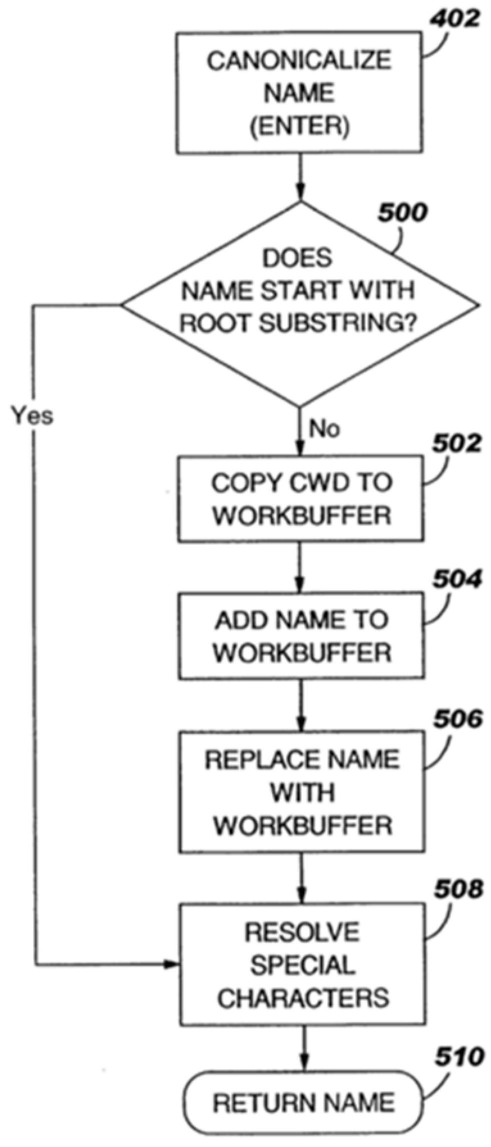

400
ENTER
402
CANONICALIZE NAME
404
PARSE PATHNAME & REPLACE SUBSTRINGS WITH TOKENS
406
VALIDATE PATH
408
PATH VALID?
No
410
RETURN ERROR
Yes
412
PERFORM FILE SYSTEM OPERATION (CREATE, DELETE, OPEN, ETC.

’345 patent at 10:58-65, Fig. 4.

> FIG. 5 illustrates the specific acts of the canonicalization process 402 of FIG. 4. It begins in decision block 500 with a determination if the name starts with a root substring. If it does, then processing jumps to logic block 508 for resolution of special characters in the name. If the name does not start with a root substring, then in logic block 502 the current working directory is copied to a work buffer. The content of the work buffer at this point in the process is .backslash.dir1.backslash.dir2. Next, the name (i.e., filename) is added to the work buffer as indicated in logic block 504. The content of the work buffer at this point

is .backslash.dir1.backslash.dir2.backslash.filename. In logic block 506, the name is replaced with the work buffer contents. The process concludes in logic block 508 with the resolution of special characters such as ".." or ".". The canonicalization process exits back to the many processing logic in termination block 510.

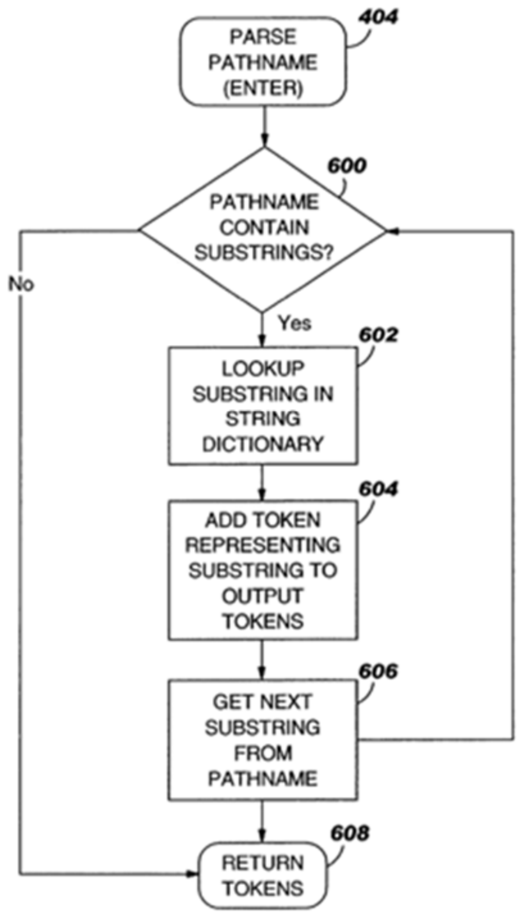
FIG. 5


402
CANONICALIZE NAME (ENTER)
500
DOES NAME START WITH ROOT SUBSTRING?
Yes
No
502
COPY CWD TO WORKBUFFER
504
ADD NAME TO WORKBUFFER
506
REPLACE NAME WITH WORKBUFFER
508
RESOLVE SPECIAL CHARACTERS
510
RETURN NAME

’345 patent at 10:66-11:14, Fig. 5.

FIG. 6 illustrates a flowchart of the parsing process 404 of the present invention. It commences with the entry of decision block 600 which initiates an iterative routine to perform as long as the pathname contains substrings. The iterative routine begins in logic block 602 in which a substring is looked up in the string dictionary. If the substring does not exist then a new token is created to represent that substring. In logic block 604, the token representing the substring is added to a list of output tokens for the pathname. The next act is to get the next substring from the pathname as indicated in logic block 606. The iterative routine loops back to decision block 600. After the entire pathname has been parsed into substrings and replaced with tokens (DONE indication out of decision block 600), the parsing process retuns the tokens found as indicated in termination block 608.

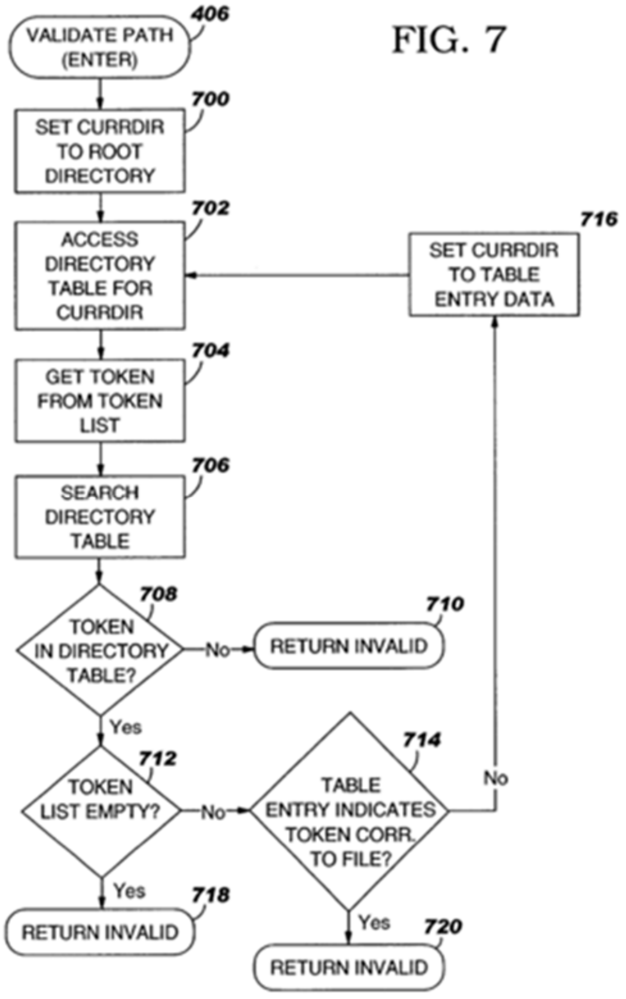
FIG. 6


404
PARSE PATHNAME (ENTER)
600
PATHNAME CONTAIN SUBSTRINGS?
No
Yes
602
LOOKUP SUBSTRING IN STRING DICTIONARY
604
ADD TOKEN REPRESENTING SUBSTRING TO OUTPUT TOKENS
606
GET NEXT SUBSTRING FROM PATHNAME
608
RETURN TOKENS

’345 patent at 11:15-11:30, Fig. 6.

FIG. 7 illustrates a flowchart of the validation process 406 of the present invention. The token list is input to logic block 700 in which the current directory is set to the root directory. In logic block 702, the directory table is accessed for the current directory. This is followed in logic block 704 with the act of getting a token from the token list. Next, in logic block 706, a search is performed to locate the token in the directory table. In decision block 708, a test is made to determine if the token was found in the directory table. If the search failed, then an invalid pathname indication is returned to the main processing logic via termination block 710. If the search was successful, processing continues in decision block 712, in which a test is made to determine if the token list is empty. If not, the processing continues in decision block 714 in which a determination is made as to whether or not the directory table entry found is for a file (rather than for a directory). If the directory table entry is for a directory, then processing continues in logic block 716 in which the current directory is set to the table entry data; processing then returns to logic block 702. If the directory table entry found in decision block 714 is for a file, then processing ends in termination block 720 with an invalid pathname indication. If, in decision block 712, the token list was found to be empty (i.e., all tokens have been processed) then processing exits in termination block 718 with the return of an valid pathname.

FIG. 7


406
VALIDATE PATH (ENTER)
700
SET CURRDIR TO ROOT DIRECTORY
702
ACCESS DIRECTORY TABLE FOR CURRDIR
716
SET CURRDIR TO TABLE ENTRY DATA
704
GET TOKEN FROM TOKEN LIST
706
SEARCH DIRECTORY TABLE
708
TOKEN IN DIRECTORY TABLE?
No
710
RETURN INVALID
Yes
712
TOKEN LIST EMPTY?
No
714
TABLE ENTRY INDICATES TOKEN CORR. TO FILE?
No
Yes
718
RETURN INVALID
Yes
720
RETURN INVALID

’345 patent at 11:31-11:56, Fig. 7.

107. Figures 8A and 8B contrast the prior art with the inventions of the ’345 patent and the accompanying text explains the advantages of the inventions of the ’345 patent over the prior art:

> FIGS. 8A-8B indicate both the prior art and the inventive method of storing directory and file names on a storage device, such as a disk. FIG. 8A shows a linked list structure with dir1 stored in root block memory location 80, dir2 stored in subdirectory block memory location 82, the filename stored in subdirectory block memory location 84, and the actual file stored at memory location 86. FIG. 8B indicates the method of storing directory and pathnames according to the present invention. Token t1 is stored in root block memory location 90, token t2 is stored in subdirectory block memory location 92, token t3 is stored in subdirectory block memory location 94 which contains a pointer to the file stored at memory location 96. Also shown in FIG. 8B is the string dictionary 98 corresponding to this simple example.

FIG. 8A

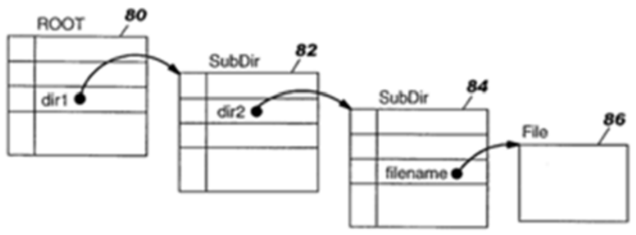
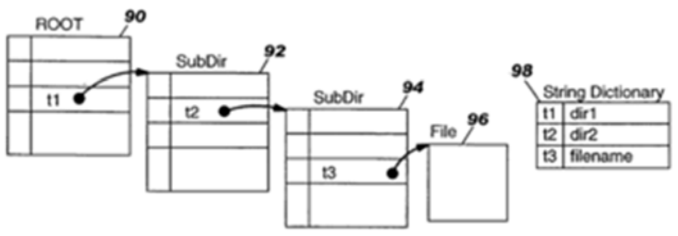
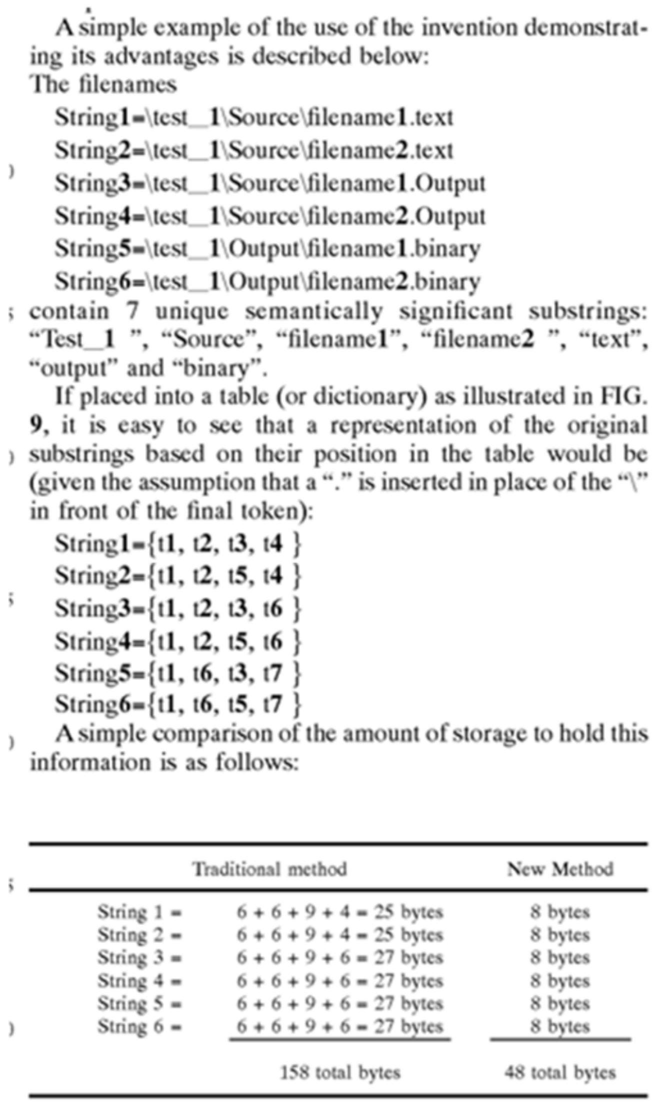


FIG. 8B





A simple example of the use of the invention demonstrating its advantages is described below:
The filenames

String**1**=\test__**1**\Source\filename**1**.text
String**2**=\test__**1**\Source\filename**2**.text
String**3**=\test__**1**\Source\filename**1**.Output
String**4**=\test__**1**\Source\filename**2**.Output
String**5**=\test__**1**\Output\filename**1**.binary
String**6**=\test__**1**\Output\filename**2**.binary

contain 7 unique semantically significant substrings: "Test__**1** ", "Source", "filename1", "filename**2** ", "text", "output" and "binary".

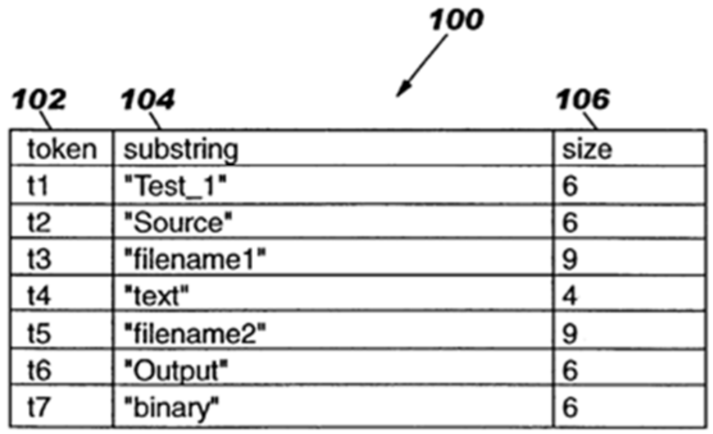
If placed into a table (or dictionary) as illustrated in FIG. **9**, it is easy to see that a representation of the original substrings based on their position in the table would be (given the assumption that a "." is inserted in place of the "\" in front of the final token):

String**1**={t**1**, t**2**, t**3**, t**4** }
String**2**={t**1**, t**2**, t**5**, t**4** }
String**3**={t**1**, t**2**, t**3**, t**6** }
String**4**={t**1**, t**2**, t**5**, t**6** }
String**5**={t**1**, t**6**, t**3**, t**7** }
String**6**={t**1**, t**6**, t**5**, t**7** }

A simple comparison of the amount of storage to hold this information is as follows:

	Traditional method	New Method
String 1 =	6 + 6 + 9 + 4 = 25 bytes	8 bytes
String 2 =	6 + 6 + 9 + 4 = 25 bytes	8 bytes
String 3 =	6 + 6 + 9 + 6 = 27 bytes	8 bytes
String 4 =	6 + 6 + 9 + 6 = 27 bytes	8 bytes
String 5 =	6 + 6 + 9 + 6 = 27 bytes	8 bytes
String 6 =	6 + 6 + 9 + 6 = 27 bytes	8 bytes
	158 total bytes	48 total bytes

FIG. 9



token	substring	size
t1	"Test_1"	6
t2	"Source"	6
t3	"filename1"	9
t4	"text"	4
t5	"filename2"	9
t6	"Output"	6
t7	"binary"	6

However, this greater than 3 to 1 comparison ratio is not quite entirely complete in that there is an "overhead" of 81 bytes to store the substrings in a dictionary (as null-terminated strings) along with the pointers to locate them. This overhead, while not negligible, is not as significant as the savings in replacing substrings with 2-byte numeric tokens.

The difference in speed of comparison is not quite so readily calculated. It is clear that comparing a new string:
StringN=.backslash.Test_1.backslash.Output.backslash.filename2.binary.NEW

with String6, character by character, would involve 32 comparisons of single bytes until a mismatch is found. A simple comparison of the two strings using the token-scheme would require four comparisons of 2-byte tokens.

Again, this 8 to 1 ratio is not entirely complete in that the conversion of the strings into substrings and proper insertion into the table require some overhead, but in a file system where locating information is much more frequent than inserting, removing or renaming it, this overhead is not as significant as the savings in numeric comparisons verus string comparisons.

A third advantage that is usually involved whenever data compression is present is the additional security for a file system that uses the new method. Several schemes could be easily applied to prevent the string dictionary from being accessed even though the file and directory names may be available. This is the "shared-secret" type of security and is the most difficult to decrypt. While the substrings themselves can also be encrypted, it would be easier to take advantage of the clean split between the semantic information embodied in the tokens and the human-readable form of the strings to deter someone from locating secure information in a file system.

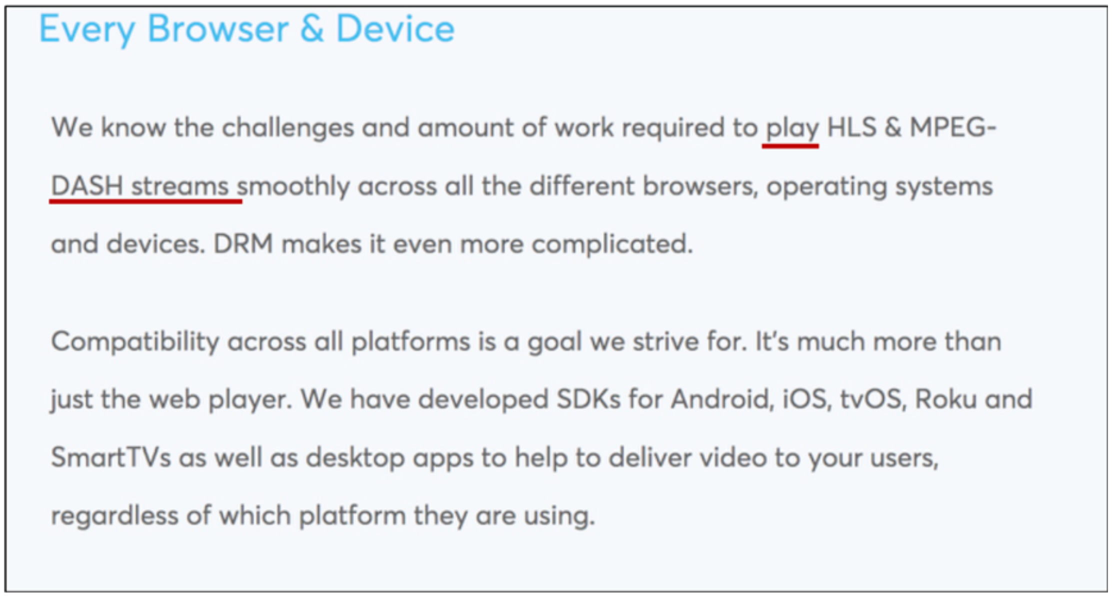
The fourth advantage is that of the additional flexibility that tokenizing the substrings provides. Since the actual substrings are stored in a separate place from the directory and file information in the native file system, limits on the length of a substring, overall length of a path (composed of many substrings) as well as the permissible characters in any substring can be much different than those imposed by the native file system. As long as the sequence of tokens can be uniquely mapped to a native file system resource practically any string can be accommodated. The tokens are used only to uniquely represent the substrings, wherever they may be used in a file system name. A clear example is the above use of "Output" as both a sub-directory name and as a file "extension" in String3 and String5 for instance.

’345 patent at 11:57-13:23, Figs. 8A, 8B and 9.

108. As set forth above, claim 1 of the '345 patent presented an unconventional method for canonicalization for computers that led to better performance of computers and enhanced storage. In light of the foregoing, a person of ordinary skill in the art would understand that claim 1 of the '345 patent is directed to a method for replacing substrings in file and directory pathnames with tokens in a computer-implemented file system. Moreover, a person of ordinary skill in the art would understand that claim 1 of the '345 patent contains that corresponding inventive concept of replacing substrings in file and directory pathnames with tokens in a computer-implemented file system by parsing pathnames and replacing each substring with an associated token and validating the parsed pathname containing a list of tokens.

109. Upon information and belief, Bitmovin has directly infringed at least claim 1 of the '345 patent by making, using, testing, selling, offering for sale, importing and/or licensing in the United States without authority products and services that perform a method for replacing substrings in file and directory pathnames with tokens in a computer-implemented file system, including an MPEG-DASH compatible video player (collectively "the '345 Accused Infringing Devices") in an exemplary manner as described below.

110. The '345 Accused Infringing Devices perform a method for replacing substrings in file and directory pathnames with tokens in a computer-implemented file system. The '345 Accused Infringing Devices include a MPEG-DASH compatible video player. DASH video streams include a media presentation description (MPD) which is a manifest of the media segments that make up the complete media presentation. The MPD contains file and directory pathnames to access these segments in the form of HTTP URLs.

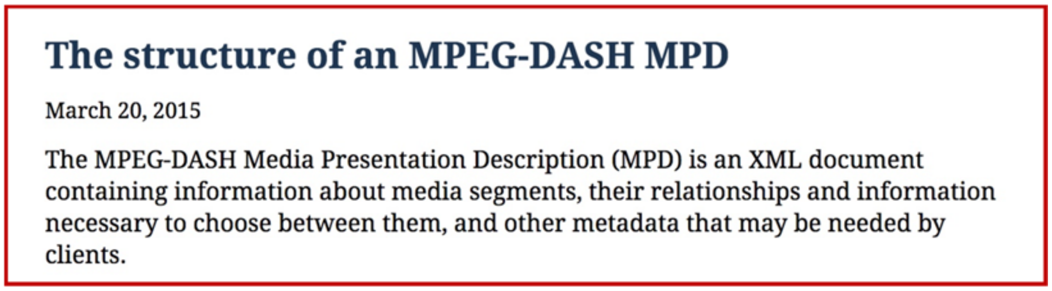

Every Browser & Device

We know the challenges and amount of work required to play HLS & MPEG-DASH streams smoothly across all the different browsers, operating systems and devices. DRM makes it even more complicated.

Compatibility across all platforms is a goal we strive for. It's much more than just the web player. We have developed SDKs for Android, iOS, tvOS, Roku and SmartTVs as well as desktop apps to help to deliver video to your users, regardless of which platform they are using.

Source: https://bitmovin.com/video-player/

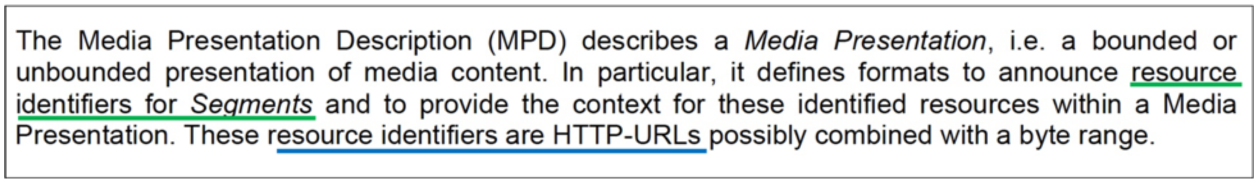
111. MPEG DASH in the '345 Accused Infringing Devices has a mechanism whereby URLs to access segment files can use a SegmentTemplate to specify file and pathnames. This mechanism allows DASH video players to replace specific substrings (identifiers) in the template with dynamic numbers (tokens) in a computer implemented file system (URLs).


The structure of an MPEG-DASH MPD

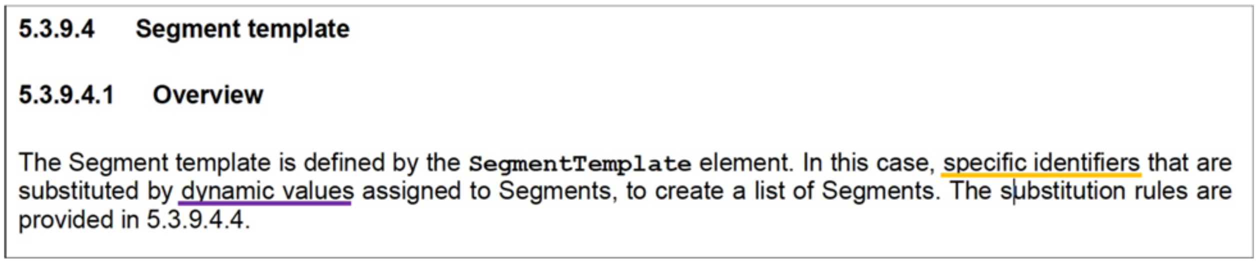
March 20, 2015

The MPEG-DASH Media Presentation Description (MPD) is an XML document containing information about media segments, their relationships and information necessary to choose between them, and other metadata that may be needed by clients.

Source: https://www.brendanlong.com/the-structure-of-an-mpeg-dash-mpd.html


The Media Presentation Description (MPD) describes a *Media Presentation*, i.e. a bounded or unbounded presentation of media content. In particular, it defines formats to announce resource identifiers for *Segments* and to provide the context for these identified resources within a Media Presentation. These resource identifiers are HTTP-URLs possibly combined with a byte range.

Source: ISO IEC 23009-1:2014, "Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats", p7

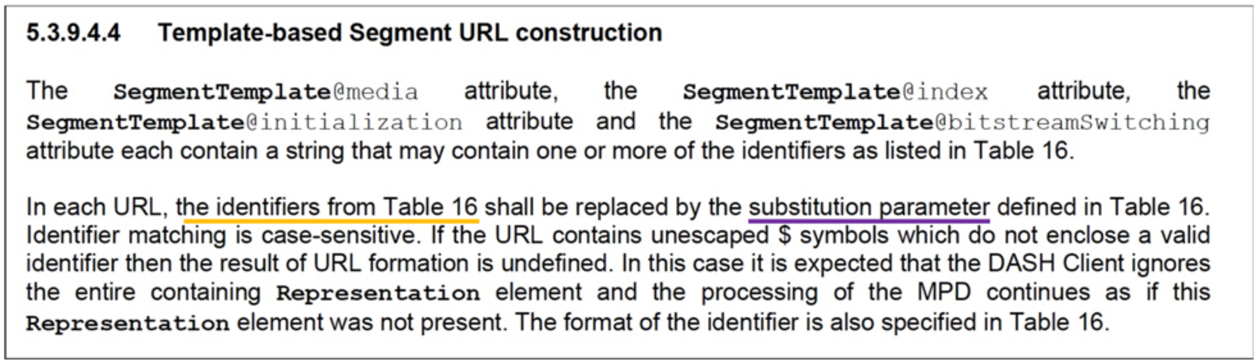


5.3.9.4 Segment template

5.3.9.4.1 Overview

The Segment template is defined by the **`SegmentTemplate`** element. In this case, specific identifiers that are substituted by dynamic values assigned to Segments, to create a list of Segments. The substitution rules are provided in 5.3.9.4.4.

Source: ISO IEC 23009-1:2014, "Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats", p53



5.3.9.4.4 Template-based Segment URL construction

The **`SegmentTemplate`**`@media` attribute, the **`SegmentTemplate`**`@index` attribute, the **`SegmentTemplate`**`@initialization` attribute and the **`SegmentTemplate`**`@bitstreamSwitching` attribute each contain a string that may contain one or more of the identifiers as listed in Table 16.

In each URL, the identifiers from Table 16 shall be replaced by the substitution parameter defined in Table 16. Identifier matching is case-sensitive. If the URL contains unescaped $ symbols which do not enclose a valid identifier then the result of URL formation is undefined. In this case it is expected that the DASH Client ignores the entire containing **`Representation`** element and the processing of the MPD continues as if this **`Representation`** element was not present. The format of the identifier is also specified in Table 16.

Source: ISO IEC 23009-1:2014, "Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats", p53

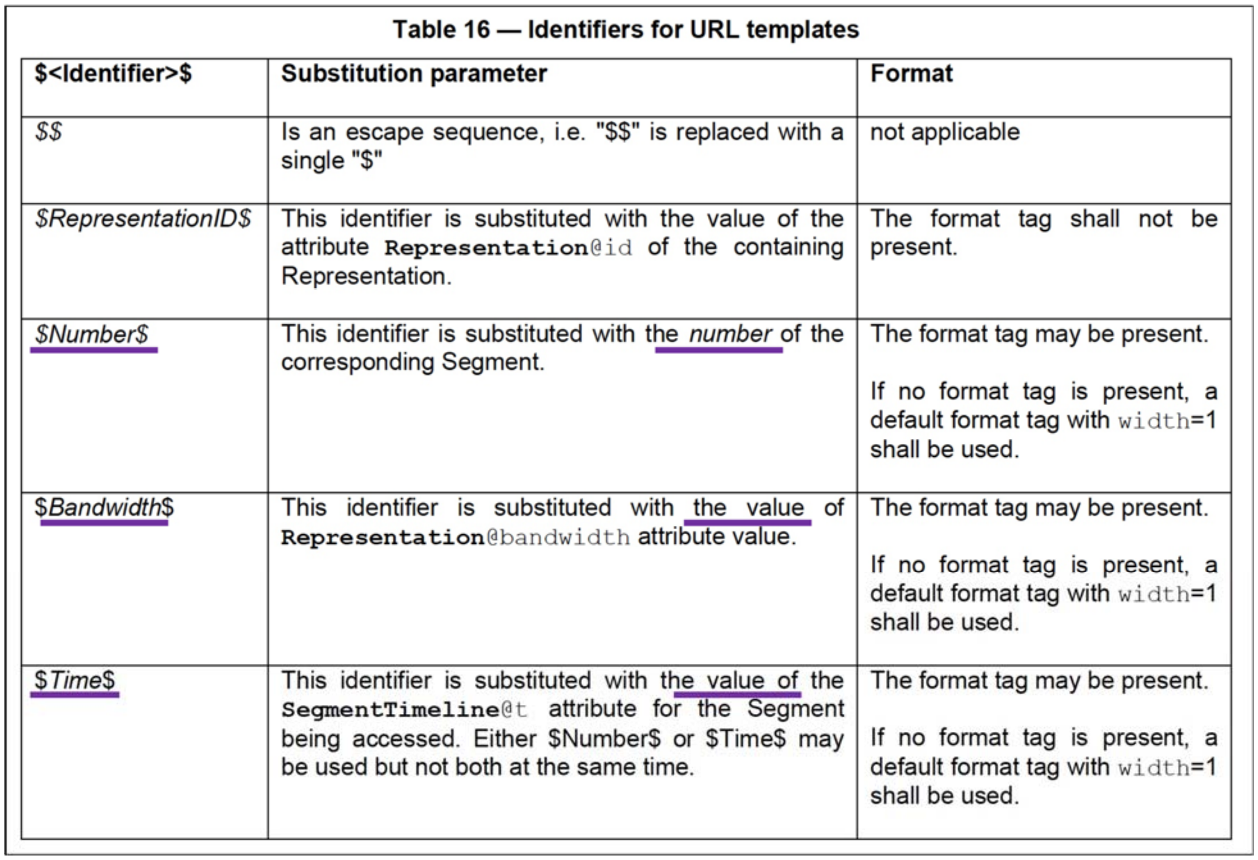


Table 16 — Identifiers for URL templates

$<Identifier>$	Substitution parameter	Format
$$	Is an escape sequence, i.e. "$$" is replaced with a single "$"	not applicable
$RepresentationID$	This identifier is substituted with the value of the attribute **Representation**@id of the containing Representation.	The format tag shall not be present.
$Number$	This identifier is substituted with the *number* of the corresponding Segment.	The format tag may be present. If no format tag is present, a default format tag with width=1 shall be used.
$*Bandwidth*$	This identifier is substituted with the value of **Representation**@bandwidth attribute value.	The format tag may be present. If no format tag is present, a default format tag with width=1 shall be used.
$*Time*$	This identifier is substituted with the value of the **SegmentTimeline**@t attribute for the Segment being accessed. Either $Number$ or $Time$ may be used but not both at the same time.	The format tag may be present. If no format tag is present, a default format tag with width=1 shall be used.

Source: ISO IEC 23009-1:2014, “Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats”, p55

112. The ’345 Accused Infringing Devices name a string to be converted into a list of tokens. For example, the ’345 Accused Infringing Devices read DASH MPD files to play media. MPD files can include SegmentTemplates with name strings according to the ISO IEC 23009-1 specification.

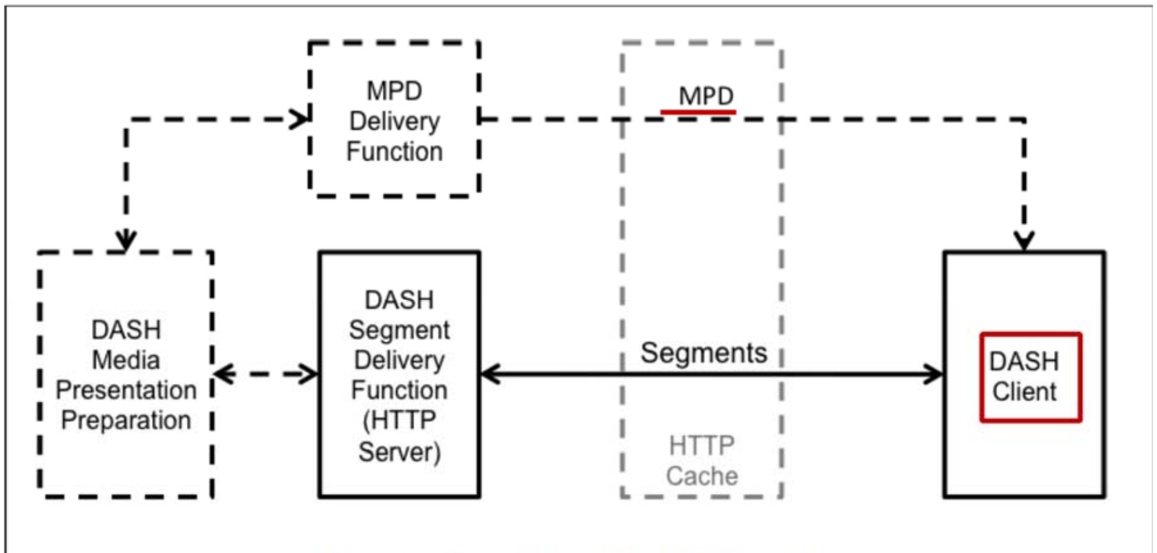
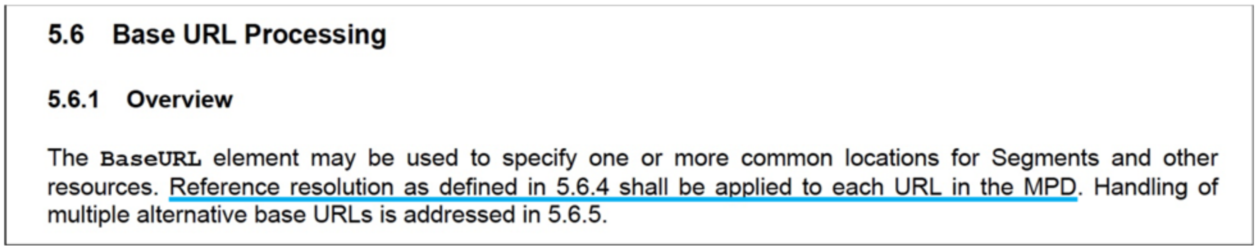

MPD Delivery Function
MPD
DASH Media Presentation Preparation
DASH Segment Delivery Function (HTTP Server)
Segments
HTTP Cache
DASH Client

Figure 1 — Example system for DASH formats

Source: ISO IEC 23009-1:2014, "Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats," p8

113. The '345 Accused Infringing Devices canonicalize a current working directory and the name string to form a pathname containing a plurality of substrings. For example, the '345 Accused Infringing Devices use a canonicalization process which converts the partial path/file name in the template into a complete path/file name using the MPEG DASH BaseURL mechanism. The MPEG-DASH specification requires that URL references in an MPD use reference resolution (canonicalization) for each URL in the MPD, including those related to media segments.



5.6 Base URL Processing

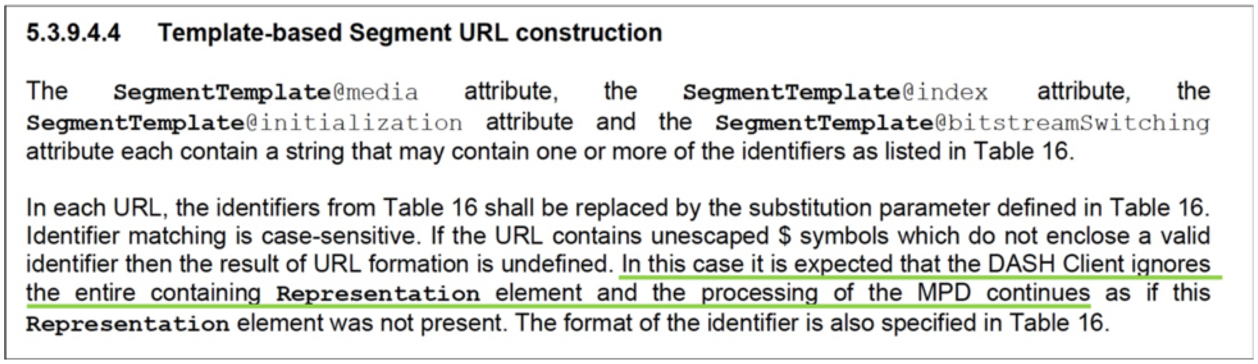
5.6.1 Overview

The `BaseURL` element may be used to specify one or more common locations for Segments and other resources. Reference resolution as defined in 5.6.4 shall be applied to each URL in the MPD. Handling of multiple alternative base URLs is addressed in 5.6.5.

Source: ISO IEC 23009-1:2014, "Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats", p64

114. The ’345 Accused Infringing Devices, according to the required behavior in the MPEG-DASH specification, parse the pathname and replace the substrings in Table 16 with the associated token.

115. The ’345 Accused Infringing Devices validate the parsed pathname and should ignore invalid pathnames within the context (Representation) in which they were defined.


5.3.9.4.4 Template-based Segment URL construction

The **SegmentTemplate**@media attribute, the **SegmentTemplate**@index attribute, the **SegmentTemplate**@initialization attribute and the **SegmentTemplate**@bitstreamSwitching attribute each contain a string that may contain one or more of the identifiers as listed in Table 16.

In each URL, the identifiers from Table 16 shall be replaced by the substitution parameter defined in Table 16. Identifier matching is case-sensitive. If the URL contains unescaped $ symbols which do not enclose a valid identifier then the result of URL formation is undefined. In this case it is expected that the DASH Client ignores the entire containing **Representation** element and the processing of the MPD continues as if this **Representation** element was not present. The format of the identifier is also specified in Table 16.

Source: ISO IEC 23009-1:2014, “Information technology — Dynamic adaptive streaming over HTTP (DASH) —Part 1: Media presentation description and segment formats,” p. 54

116. Bitmovin has thus infringed at least claim 1 of the ’345 patent by making, using, testing, selling, offering for sale, importing and/or licensing the ’345 Accused Infringing Devices, and operating them such that all steps of at least claim 1 are performed.

117. Bitmovin has induced infringement of least claim 1 of the ’345 patent since the filing of this action on January 30, 2019. *See, e.g.*, *DermaFocus LLC v. Ulthera, Inc.*, 201 F. Supp. 3d 465, 470-472 (D. Del. Aug. 16, 2016); *Softwview LLC v. Apple Inc.*, 2012 WL 3061027, at *7 (D. Del. July 26, 2012); *Apeldyn Corp. v. Sony Corp.*, 852 F. Supp. 2d 568, 573-74 (D. Del. 2012). Bitmovin’s customers are direct infringers of claim 1 of the ’345 patent when the customers use the ’345 Accused Infringing Devices (*i.e.*, Bitmovin’s MPEG-DASH compatible video players) as described above in connection with Bitmovin’s own direct infringement. Having knowledge of its own infringement, Bitmovin has, since the filing of the complaint knowingly induced infringement and possessed the specific intent to encourage

infringement of its customers by intentionally instructing its customers to infringe through videos, demonstrations, brochures and user guides, such as those located at https://bitmovin.com/video-player/; https://bitmovin.com/docs/player/tutorials; https://bitmovin.com/docs/player/quickstarts; https://bitmovin.com/docs/player/faqs; https://bitmovin.com/docs/player/api-reference; https://bitmovin.com/docs/player/sdks; https://bitmovin.com/docs/player/releases; and related domains and subdomains. Bitmovin is thereby liable for infringement of the ’345 patent under 35 U.S.C. § 271 (b). *See*, *e.g.*, *DermaFocus*, 201 F. Supp. 3d at 471 (“Service of the original complaint in 2015, of course, gave defendant actual knowledge of the ’559 patent. Defendant argues that, nevertheless, the FAC contains insufficient facts relating to whether defendant has the additional knowledge that third parties (its customers) are infringing the patent. (D.I. 13 at 5) Having determined, however, that plaintiff adequately pled direct infringement, and given the information contained in the FAC regarding defendant’s promotional and educational materials (D.I. 11, exs. B, C and E), as well as use of the accused Ulthera System by a local physician, it is plausible to infer that defendant knew that the intended use of the Ulthera System (for which defendant’s customers received instructions) was infringing. The court finds these allegations sufficiently to plead induced infringement, that is, the FAC contains facts from which it is plausible to infer that defendant knew that its conduct would induce infringement by its customers, and had the specific intent to make it so.”).

118. Bitmovin is also liable for contributory infringement of least claim 1 of the ’345 patent since the filing of this action on January 30, 2019 for the same reasons it is liable for induced infringement and the following reasons. The portion of the ’345 Accused Infringing Devices (*i.e.*, Bitmovin’s MPEG-DASH compatible video players) that replaces substrings in file

and directory pathnames with tokens in the manner described above (which is herein incorporated by reference) is a component of the '345 Accused Infringing Devices and is a material part of the invention of the '345 patent. Since the filing of the complaint, Bitmovin has knowledge that this component is especially adapted for infringement of the '345 patent based on Uniloc's infringement allegations and is not a staple article suitable for substantial non-infringing use of MPEG-DASH compatible video players and necessarily infringes when used in the manner described above. *DermaFocus*, 201 F. Supp. 3d at 471-72 ("With respect to contributory infringement, the FAC alleges that defendant: (1) had (at least post-suit) knowledge of the patent; (2) is selling its Ulthera System which is especially made for infringing use; (3) had knowledge of the infringing use; (4) the Ulthera System has no substantial non-infringing use: and (5) there is direct infringement. (D.I. 111 at ¶¶ 15, 16 Such allegations have passed muster under *Twombley*, *Iqbal*, and their progeny in the past."). Bitmovin is thereby liable for infringement of the '345 patent under 35 U.S.C. § 271(c).

119. Bitmovin's acts of direct and indirect infringement have caused damage to Uniloc, and Uniloc is entitled to recover damages sustained as a result of Bitmovin's wrongful acts in an amount subject to proof at trial.

PRAYER FOR RELIEF

WHEREFORE, Uniloc 2017 respectfully requests the following relief:

A. A judgment that Bitmovin has infringed the ’712 patent;

B. A judgment that Bitmovin has infringed the ’118 patent;

C. A judgment that Bitmovin has infringed the ’005 patent;

D. A judgment that Bitmovin has infringed the ’345 patent;

E. A judgment that Uniloc be awarded damages adequate to compensate it for Bitmovin’s past infringement and any continuing or future infringement of the ’712 patent, the ’118 patent, the ’005 patent and the ’345 patent, including pre-judgment and post-judgment interest costs and disbursements as justified under 35 U.S.C. § 284 and an accounting;

F. That this be determined to be an exceptional case under 35 U.S.C. § 285;

G. That Uniloc be granted its reasonable attorneys’ fees in this action;

H. That this Court award Uniloc its costs; and

I. That this Court award Uniloc such other and further relief as the Court deems proper.

DEMAND FOR JURY TRIAL

Uniloc hereby demands trial by jury on all claims and issues so triable.

DATED: July 22, 2019

Respectfully submitted,

FARNAN LLP

/s/ Michael J. Farnan

Brian E. Farnan (Bar No. 4089)
Michael J. Farnan (Bar No. 5165)
919 North Market Street, 12th Floor
Wilmington, DE 19801
phone 302-777-0300
fax 302-777-0301
bfarnan@farnanlaw.com
mfarnan@farnanlaw.com

M. Elizabeth Day (admitted *pro hac vice*)
David Alberti (admitted *pro hac vice*)
Sal Lim (admitted *pro hac vice*)
Marc Belloli (admitted *pro hac vice*)
Feinberg Day Kramer Alberti Lim Tonkovich & Belloli LLP
1600 El Camino Real, Suite 280
Menlo Park, CA 94025
Tel: 650.618.4360
Fax: 650.618.4368
eday@feinday.com
dalberti@feinday.com
slim@feinday.com
mbelloli@feinday.com

Attorneys for
Uniloc 2017 LLC

EXHIBIT A

US006628712B1

(12) **United States Patent**
Le Maguet

(10) **Patent No.: US 6,628,712 B1**
(45) **Date of Patent: Sep. 30, 2003**

(54) SEAMLESS SWITCHING OF MPEG VIDEO STREAMS

(75) Inventor: **Yann Le Maguet**, Paris (FR)

(73) Assignee: **Koninklijke Philips Electronics N.V.**, Eindhoven (NL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 456 days.

(21) Appl. No.: **09/708,165**

(22) Filed: **Nov. 8, 2000**

(30) **Foreign Application Priority Data**

Nov. 23, 1999 (EP) 99402911

(51) **Int. Cl.**^{7} **H04N 7/12**
(52) **U.S. Cl.** **375/240.12**; 375/240.26
(58) **Field of Search** 375/240.02, 240.1, 375/240.12, 240.16, 240.13, 240.14, 240.15, 240.25, 240.26; 348/423.1, 425.1, 425.3, 416.1; 386/111; 382/235–236, 238; 358/261.2, 430

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,208,759	B1	*	3/2001	Wells	382/232
6,314,138	B1	*	11/2001	Lemaguet	375/240
6,393,057	B1	*	5/2002	Thoreau et al.	375/240
6,483,543	B1	*	11/2002	Zhang et al.	348/390.1
6,529,555	B1	*	3/2003	Saunders et al.	375/240.26
6,542,546	B1	*	4/2003	Vetro et al.	375/240.12

FOREIGN PATENT DOCUMENTS

EP	001079630 A1	*	2/2001	H04N/7/24
WO	WP 97 08898	*	3/1997	H04N/7/26
WO	WO9905870		2/1999	H04N/7/58

OTHER PUBLICATIONS

Youn et al., "Adaptive motion vector refinement for high performance transcoding", IEEE, International Conference on Image Processing, vol. 3, pp. 596–600.*

* cited by examiner

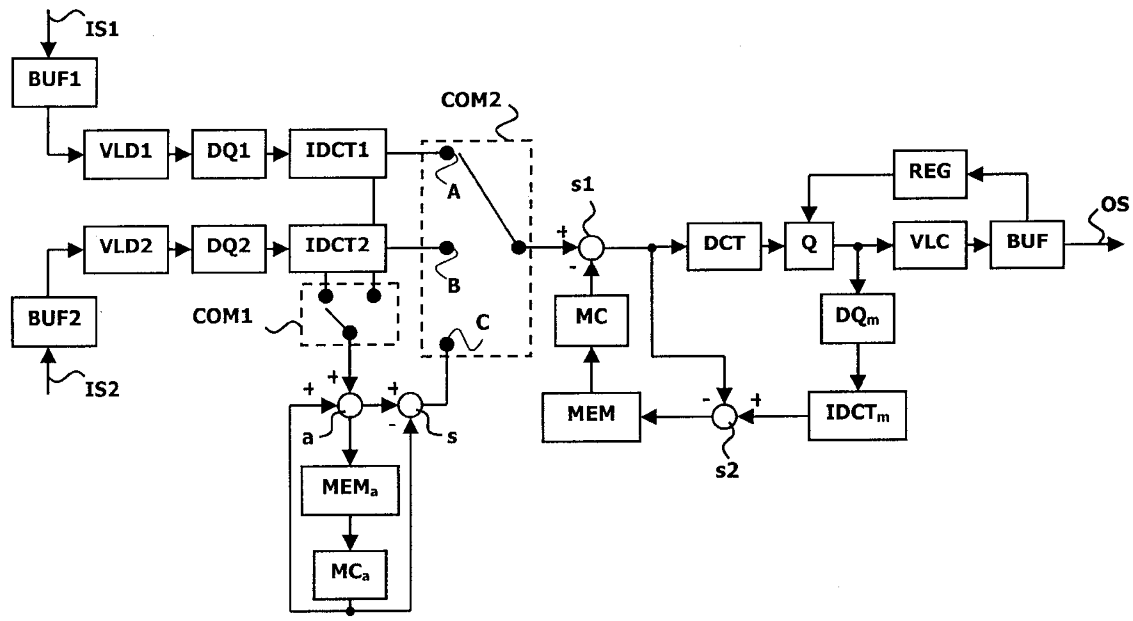
Primary Examiner—Vu Le
(74) *Attorney, Agent, or Firm*—Russell Gross

(57) **ABSTRACT**

A switching device SW allows to switch from a first compressed data input stream IS1 to a second compressed data input stream IS2, resulting in a compressed data output stream OS. This switching device comprises a buffer system BS intended to store the data contained in the first and second input streams, and control means CONT which controls the storage of the input streams in the buffer system in order to switch, at a switch request SWR, from the first input stream to the second input stream, using a commutation device COM.

A transcoding system TS is intended to receive the data stream at the output of the commutation device and to provide the output stream in a seamless way. The use of a transcoding system allows to avoid an underflow or an overflow of the buffer of the decoder that will have to decode the output stream. Moreover, said transcoding system allows to encode the output stream at a bit rate R, where R may be different from the bit rate R1 of the first input stream and the bit rate R2 of the second input stream.

8 Claims, 6 Drawing Sheets



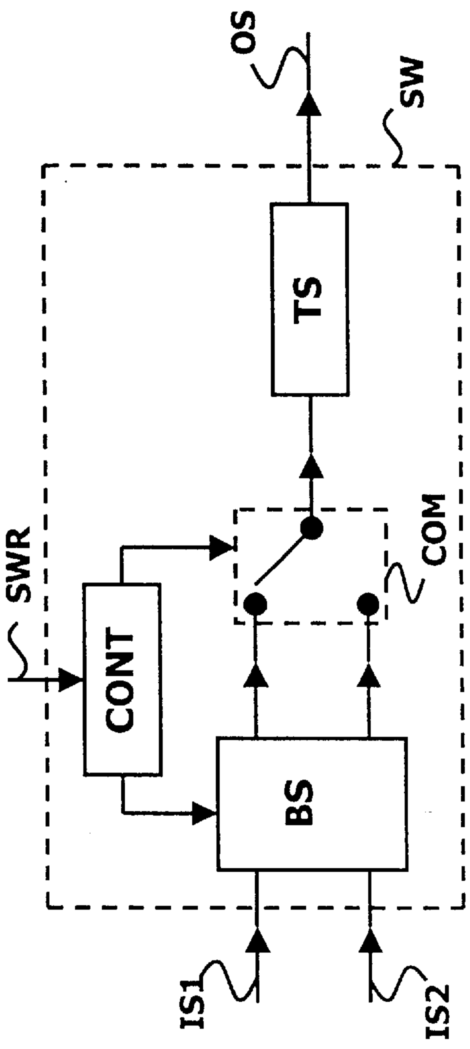

OS
SW
TS
COM
SWR
CONT
BS
IS1
IS2

FIG. 1

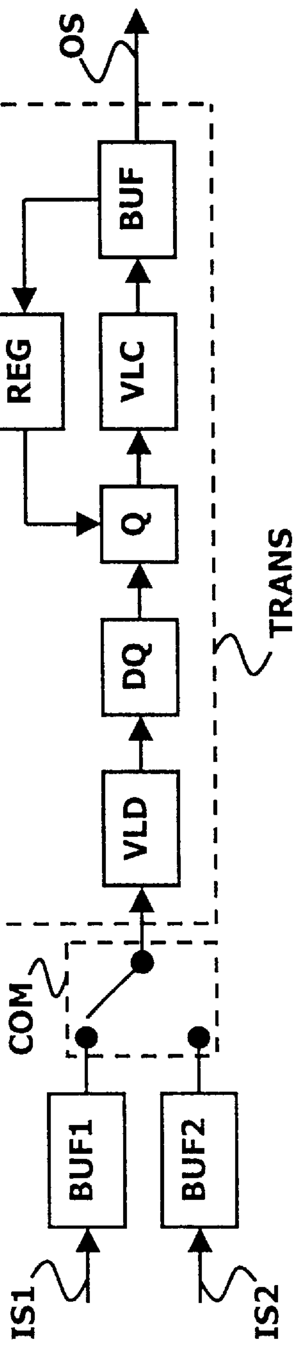

OS
BUF
REG
VLC
Q
DQ
VLD
TRANS
COM
BUF1
BUF2
IS1
IS2

FIG. 2

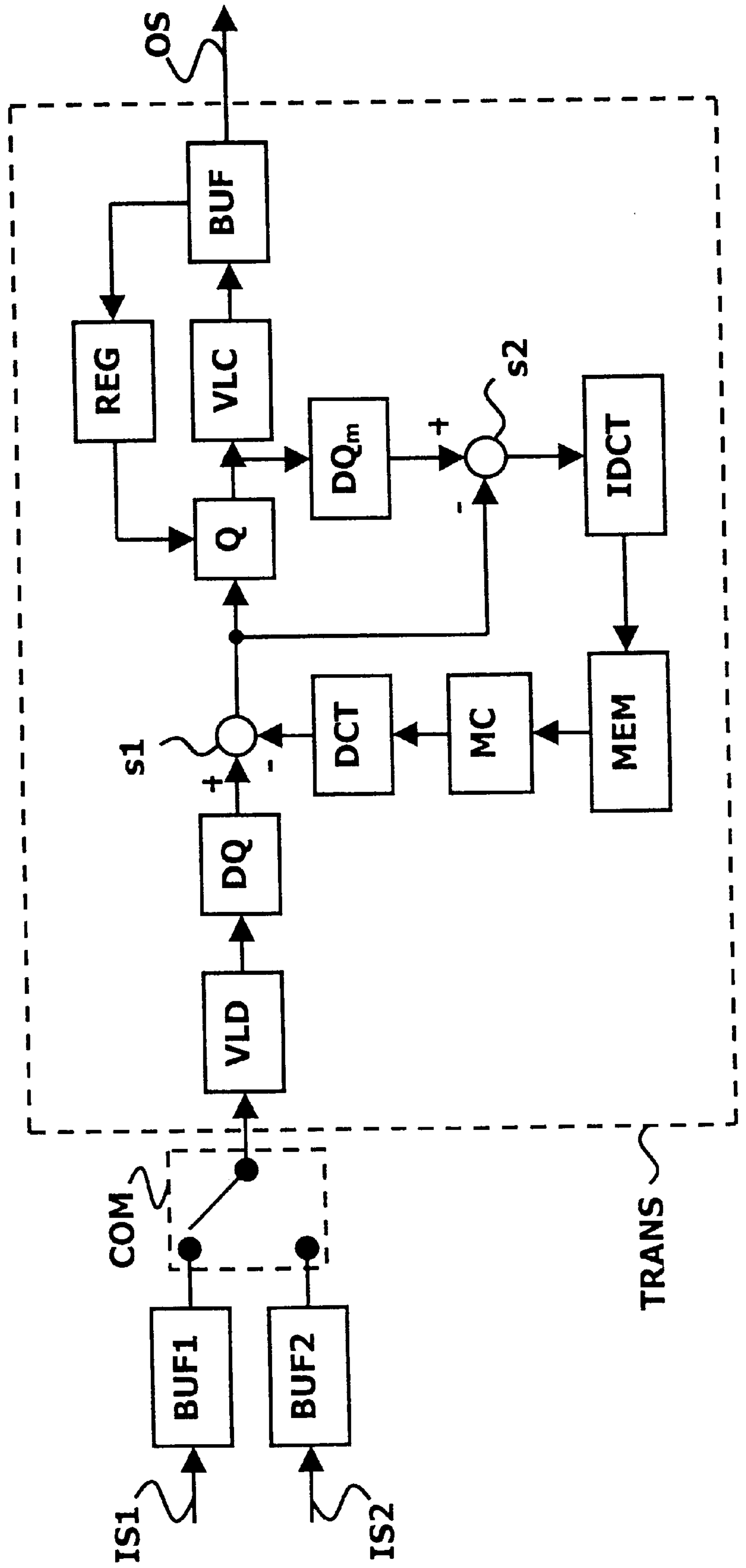

OS
BUF
REG
VLC
DQ_m
+
s2
-
IDCT
Q
s1
+
-
DCT
MC
MEM
DQ
VLD
COM
TRANS
BUF1
BUF2
IS1
IS2

FIG. 3

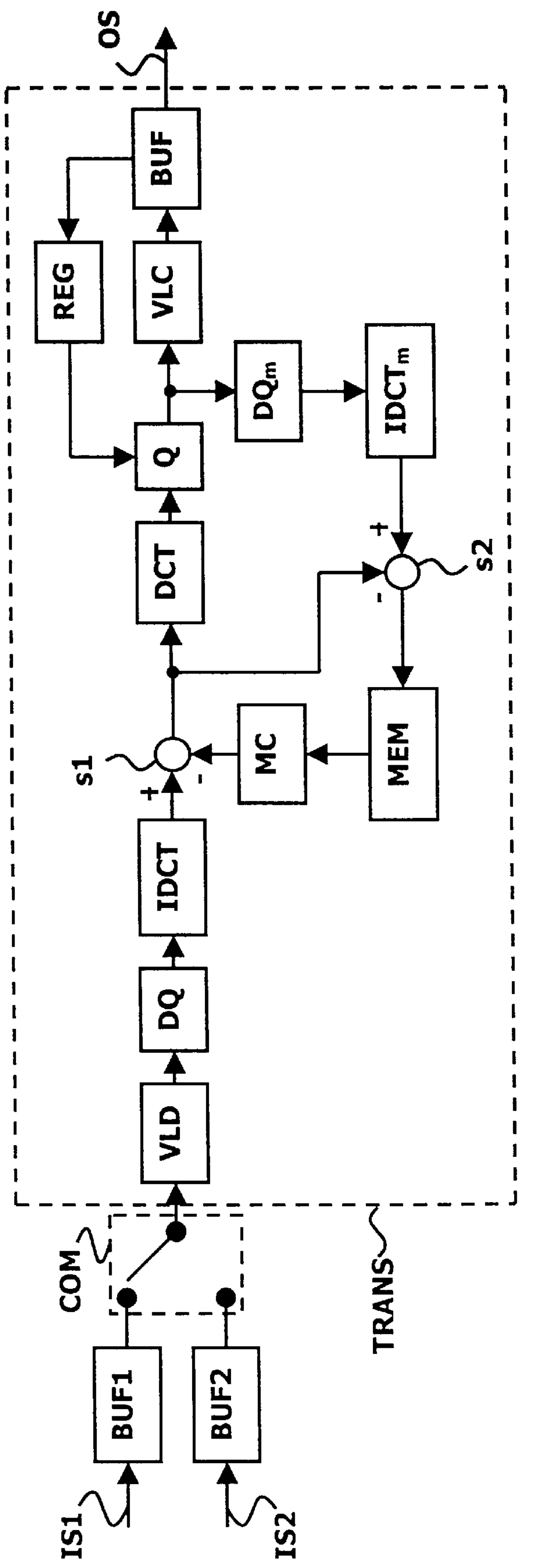

OS
BUF
REG
VLC
DQm
IDCTm
Q
DCT
+
-
s2
s1
+
-
MC
MEM
IDCT
DQ
VLD
COM
TRANS
BUF1
BUF2
IS1
IS2

FIG. 4

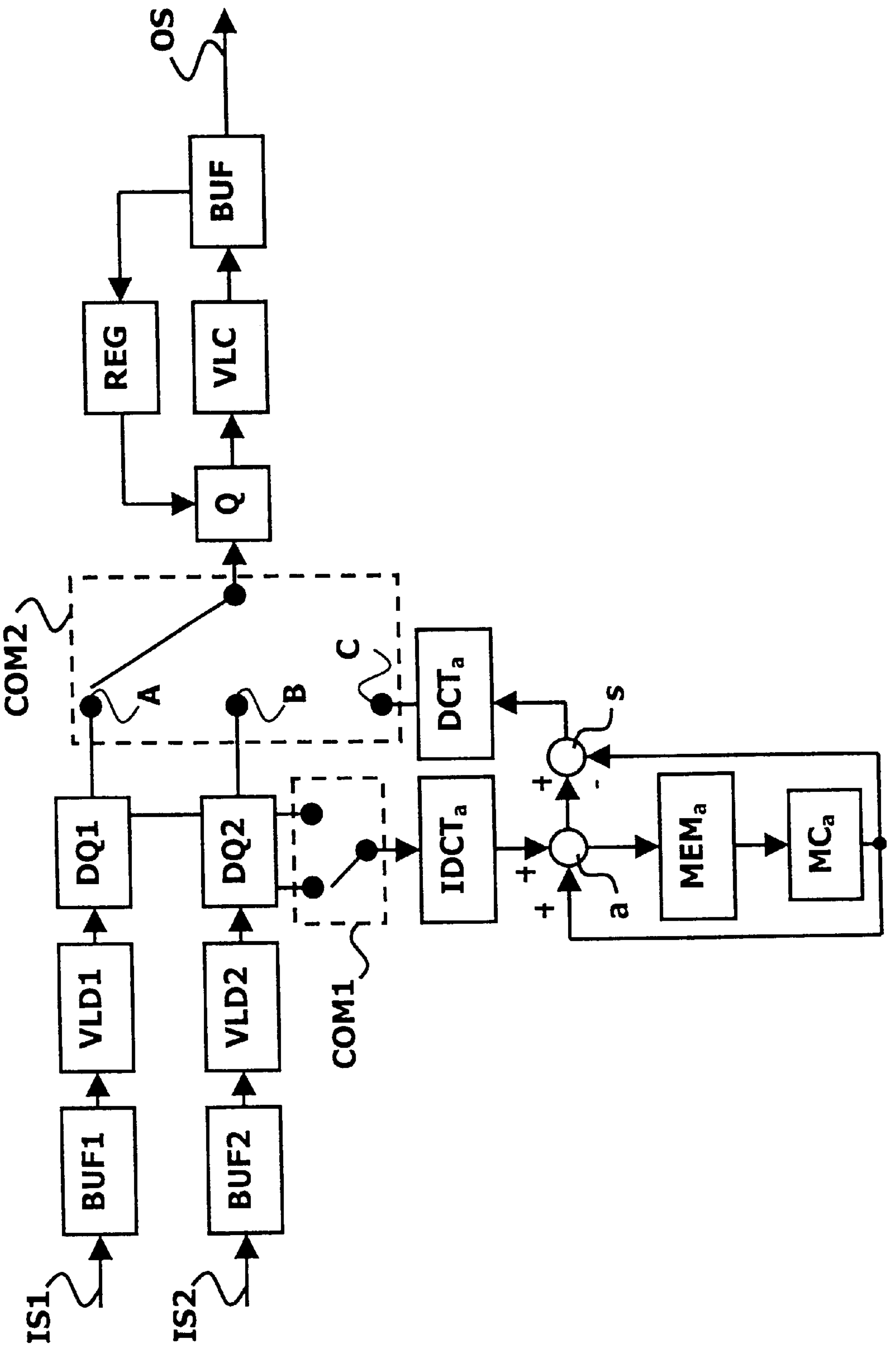

IS1
IS2
BUF1
BUF2
VLD1
VLD2
DQ1
DQ2
COM1
COM2
A
B
C
IDCT$_a$
DCT$_a$
MEM$_a$
MC$_a$
a
s
+
+
+
-
Q
REG
VLC
BUF
OS

FIG. 5

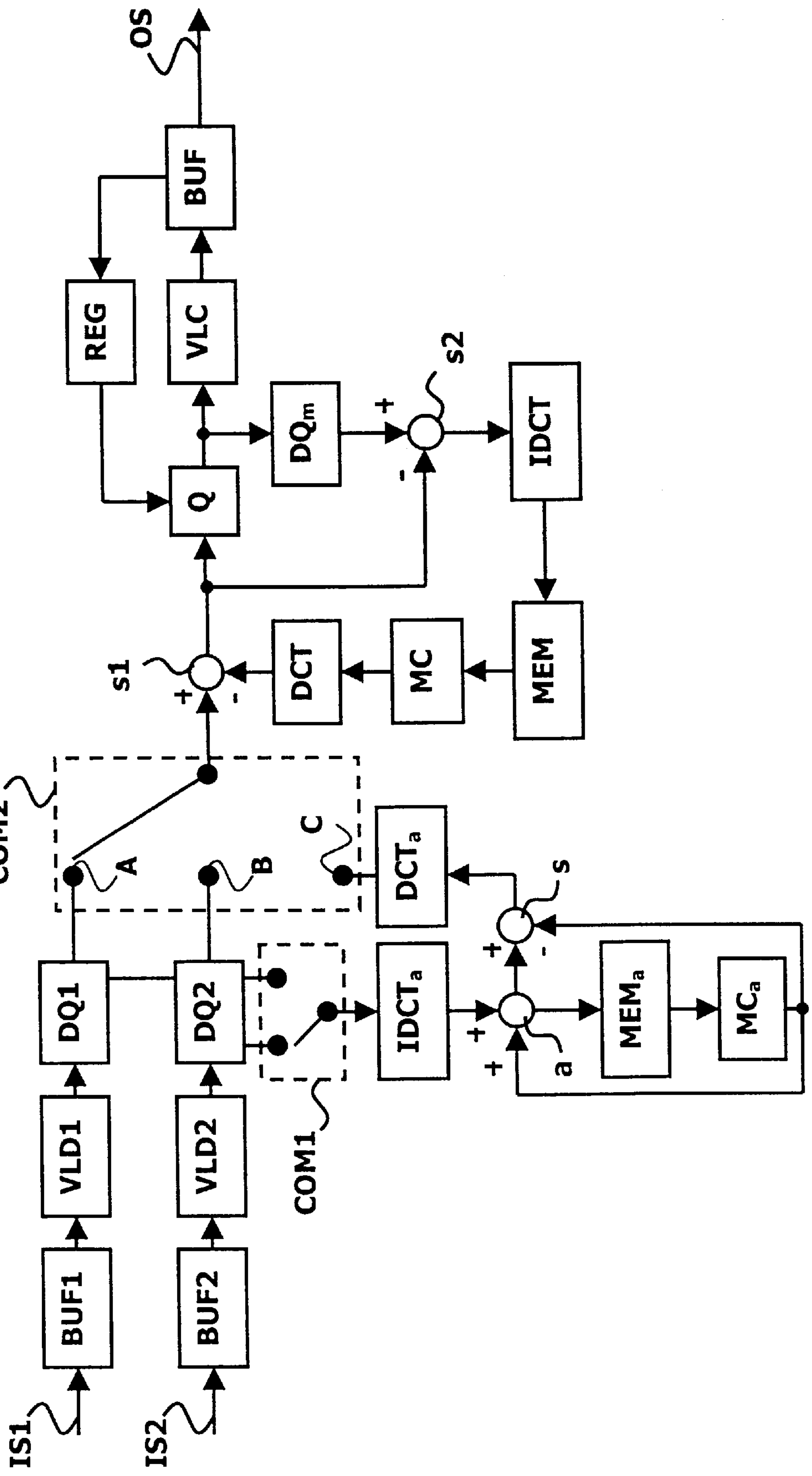

IS1
IS2
BUF1
BUF2
VLD1
VLD2
DQ1
DQ2
COM1
COM2
A
B
C
IDCT$_a$
DCT$_a$
MEM$_a$
MC$_a$
a
s
+
+
+
-
s1
+
-
DCT
MC
MEM
Q
REG
VLC
BUF
OS
DQ$_m$
s2
+
-
IDCT

FIG. 6

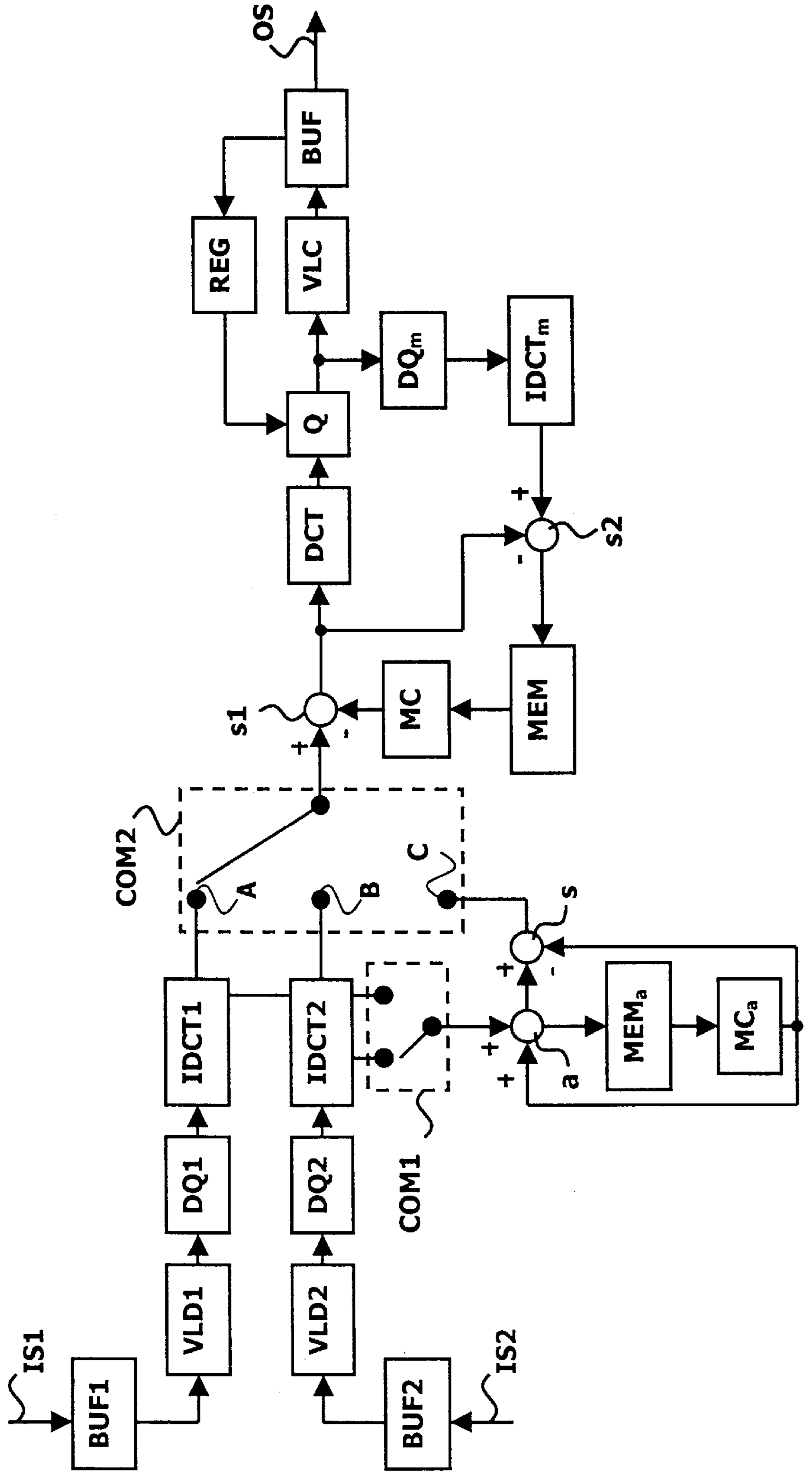

IS1
BUF1
VLD1
DQ1
IDCT1
IS2
BUF2
VLD2
DQ2
IDCT2
COM1
COM2
A
B
C
a
s
MEM_a
MC_a
s1
MC
MEM
DCT
Q
REG
VLC
BUF
OS
DQ_m
$IDCT_m$
s2
+
-

FIG. 7

SEAMLESS SWITCHING OF MPEG VIDEO STREAMS

FIELD OF THE INVENTION

The present invention relates to a method of and its corresponding device for switching from a first compressed data input stream to a second compressed data input stream, resulting in a compressed data output stream.

Such an invention can be useful, for example, for switching and editing MPEG compressed video signals.

BACKGROUND OF THE INVENTION

International patent application WO 99/05870 describes a method and device of the above kind. This patent application relates, in encoding/decoding systems, to an improved method of switching from a first encoded video sequence to a second one. In order to avoid underflow or overflow of the decoded buffer, a transcoding of the input streams is used to shift the temporal position of the switching point and to obtain at the output of the transcoders, streams containing an identical entry point and the same decoder buffer characteristics.

The previously described method has several major drawbacks. According to the background art, the output bit rate of each transcoder is equal to its input bit rate, which makes the switching method not very flexible. Moreover, said method implies that the first picture of the second video sequence just after the switch will be an Intra-coded (I) picture.

Finally, the solution of the background art is rather complex and costly to implement as the switching device needs two transcoders.

SUMMARY OF THE INVENTION

It is an object of the invention to provide a method of switching and its corresponding device that is both flexible and easy to implement.

To this end, the invention relates to a switching device as described in the field of the invention and comprising:

a buffer system intended to store the data contained in the first and second input streams,

control means intended to control the storage of the input streams in the buffer system in order to switch, at a switch request, from the first input stream to the second input stream using a commutation device,

and a transcoding system intended to provide the output stream in a seamless way from the output of the commutation device.

The present invention allows to switch from a first compressed data stream encoded at a bit rate R1 to a second compressed data stream encoded at a bit rate R2, the output stream resulting from the switch being encoded again, using the transcoding system, at a bit rate R where R may be different from R1 and R2. Thus, such a switching device has a flexible behavior.

The switching device according to the invention is also characterized in that:

the buffer system comprises a first buffer and a second buffer intended to store the data contained in the first and the second input stream, respectively,

the transcoding system comprises one transcoder,

the commutation device is controlled to switch from the output of the first buffer to the output of the second

buffer when said first buffer has transmitted a set of M pictures of the first input stream, said second buffer being controlled by the control means to transmit an I picture,

and said switching device comprises means for generating B pictures without forward predictions for a set of M pictures of the second input stream including said I picture.

As this switching device uses only one transcoder, its implementation will be less complex and less expensive.

Finally, the switching device according to the invention is characterized in that:

the buffer system comprises a first buffer and a second buffer intended to store the data contained in the first and the second input stream, respectively,

the transcoding system comprises, in association with each input stream, first means for decoding and second means for decoding,

the commutation device is controlled to switch from the first input stream after decoding by the first means to the second input stream after decoding by the second means when the first buffer has transmitted a set of M pictures of the first input stream, the second buffer being controlled by the control means to transmit an I picture or a P picture, which is re-encoded as an I picture using decoding-encoding means,

and said switching device comprises means for generating B pictures without forward predictions for a set of M pictures of the second input stream including said I picture.

Such a switching device allows to switch to a second compressed video stream that is starting with a P picture. Thus, the flexibility of the system is increased.

These and other aspects of the invention will be apparent from and elucidated with reference to the embodiments described hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will now be described, by way of example, with reference to the accompanying drawings, wherein:

FIG. **1** is a block diagram corresponding to a switching device according to the invention,

FIG. **2** is a block diagram corresponding to a switching device according to a first embodiment and comprising a transcoder using only requantization means,

FIG. **3** is a block diagram corresponding to a switching device according to a first embodiment and comprising a transcoder using motion compensation means,

FIG. **4** is a block diagram corresponding to a switching device according to a first embodiment and comprising a transcoder using improved motion compensation means,

FIG. **5** is a block diagram corresponding to a switching device according to a second embodiment and comprising a transcoding system using only requantization means,

FIG. **6** is a block diagram corresponding to a switching device according to a second embodiment and comprising a transcoding system using motion compensation means,

FIG. **7** is a block diagram corresponding to a switching device according to a second embodiment and comprising a transcoding system using improved motion compensation means.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention relates to an improved device for switching and editing of compressed data signals. It relates, more especially, to MPEG signals but is also applicable to any type of compressed data such as, for example, those provided by H-261 or H-263 standards of the International Telecommunication Union (ITU). The principle of the switching device according to the invention is depicted in FIG. **1**.

Such a switching device SW allows to switch from a first compressed data input stream **IS1** to a second compressed data input stream **IS2**, resulting in a compressed data output stream OS.

This switching device comprises a buffer system BS intended to store the data contained in the first and second input streams, and control means CONT which controls the storage of the input streams in the buffer system in order to switch, at a switch request SWR, from the first input stream to the second input stream, using a commutation device COM.

A transcoding system TS is intended to receive the data stream at the output of the commutation device and to provide the output stream in a seamless way. The use of a transcoding system allows to avoid an underflow or an overflow of the buffer of the decoder that will have to decode the output stream. Moreover, said transcoding system allows to encode the output stream at a bit rate R, where R may be different from the bit rate R**1** of the first input stream and the bit rate R**2** of the second input stream.

The present invention will now be described more specifically for MPEG video data switching. FIG. **2** is a block diagram corresponding to a first embodiment of a switching device of MPEG video streams. In this first embodiment, the switching device comprises:

- a first buffer BUF**1** and a second buffer BUF**2** intended to store the data contained in the first and the second input stream, respectively,
- a commutation device COM, and
- a transcoder TRANS.

The switching operation from a first video input stream to a second video input stream can be performed if the second input stream starts, in the order of transmission, with a picture with no reference to the past (Intra-coded (I) picture) and if the last presented picture of the first input stream, in the order of display, has no reference to the future (Predicted (P) picture or I picture). Furthermore, the bidirectional (B) pictures following, in the order of transmission, the first inserted picture of the second input stream shall not contain forward predictions, that is, the first inserted Group Of Pictures (GOP) of the second input stream has to be a closed GOP. For this purpose, the switching device according to the invention also comprises means for generating B pictures without forward predictions for the first set of M pictures of the second input stream transmitted after switching, M being the distance between two consecutively I or P pictures.

As a first consequence, if the second input stream does not start with a closed GOP, then the first B pictures following the I picture will be:

- either ejected
- or replaced by Uniform Color (UC) pictures obtained by freezing the last picture, in the order of display of the first input stream, or by freezing the first I picture of the second input stream.

As a second consequence, the commutation device is intended to switch from the output of the first buffer to the output of the second buffer when said first buffer has transmitted a set of M pictures, said second buffer being ready to transmit an I picture. To this end, the two buffers are at least N pictures long, where N is the distance between two consecutive I pictures and are filled using a writing pointer and read using a reading pointer, the writing and reading pointers being controlled by a controller. At a switch request, the reading of the current set of M pictures ($P_kB_{k-2}B_{k-1}$ or $I_kB_{k-2}B_{k-1}$ in the order of transmission and $B_{k-2}B_{k-1}P_k$ or $B_{k-2}B_{k-1}I_k$ in the order of display) in the first buffer is first completed, then the commutation device switches to the output of the second buffer while the reading pointer of the second buffer is positioned at the beginning of the current I picture.

The transcoder according to the invention comprises a Variable Length Decoding block VLD and a dequantization block DQ for decoding the incoming stream, connected in series to a quantization block Q and a Variable Length Coding block VLC for re-encoding the stream, and a buffer BUF. To prevent overflow or underflow of this buffer, a regulation REG is performed; the buffer occupancy is controlled by a feedback to the DCT coefficient quantization. When switching from a video sequence encoded at a bit rate R**1** to another one that has been separately encoded at a bit rate R**2**, the respective decoder buffer delays at the switching point do not match. The role of the transcoder is to compensate the difference between these buffer delays in order to provide the output stream OS in a seamless way. Furthermore, the encoded bit rate R of the output stream can be chosen by the user.

In this first embodiment, the first picture of the second input stream can only be an I picture, as this first picture must not have reference to previous pictures, which are included in the first input stream. Moreover, the switching operation is reversible, which means that, at a switch request, a switch can also be made from the second input stream to the first input stream.

The transcoder of FIG. **2** is a simple one which mainly contains requantization means. FIGS. **3** and **4** show a switching device comprising more complex transcoders using motion compensation means. Such motion compensation means are used to correct the error drift on P/B pictures that occurs when only requantization means are used.

In FIG. **3**, the transcoder comprises:

- a decoding channel comprising a Variable Length Decoding block VLD connected in series to a dequantization block DQ,
- an encoding-decoding channel comprising a quantization block Q connected in series to a Variable Length Coding block VLC, the output of the quantization block also being connected to an extra dequantization block DQ_m,
- an interface sub-assembly connected between the decoding channel and the encoding-decoding channel, and comprising:
  - a first subtractor s**1**, whose positive input receives the output of the decoding channel and whose output is connected to the input of the Q block,
  - a second subtractor s**2**, whose positive input receives the output of the DQ_m block and whose negative input is connected to the output of the first subtractor,
  - an Inverse Discrete Cosine Transform block IDCT, a frame memory MEM, a motion compensation block MC and a Discrete Cosine Transform block DCT, all connected in series between the output of the second subtractor and the negative input of the first subtractor, the motion compensation being performed from motion vectors representing the motion of each macro-block of the current picture relative to the corresponding macro-block of a previous picture in the transmission order.

In FIG. **4**, the transcoder is more sophisticated and comprises:

a decoding channel comprising a Variable Length Decoding block VLD connected in series to a dequantization block DQ and an Inverse Discrete Cosine Transform block IDCT,

an encoding-decoding channel comprising a Discrete Cosine Transform block DCT connected in series to a quantization block Q and a Variable Length Coding block VLC, the output of the quantization block also being connected to an extra dequantization block DQ_m followed by an extra Inverse Discrete Cosine Transform block $IDCT_m$,

an interface sub-assembly, connected between the decoding channel and the encoding-decoding channel, and comprising:

a first subtractor s**1**, whose positive input receives the output of the decoding channel and whose output is connected to the input of the DCT block,

a second subtractor s**2**, whose positive input receives the output of the $IDCT_m$ block and whose negative input is connected to the output of the first subtractor,

a frame memory MEM and a motion compensation block MC connected in series between the output of the second subtractor and the negative input of the first subtractor.

FIG. **5** is a block diagram corresponding to a second embodiment of the switching device. In this second embodiment, the switching device comprises:

a buffer system comprising a first buffer BUF**1** and a second buffer BUF**2**, said buffers being at least M pictures long,

a transcoding system comprising:

a first decoding channel, whose input corresponds to the output of the first buffer, comprising a first Variable Length Decoding block VLD**1** connected in series to a first dequantization block DQ**1**,

a second decoding channel, whose input corresponds to the output of the second buffer, comprising a second Variable Length Decoding block VLD**2** connected in series to a second dequantization block DQ**2**,

an encoding channel comprising a quantization block Q connected in series to a Variable Length Coding block VLC and a buffer BUF providing the encoding output stream OS, regulation means REG for controlling the buffer occupancy by a feedback to the quantization block,

a commutation device comprising a first commutator COM**1**, whose inputs are the outputs of the dequantization blocks DQ**1** and DQ**2** and which is connected, before switching from the first input stream to the second input stream, to the output of the DQ**2** block, and a second commutator COM**2** having three inputs A, B and C, whose input A is the output of the dequantization block DQ**1**, whose input B is the output of the dequantization block DQ**2** and whose output C is the input of the encoding channel,

and a decoder comprising

an Inverse Discrete Cosine Transform block $IDCT_a$,

an adder a, whose first input is the output of the $IDCT_a$ block and a subtractor s, whose positive input is the output of the adder,

a frame memory MEM_a and a motion compensation block MC_a connected in series, on the one hand to the output of the adder and, on the other hand to the second input of the adder and the negative input of the subtractor,

and a Discrete Cosine Transform block DCT_a, which receives the output of the subtractor and whose output is the third input C of the commutator COM**2**.

In comparison with the previous schemes, the decoder has been added and allows to switch from a first input stream to a second input stream at a P picture of said second input stream. For this purpose, the decoder decodes all the P picture of the second input stream arriving before the switch from the first input stream to the second input stream. During this step, the first input stream is transcoded. Once the user wants to switch to the second input stream, the last decoded P picture, provided at the output of the $IDCT_a$ block, is re-encoded as an I picture provided at the output of the adder. Furthermore, the B pictures following this new I picture are modified into B pictures having only backward vectors thanks to the motion compensation means MC_a and the subtractor. B pictures without forward predictions are, for example, uniform color pictures as previously described in the first embodiment.

As a consequence, at a switch request, the reading of the current set of M pictures in the first buffer is completed first. Then, the commutator COM**2** switches from input A to input C, the decoder being ready to transmit the decoded P picture that has been re-encoded as an I picture and the rest of the set of M pictures. Finally, the commutator COM**2** switches from C to B, the reading pointer of the second buffer being positioned at the beginning of the second set of M pictures.

In this second embodiment, the first picture of the second input stream can be an I picture or a P picture. The switching operation is also reversible, which means that a switch can be made, at a switch request, from the second input stream to the first input stream, the commutator COM**1** being connected, before the switch, to the output of the DQ**1** block and the commutator COM**2** being positioned at input B.

The transcoder of FIG. **5** is a simple one that mainly contains a requantization step. FIGS. **6** and **7** show a switching device comprising more complex transcoders using motion compensation means.

In FIG. **6**, the transcoder comprises:

an interface sub-assembly, connected between the second commutator and the encoding channel, and comprising:

a first subtractor s**1**, whose positive input receives the output of the second commutator and whose output is connected to the input of the Q block,

a second subtractor s**2**, whose positive input receives the output of a dequantization block DQ_m connected to the output of the Q block, and whose negative input is connected to the output of the first subtractor,

an Inverse Discrete Cosine Transform block IDCT, a frame memory MEM, a motion compensation block MC and a Discrete Cosine Transform block DCT, all connected in series between the output of the second subtractor and the negative input of the first subtractor.

In FIG. **7**, the transcoder comprises:

the two decoding channels described in FIG. **5** with, in addition, an Inverse Discrete Cosine Transform block IDCT**1** or IDCT**2** connected between the output of DQ**1** or DQ**2** block and the input A or B of the second commutator, respectively,

the encoding channel described in FIG. **5** with, in addition, a Discrete Cosine Transform block DCT located before the Q block,

a third decoding channel connected to the output of the Q block and comprising an extra dequantization block DQ_m followed by an extra Inverse Discrete Cosine Transform block $IDCT_m$,

an interface sub-assembly, connected between the second commutator and the encoding channel, and comprising:
a first subtractor **s1**, whose positive input receives the output of second commutator and whose output is connected to the input of the DCT block,
a second subtractor s**2**, whose positive input receives the output of the IDCT_m block and whose negative input is connected to the output of the first subtractor,
a frame memory MEM and a motion compensation block MC connected in series between the output of the second subtractor and the negative input of the first subtractor.

What is claimed is:

1. A device for switching (SW) from a first compressed data input stream (IS**1**) to a second compressed data input stream (IS**2**), resulting in a compressed data output stream (OS), said switching device comprising:
a buffer system (BS) to store the data contained in the first and second input streams,
control means (CONT) to control the storage of the input streams in the buffer system in order to switch, at a switch request (SWR), from the first input stream to the second input stream using a commutation device (COM),
and a transcoding system (TS) including a quantization block and a buffer, wherein occupancy of the buffer in the transcoding system is controlled by feedback to the quantization block to provide the output stream in a seamless way from the output of the commutation device.

2. A switching device for switching (SW) from a first compressed data input stream (IS**1**) to a second compressed data input stream (IS**2**), resulting in a compressed data output stream (OS), said switching device comprising:
a buffer system (BS) intended to store the data contained in the first and second input streams,
control means (CONT) to control the storage of the input streams in the buffer system in order to switch, at a switch request (SWR), from the first input stream to the second input stream using a commutation device (COM),
and a transcoding system (TS) to provide the output stream in a seamless way from the output of the commutation device,
wherein the buffer system comprises a first buffer (BUF**1**) and a second buffer (BUF**2**) intended to store the data contained in the first and the second input stream, respectively,
wherein the transcoding system comprises one transcoder,
the commutation device is controlled to switch from the output of the first buffer to the output of the second buffer when said first buffer has transmitted a set of M pictures of the first input stream, said second buffer being controlled by the control means to transmit an I picture,
and said switching device comprises means for generating B pictures without forward predictions for a set of M pictures of the second input stream including said I picture.

3. A switching device for switching (SW) from a first compressed data input stream (IS**1**) to a second compressed data input stream (IS**2**), resulting in a compressed data output stream (OS), said switching device comprising:
a buffer system (BS) to store the data contained in the first and second input streams,

control means (CONT) to control the storage of the input streams in the buffer system in order to switch, at a switch request (SWR), from the first input stream to the second input stream using a commutation device (COM),
and a transcoding system (TS) to provide the output stream in a seamless way from the output of the commutation device,
wherein the buffer system comprises a first buffer and a second buffer intended to store the data contained in the first and the second input stream, respectively,
wherein the transcoding system comprises, in association with each input stream, first means for decoding and second means for decoding,
the commutation device is controlled to switch from the first input stream after decoding by the first means to the second input stream after decoding by the second means when the first buffer has transmitted a set of M pictures of the first input stream, the second buffer being controlled by the control means to transmit an I picture or a P picture, which is re-encoded as an I picture using decoding-encoding means,
and said switching device comprises means for generating B pictures without forward predictions for a set of M pictures of the second input stream including said I picture.

4. A method of switching from a first compressed data input stream to a second compressed data input stream, resulting in a compressed data output stream, said method of switching comprising the steps of:
buffering, in which the data contained in the first and the second input stream are stored,
controlling the storage of the input streams during the buffering step in order to switch, at a switch request, from the first input stream to the second input stream,
transcoding the stream provided by the control step, the transcoding includes controlling occupancy of a buffer by feedback to DCT coefficient quantization in order to provide the output stream in a seamless way.

5. A method of switching from a first compressed data input stream to a second compressed data input stream, resulting in a compressed data output stream, said method of switching comprising the steps of:
buffering, in which the data contained in the first and the second input stream are stored,
controlling the storage of the input streams during the buffering step in order to switch, at a switch request, from the first input stream to the second input stream,
transcoding the stream provided by the control step in order to provide the output stream in a seamless way,
wherein the transcoding step comprises one transcoding channel,
the control step allows to switch, at a switch request, from the first input stream to the second input stream when the buffering step has transmitted a set of M pictures of the first input stream, the buffering step being controlled to transmit an I picture,
and said method of switching comprises a step of generating B pictures without forward predictions for a set of M pictures of the second input stream including said I picture.

6. A method of switching from a first compressed data input stream to a second compressed data input stream, resulting in a compressed data output stream, said method of switching comprising the steps of:

buffering, in which the data contained in the first and the second input stream are stored,

controlling the storage of the input streams during the buffering step in order to switch, at a switch request, from the first input stream to the second input stream,

transcoding the stream provided by the control step in order to provide the output stream in a seamless way

wherein the transcoding step comprises a first sub-step of decoding the first input stream and a second sub-step of decoding the second input stream,

the control step allows to switch, at a switch request, from the first input stream after the first decoding step to the second input stream after the second decoding step when the buffering step has transmitted a set of M pictures of the first input stream, the buffering step being controlled to transmit an I picture or a P picture, which is re-encoded as an I picture using a decoding-encoding step,

and said method of switching comprises a step of generating B pictures without forward predictions for a set of M pictures of the second input stream including said I picture.

7. A device for switching (SW) from a first compressed data input stream (IS1) to a second compressed data input stream (IS2), resulting in a compressed data output stream (OS), said switching device comprising:

a buffer system (BS) intended to store the data contained in the first and second input streams,

control means (CONT) intended to control the storage of the input streams in the buffer system in order to switch, at a switch request (SWR), from the first input stream to the second input stream using a commutation device (COM),

and a transcoding system (TS) intended to provide the output stream in a seamless way from the output of the commutation device,

means for generating B pictures without forward predictions for a set of M pictures of the second input stream including an I picture.

8. A method of switching from a first compressed data input stream to a second compressed data input stream, resulting in a compressed data output stream, said method of switching comprising the steps of:

buffering, in which the data contained in the first and the second input stream are stored,

controlling the storage of the input streams during the buffering step in order to switch, at a switch request, from the first input stream to the second input stream,

transcoding the stream provided by the control step in order to provide the output stream in a seamless way,

generating B pictures without forward predictions for a set of M pictures of the second input stream including an I picture.

* * * * *

EXHIBIT B

US006895118B2

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

(10) **Patent No.: US 6,895,118 B2**
(45) **Date of Patent: May 17, 2005**

(54) **METHOD OF CODING DIGITAL IMAGE BASED ON ERROR CONCEALMENT**

(75) Inventors: **Stephane Edouard Valente**, Paris (FR); **Cecile Dufour**, Paris (FR)

(73) Assignee: **Koninklijke Philips Electronics N.V.**, Eindhoven (NL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 564 days.

(21) Appl. No.: **10/086,741**

(22) Filed: **Mar. 1, 2002**

(65) **Prior Publication Data**

US 2003/0031261 A1 Feb. 13, 2003

(30) **Foreign Application Priority Data**

Mar. 6, 2001 (FR) 01 03047

(51) **Int. Cl.**^{7} **G06K 9/36**
(52) **U.S. Cl.** **382/232**
(58) **Field of Search** 382/232–233, 382/236, 238–239, 240, 248, 250–252; 348/384.1, 390.1, 391.1, 394.1, 395.1, 400.1–404.1, 407.1–416.1, 420.1, 421.1, 425.1, 430.1, 431.1; 375/240, 240.01, 240.02, 240.12–240.2, 240.24–240.28

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,455,629	A	*	10/1995	Sun et al.	375/240.27
6,359,121	B1	*	3/2002	Ebenezer et al.	534/634
6,445,823	B1	*	9/2002	Liang	382/232
6,480,543	B1	*	11/2002	Pau et al.	375/240.16
6,658,157	B1	*	12/2003	Satoh et al.	382/239
6,690,833	B1	*	2/2004	Chiang et al.	382/236

OTHER PUBLICATIONS

"Geometric–Structure–Based Error Concealment with Novel Applications in Block–Based Low–Bit–Rate Coding" by W. Zeng and B. Liu in IEEE Transactions on Circuits and Systems For Video Technology, vol. 9, No. 4, Jun. 1999.

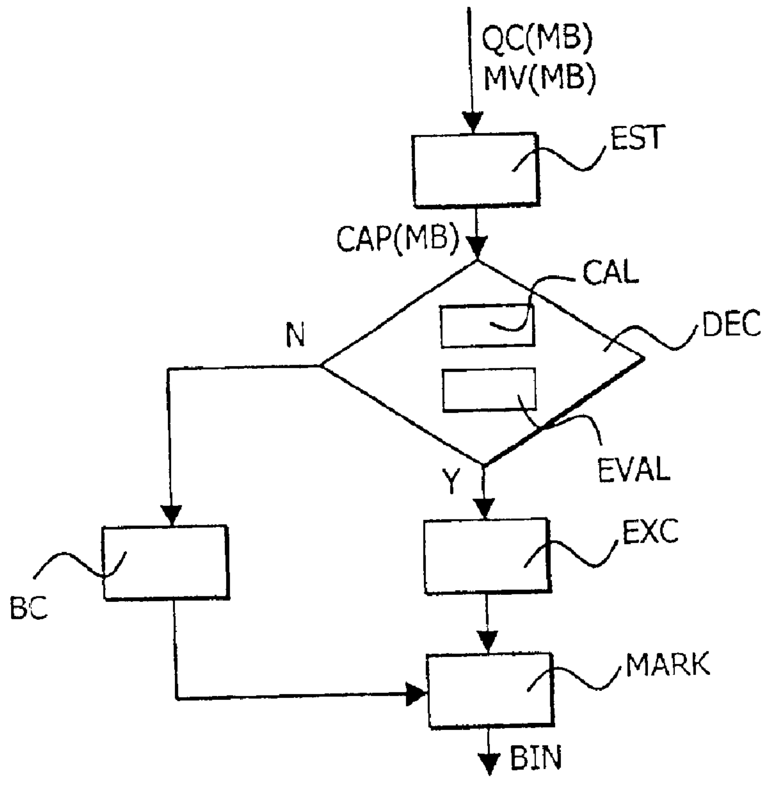
* cited by examiner

Primary Examiner—Jose L. Couso

(57) **ABSTRACT**

The invention relates to a method of coding a digital image comprising macroblocks in a binary data stream, comprising an estimation step, for macroblocks, of a capacity to be reconstructed by an error concealment method, a decision step for excluding macroblocks from the coding, a decision to exclude a macroblock from coding being made on the basis of the capacity of such macroblock to be reconstructed and a step of inserting a resynchronization marker into the binary data stream following the exclusion of one or more macroblocks.

10 Claims, 2 Drawing Sheets



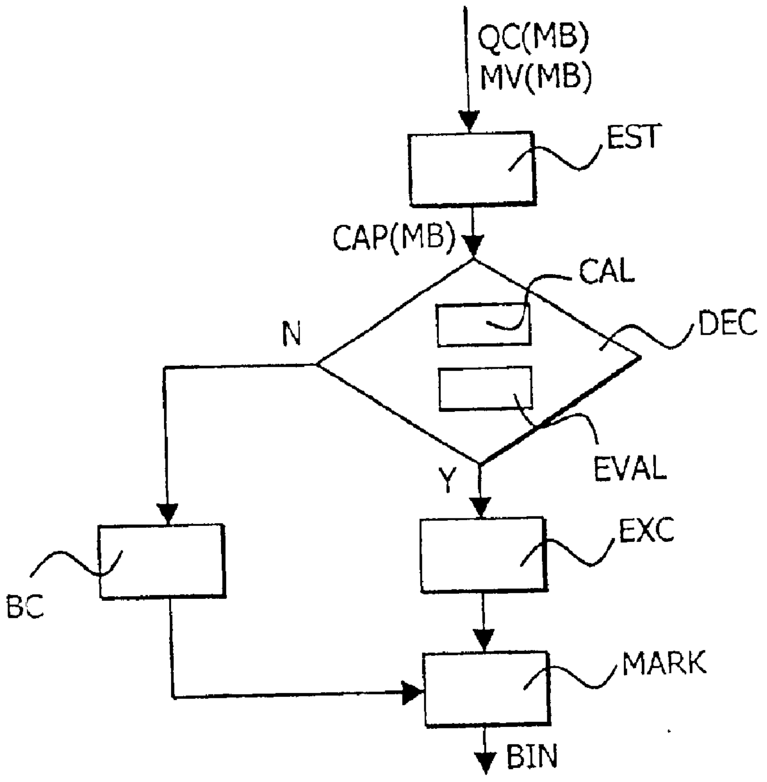


FIG.1

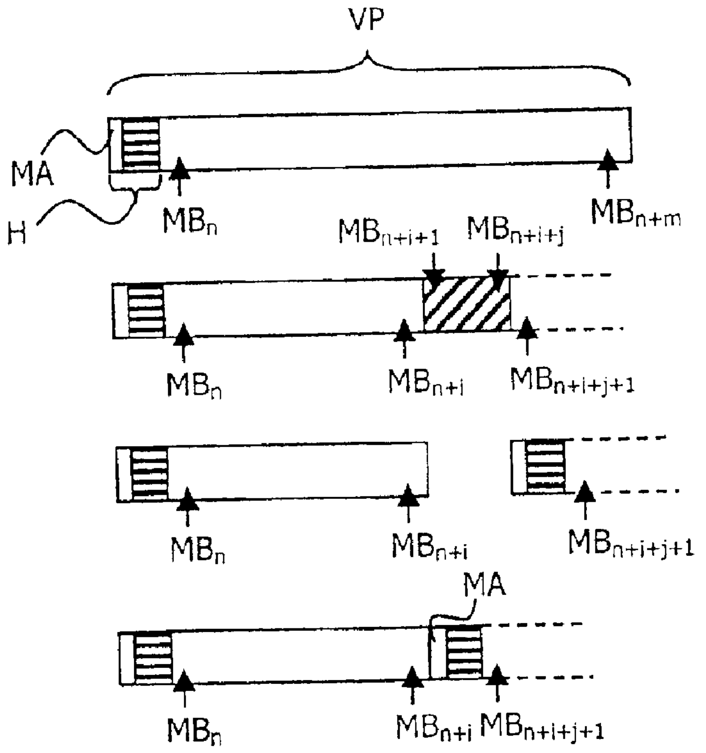


FIG.2a
FIG.2b
FIG.2c
FIG.2d

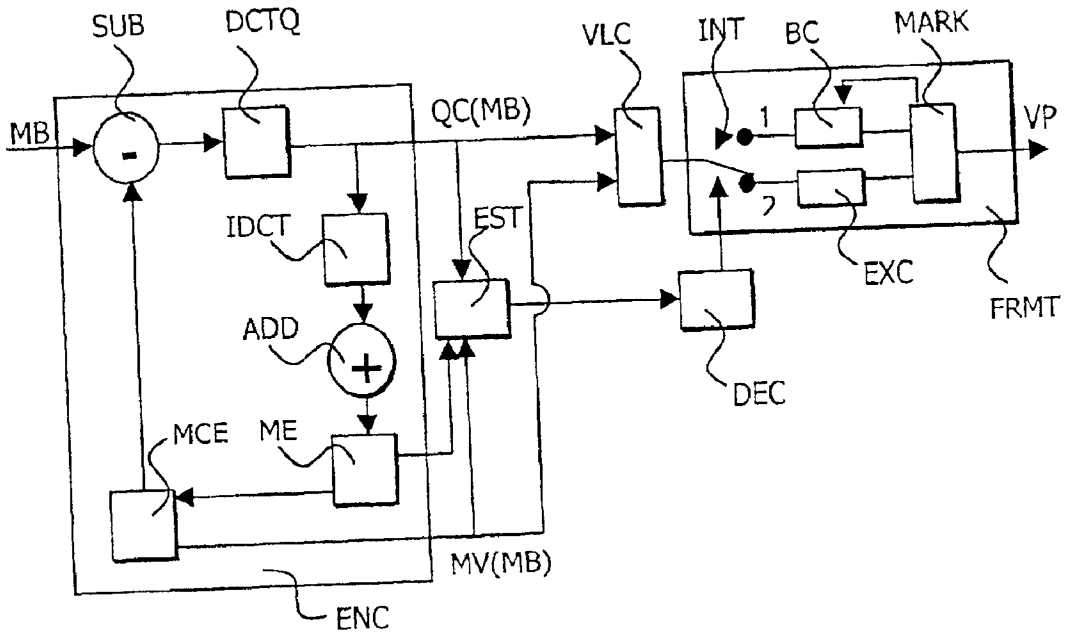

SUB
DCTQ
VLC
INT
BC
MARK
MB
QC(MB)
1
VP
-
2
IDCT
EST
EXC
FRMT
ADD
+
DEC
MCE
ME
MV(MB)
ENC

FIG.3

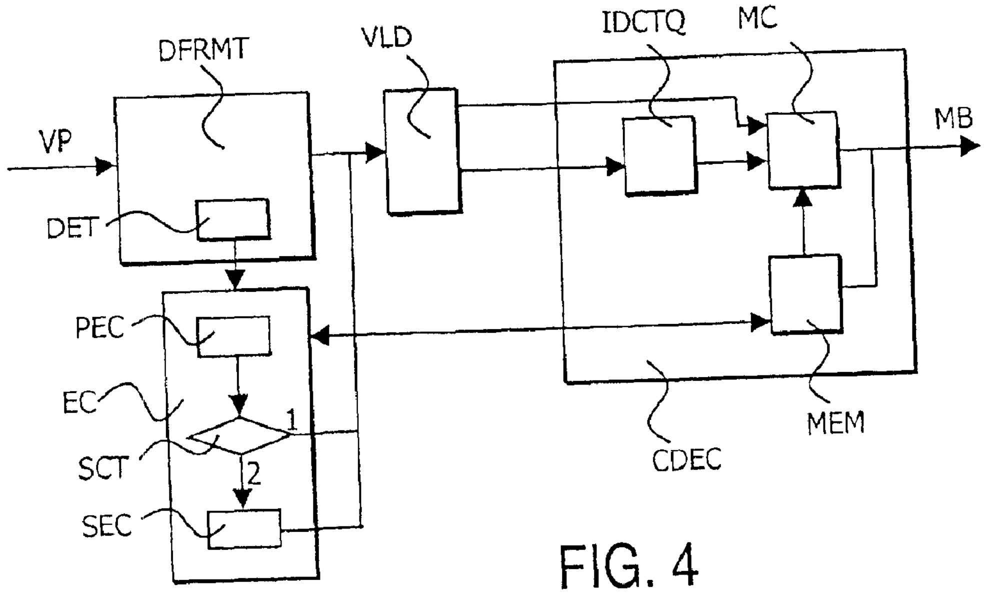

DFRMT
VLD
IDCTQ
MC
MB
VP
DET
PEC
EC
1
SCT
2
SEC
CDEC
MEM

FIG. 4

METHOD OF CODING DIGITAL IMAGE BASED ON ERROR CONCEALMENT

DESCRIPTION

The invention relates to a method of coding a digital image comprising macroblocks in a binary data stream, the method comprising:

an estimation step, for macroblocks, of capacity to be reconstructed via an error concealment method,

a decision step for macroblocks to be excluded from the coding, a decision to exclude a macroblock from coding being made on the basis of the capacity of such macroblock to be reconstructed.

A coding method of such type is known from the document "Geometric-Structure-Based Error Concealment with Novel Applications in Block-Based Low-Bit-Rate Coding" by W. Zeng and B. Liu in IEEE Transactions on Circuits and Systems For Video Technology, Vol. 9, No. 4, Jun. 1999. That document describes exclusions of blocks belonging to macroblocks, block combination, said macroblocks being capable of being intercoded or intracoded. That document proposes harmonizing this block exclusion with video coding standards, either, in a first solution, by replacing uncoded blocks with constant blocks, black blocks for example, subsequently detected by the receiver, or, in a second solution, by modifying the word that defines which blocks are coded within a macroblock, such modification taking place at the same time as a modification of the address words of the macroblocks when all the blocks in a macroblock are excluded. A certain number of bits are allocated to communicate the address of the excluded blocks in the interceded macroblocks.

The invention is associated with the following considerations:

The MPEG-4 standard defines a coding syntax and proposes a certain number of tools for managing transmission errors. These tools for managing transmission errors impose certain constraints. Among these tools the MPEG-4 standard proposes tools for resynchronizing the binary data stream which periodically insert resynchronization markers into the data stream. These markers are used by the receiver which is resynchronized thanks to them during decoding. When an error occurs in the data stream, the receiver cannot read the data any more until it detects a subsequent resynchronization marker. The set formed by the marker and data between this marker and the following marker, is called a video packet. The resynchronization marker is included in a header element of the video packet. The header element also contains the number of the first macroblock of the video packet, to allow spatial resynchronization, and parameters that permit the receiver to continue decoding. The numbers of the subsequent macroblocks are not present in the data stream. Resynchronization as defined in the MPEG-4 standard can thus be qualified as point resynchronization, because it only exists for certain items of data in a stream, the rest of the stream being interpreted passively. In this case it is therefore impossible to change the addresses of the macroblocks or indicate which blocks are not coded, according to the second solution proposed in the document cited in the foregoing. All macroblocks are thus decoded and placed sequentially, giving rise to graphical "lag" errors of image elements if macroblocks have been excluded. The first solution proposed in the document cited involves detection by the decoder of the constant blocks replacing the excluded blocks. No provision for such detection is made in the MPEG-4 syntax, and this will cause graphical errors on most receivers.

It is an object of the present invention to suggest a coding method that includes an exclusion of macroblocks having a certain capacity to be reconstructed from the coding compatible with coding standards which include point resynchronization means.

Indeed, a coding method as defined in the introductory paragraph is characterized according to the invention in that it also includes a step of inserting a resynchronization marker into the binary data stream after the exclusion of one or more macroblocks.

The resynchronization marker represents a certain number of bits in the data stream (at least between 17 and 23 bits). It is a further object of the present invention to reduce the binary data stream associated with the transmission of digital images by excluding macroblocks. Given the fact that according to the invention the exclusion of one or, more generally, several macroblocks leads to the insertion of a resynchronization marker which represents a certain number of bits, this exclusion of macroblocks can contribute nothing in terms of reducing the size of the binary data stream.

In a particularly advantageous embodiment, the coding method is characterized in that the decision step includes a substep of evaluation of the reduction of the binary data stream effected by an exclusion of macroblocks, the decision to exclude macroblocks being made as a function of the reduction of the binary data stream resulting from said exclusion.

The present invention may be implemented in a coder, for example a video coder. The present invention also relates to a data stream such as is produced via a method according to the invention. In consequence, the invention also relates to a decoding method and a decoder that allows correct decoding of a data stream such as is produced by a method according to the invention. Finally, the invention relates to computer programs for implementing the various steps of the method according to the invention.

These and other aspects of the invention are apparent from and will be elucidated, by way of non-limitative example, with reference to the embodiment(s) described hereinafter.

IN THE DRAWINGS

FIG. **1** is a functional diagram representing the various steps of a coding method of a digital image comprising macroblocks in a binary data stream.

FIG. **2** shows the effect of the method according to the invention on a stream comprising coded data of a digital image comprising macroblocks.

FIG. **3** is a schematic diagram of a video coder according to the invention.

FIG. **4** is a schematic diagram of a video decoder according to the invention.

FIG. **1** is a functional diagram representing the various steps of a method of coding a digital image containing macroblocks MB in a binary data stream according to the invention. In the embodiment illustrated here, the method according to the invention relates specifically to the portion of the coding that is performed on macroblocks MB that have already been converted in the form of a set of quantized coefficients QC(MB) and motion vectors MV(MB). It is these quantized coefficients QC(MB) and these motion vectors MV(MB) that are inserted at the beginning of the method in this embodiment. A first estimation step EST estimates a capacity CAP(MB) of the macroblocks to be reconstructed by an error concealment method. The estimation of this capacity may include various criteria.

These criteria correspond to the error concealment means available in the decoders. Accordingly, two major classes of error concealment are possible: spatial error concealment and temporal error concealment. In particular, the homogenous regions and the regions of homogenous motion tend to manifest a certain capacity to be reconstructed by an error concealment method.

For example, a first set of criteria relates to the homogenous regions and thus performs spatial error concealment: adjacent repetition of similar macroblocks in a sequence of macroblocks, or the facility with which a macroblock can be reconstructed by spatial interpolation from its neighbors. These two criteria may be practically and simply evaluated, for example, by calculating a difference between the macroblock that is considered for exclusion and one or more macroblocks that are either adjacent or spatially interpolated from the neighboring macroblocks, which calculation is performed as part of the estimation step EST.

Another set of criteria is associated with the regions of homogenous motion and performs temporal error concealment. For example, macroblocks belonging to regions of homogenous motion can thus be excluded, while the motion vectors of the neighboring macroblocks can be used to interpolate the excluded macroblocks. This criterion may be evaluated by calculating a difference between motion vectors of neighboring macroblocks and between quantization coefficients of a residual signal of the macroblocks from one image to the next.

The capacity CAP(MB) may thus be estimated by very diverse means that are known to one skilled in the art. The capacity CAP(MB) coming from the estimation step EST, may be in binary form or may be a number whose value is determined, for example, by the degree to which the macroblock considered for exclusion differs from an interpolated macroblock.

The method according to the invention then includes a decision step DEC regarding exclusion of macroblocks from coding. This decision is made on the basis of the capacity CAP(MB) of the macroblocks to be reconstructed. If capacity CAP(MB) is binary, the macroblock is excluded for a certain bit value, if capacity CAP(MB) is a number, the macroblock is excluded, for example, for capacity values that exceed a predetermined threshold. This predetermined threshold may be fixed or modulated depending on the resources available for transmission as will be described in the following. These conditions regarding capacity CAP (MB) define "good" reconstruction capacity in the following of the description.

In a particularly advantageous embodiment of the invention, the decision step comprises a substep wherein the reduction in size of the binary data stream resulting of an exclusion of macroblocks is evaluated, the decision on whether to exclude macroblocks being based on a criterion of the reduction of the binary data stream such exclusion. This step is justified by the fact that the insertion of the resynchronization marker entails insertion of a complete header element, which represents a certain number of bits in the data stream (at least between 17 and 23 bits). A further object of the invention is to reduce the size of the binary data stream associated with the transmission of images by excluding macroblocks. Given the fact that according to the invention the exclusion of one or, more generally, several macroblocks leads to the insertion of a complete header element, which represents a certain number of bits, said exclusion of macroblocks cannot contribute anything to the reduction of the binary data stream. An evaluation of the reduction of the binary data stream effected by the exclusion of macroblocks therefore serves a practical purpose. In this step EVAL, the number of bits saved by the exclusion of a certain number of macroblocks is evaluated, this number is then compared, in the decision step DEC for example, with the number of bits represented by the insertion of a header element. The decision to exclude is made when the reduction of the binary data stream caused by the exclusion from coding of the macroblocks is not zero.

The method according to the invention advantageously includes a calculation step CAL of an output rate of the binary data stream, the decision to exclude macroblocks being made on the basis of this output rate of the binary data stream. This calculation step CAL is performed in conjunction with transmission means of the coded macroblocks that transport the binary data stream.

Said steps CAL, EST and EVAL may also be combined: for example, the result of the calculation step CAL may influence the value of the threshold that determines "good" reconstruction capacity of macroblocks, said threshold becoming increasingly stringent as the available stream becomes greater. It is thus possible to consider a step that combines the results of the evaluation step EVAL and those of the calculation step CAL for determining a threshold value for the capacities CAP(MB) beyond which macroblocks having a capacity CAP(MB) higher than this threshold are excluded in the decision step DEC.

Depending on the result of the decision step DEC, the macroblock is either inserted into the video packet in a step BC (case N) or excluded from coding in a step EXC (case Y). In step BC, the bits are counted to trigger the insertion of a resynchronization marker in a step MARK when the video packet is of sufficient size. After each step EXC, a resynchronization marker is inserted into the binary data stream in step MARK. Here, the term "synchronization marker" must be interpreted generally to include, for example in the MPEG-4 standard, such conventional markers as RESYNC, VOPStart (start of a temporal instance (plan) of a video object), GOVStart (start of a group of temporal instances of a video object), EOS (end of video session). At the end of the method a binary data stream BIN is thus obtained.

It should be noted that the way of constitution of video packets may arbitrarily use a data partitioning and that the invention is generally unaffected by the use or not or not of a data partitioning.

It should be noted that the MPEG-4 standard already proposes not to code certain macroblocks in a video object or, more generally, in a video image, indicating this absence of coding by the presence of an "uncoded" flag. The presence of this flag is interpreted by the decoders which replace the uncoded macroblock with the macroblock located in the same position in a preceding instance of the video object. In general, the instance immediately preceding the instance in question is used. As a consequence, this flag can only be used for P coded images, for which a preceding instance is available and implicitly echoed in B coded images. The insertion of a flag of this nature is therefore only useful for regions having a motion vector close to zero and for which the texture has not changed significantly from one image or instance to the next. The exclusion of macroblocks from coding according to the invention does not entail the insertion of any specific flag and the exclusion of macroblocks from coding is thus possible for all modes of I, P or B coding.

FIG. 2 illustrates the effect of the method according to the invention on a binary data stream comprising coded data of a digital image or a video object including macroblocks. FIG. 2*a* represents a video packet VP with a header element H including a resynchronization marker MA. The periodicity of the markers may be based on a number of macroblocks or, more advantageously, on a number of bits. The latter solution, preferably selected by the MPEG-4 standard, allows the markers to be distributed uniformly throughout the stream. In all cases, a resynchronization marker and the data that follow up to the next resynchronization marker define a video packet. When the periodicity of the markers is based on a number of bits, the length of these video packets is determined by a mechanism according to which, if the number of bits in the current video packet exceeds a threshold value, a new video packet is created at the start of the following macroblock by the insertion of a resynchronization marker.

In the MPEG-4 standard, information necessary for restarting the decoding procedure in the receiver, as well as the number n of the first macroblock MB(n) of the video packet and the quantization parameters necessary for decoding this first macroblock, are included in a header element that also contains the resynchronization marker. The number n of the first macroblock allows spatial resynchronization to be performed and the quantization parameters allow the differential decoding procedure to be resynchronized. The numbers of the subsequent macroblocks are not indicated.

In FIG. 2*b* the macroblocks having good capacity to be reconstructed are designated by slanted hatching. They are the j macroblocks MB_{n+i+l} to MB_{n+i+j}. When these macroblocks are inserted in the method described in the foregoing and in FIG. **1**, the construction of the video packet is interrupted by the exclusion decision EXC represented schematically in FIG. 2*c*. Here, the schematic representation illustrates a case without data partitioning, where the macroblocks follow one another in a simple, serial stream. Data partitioning does not contradict the principle of the invention. The resulting binary data stream in such case is shown in FIG. 2*d*. A resynchronization marker MA and the associated header element have been inserted in the stream at the point where the first one of the excluded macroblocks should have been, and before macroblock $MB_{n+i+j+l}$. Here, the reduction in the size of the binary data stream caused by the insertion of resynchronization marker MA and the associated header element is not zero according to FIG. **2**: the bloc representing excluded macroblocks MB_{n+i+l} to MB_{n+i+j} is larger than the size of the inserted header element. If an evaluation step EVAL is included in the method, this exclusion of macroblocks is effected; such exclusion would not take place if an evaluation step EVAL were present and if the block representing the macroblocks had been smaller than the block including the header element.

Since the binary data stream includes coded data of a digital image comprising macroblocks, said binary data stream being such that macroblocks MB_{n+i+l} to MB_{n+i+j} are not coded in the binary data stream for at least one point in the binary data stream and since such uncoded macroblocks are capable of being reconstructed by an error concealment method, said binary data stream is thus characterized according to the invention in that a resynchronization marker MA is present in the binary data stream at the location in the binary data stream where the macroblocks are not coded.

FIG. **3** is a schematic diagram of a video coder according to the invention. The video coder represented in FIG. **3** receives graphic data (images) in the form of macroblocks MB. These graphic images are converted as part of a first coding stage ENC in which the information contained in the macroblocks is coded into quantized coefficients QC(MB)

and motion vectors MV(MB) by a series of operations such as addition ADD, subtraction SUB, transformation DCTQ and IDCTQ, and motion estimation and compensation MCE. A memory MEM enables certain of these operations to be performed and serves to store the data (for example image data). The macroblocks may be interceded or intracoded. The quantized coefficients QC(MB) and the motion vectors MV(MB) are sent, on the one hand, to a variable length coder VLC and, on the other, to an estimation module EST. As is shown in FIG. **3**, the estimation module EST is advantageously coupled to memory MEM of the first coding stage ENC. In the coder VLC, the quantized coefficients QC(MB) and the motion vectors MV(MB) are converted to a first form for subsequent formatting. The capacity of the macroblocks to be reconstructed via an error concealment method is estimated in the estimation module EST. The capacity value is then sent to a decision module DEC connected to an interrupter INT which belongs to a formatter FRMT which formats the data it receives in the output format of the coder VLC. Depending on the capacity value CAP(MB) and, advantageously depending on the result of an evaluation step with respect to the reduction in size of the binary data stream and of a calculation step with respect to the output rate of the binary data stream, the decision module DEC switches interrupter INT between two coding paths. When the decision module DEC switches the interrupter to position **1**, the video packet is constructed in conventional manner, by counting the bits or the macroblocks in a module BC. The insertion of a header element including a resynchronization marker is then effected by a marking module MARK when the video packet reaches the required size. When the decision module DEC switches the interrupter to position **2**, the macroblocks which the decision module has decided to exclude are excluded from the coding by a module which carries out an exclusion step EXC. After each exclusion step EXC, a header element including a resynchronization marker is inserted in the binary data stream by module MARK. The video packets VP corresponding to those described in FIGS. **2***a* and **2***d* are thus obtained from formatter FRMT.

The arrangement of the various modules in this coder corresponds to a specific embodiment, not intended to exclude other embodiments that may be apparent to one skilled in the art.

FIG. **4** is a schematic diagram of a video decoder according to the invention. The decoder receives the coded binary data stream, for example in the form of video packets VP represented in FIG. **2**, via a transmission channel (not shown). It responds by providing a sequence of decoded macroblocks MB. The decoder includes a deformatter DFRMT, a variable length word decoder VLD, a decoding stage CDEC. The decoding stage CDEC includes an inverse transform IDCTQ , a motion compensator MC and a memory MEM. Deformatter DFRMT comprises a detection module DET for the purpose of detecting uncoded macroblocks at at least one point in the binary data stream. The detection module DET is coupled to an error concealment module EC which is designed to be particularly activated for uncoded macroblocks that have been detected in the detection step DET. The error concealment module EC activated thereby reconstructs the macroblock.

The decoder according to the invention is characterized in that the detection module DET of uncoded macroblocks comprises a detection submodule for the purpose of detecting irregular intervals between resynchronization markers. Accordingly, the substep detects uncoded macroblocks using a detection of a resynchronization marker at the very point where macroblocks are not coded. In an embodiment of the invention relating to video packets formed by counting the macroblocks to achieve the required size, the decoder according to the invention counts the macroblocks present in the preceding video packet, starting at each resynchronization marker and, using the number of the first macroblock of the video packet it has just received and the number of the first macroblock of the video packet that starts, it deduces that same macroblocks have not been coded, thus detecting the insertion of a resynchronization marker at the point where the macroblocks have not been coded. The detection module DET may also be used for detecting, for example, errors in the binary data stream, said errors being concealed by the error concealment module coupled to said detection module DET. The reconstructed macroblocks are inserted into the output data of the deformatter DFRMT according to the corresponding sequence of the image or video object packet. However, such insertion of reconstructed data may be effected at several points or steps in the decoding method depending on the effectively reconstructed data.

The decoder presented here thus implements a method of decoding a binary data stream including coded data of a digital image with macroblocks, including a step of detecting macroblocks that are uncoded in at least one point of the binary data stream, an error concealment step EC principally activated for uncoded macroblocks detected in the detection step DET, characterized in that the step of detecting uncoded macroblocks includes a detection substep for the purpose of detecting irregular intervals between the resynchronization markers.

The decoding method described here may be applied to the standards MPEG-4, H26L and others.

In an advantageous embodiment illustrated in FIG. **4**, the error concealment module EC includes first means PEC for primary reconstruction, that is to say, for example, temporal reconstruction of the error, means SCT for appraising this first reconstruction which decides whether to modify (as in case **2**) the error reconstruction or validate (as in case **1**) the first reconstruction, second means SEC for secondary reconstruction, that is to say, for example, spatial reconstruction, which is activated when the appraisal step decides upon modification of the error reconstruction. For example, an uncoded macroblock belonging to an internally coded image (I for Intracoded) will be better corrected by spatial error concealment, whereas an uncoded macroblock belonging to an externally coded image (P or B for interceded) will be better corrected by spatial or temporal error concealment. The advantageous embodiment presented here thus allows to obtain optimized reconstruction by trying and testing various types of error concealment. One versed in the art may thus employ various means for reconstruction followed by evaluation tests of the quality of the reconstruction (spatial continuity tests . . .) according to combinations of varying complexity without exceeding the scope of the invention.

There are many ways to implement the functions disclosed in the method steps according to the invention by the use of software and/or hardware available to a person of ordinary skill in the art. For this reason, the Figures are schematic in nature. Accordingly, whereas the Figures illustrate various functions carried out by various blocks, this is not to say that a single unit of software and/or hardware may not carry out several functions. This does not exclude either that a combination of software and/or hardware means permits to carry out a single function.

It follows that many modifications may be effected by a person skilled in the art without thereby exceeding the intent and scope defined in the following claims.

What is claimed is:

1. A method of coding a digital image comprising macroblocks in a binary data stream, the method comprising:

an estimation step, for macroblocks, of a capacity to be reconstructed via an error concealment method,

a decision step for macroblocks to be excluded from the coding, a decision to exclude a macroblock from coding being made on the basis of the capacity of such macroblock to be reconstructed,

characterized in that it also includes a step of inserting a resynchronization marker into the binary data stream after the exclusion of one or more macroblocks.

2. A coding method as claimed in claim **1**, characterized in that the decision step includes a substep of evaluation of the reduction of the binary data stream effected by exclusion of the macroblocks, the decision to exclude macroblocks being made as a function of a reduction of the binary data stream resulting from such exclusion.

3. A coding method as claimed in one of the claims **1** and **2**, characterized in that it includes a calculation step of a binary data stream output rate, the decision to exclude macroblocks being made on the basis of this binary data stream output rate.

4. A coder for the purpose of coding a digital image comprising macroblocks in a binary data stream, comprising

an estimation module for the purpose of estimating a capacity of macroblocks to be reconstructed by an error concealment method,

a decision module intended to decide upon an exclusion of the coding for macroblocks, a decision to exclude a macroblock being made on the basis of the capacity of said macroblock to be reconstructed,

characterized in that it also includes a module for inserting a resynchronization marker in the binary data stream following the exclusion of one or more macroblocks.

5. A coding method as claimed in claim **3**, characterized in that it includes one or more modules for the purpose of carrying out the characteristic steps of one of the claims **2** and **3**.

6. A coded data of a digital image including macroblocks embedded in a, binary data stream, the macroblocks are not coded in the binary data stream in at least one location of the binary data stream, said uncoded macroblocks having a capacity to be reconstructed by an error concealment method,

characterized in that a resynchronization marker is present in the binary data stream at the point where macroblocks are not coded.

7. A method of decoding a binary data stream containing coded data of a digital image including macroblocks, said binary data stream containing resynchronization markers at regular intervals, including:

a detection step for uncoded macroblocks in at least one location of the binary data stream,

an error concealment step notably activated for uncoded macroblocks which are detected in the detection step,

characterized in that the detection step for uncoded macroblocks includes a detection substep for the purpose of detecting irregular intervals between the resynchronization markers.

8. A decoder for decoding a binary data stream containing coded data of a digital image comprising macroblocks, including:

a detection module for detecting uncoded macroblocks in at least one location of the binary data stream,

an error concealment module intended to be notably activated for the uncoded macroblocks that are detected by the detection module, characterized in that the detection module for uncoded macroblocks includes a detection submodule for the purpose of detecting irregular intervals between the resynchronization markers.

9. A "computer program" product for a coder comprising a series of functions and a collective resource that the functions access, characterized in that the "computer program" product includes a set of instructions which, when loaded into such a coder, run the method claimed in one of the claims **1** to **3** with respect to the coder.

10. A "computer program" product for a decoder comprising a series of functions and a collective resource that the functions access, characterized in that the "computer program" product includes a set of instructions which, when loaded into such a decoder, run the method as claimed in claim **7** with respect to the decoder.

* * * * *

EXHIBIT C

US006519005B2

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

(10) **Patent No.: US 6,519,005 B2**
(45) **Date of Patent: Feb. 11, 2003**

(54) **METHOD OF CONCURRENT MULTIPLE-MODE MOTION ESTIMATION FOR DIGITAL VIDEO**

(75) Inventors: **Michael Bakhmutsky**, Spring Valley, NY (US); **Karl Wittig**, New York, NY (US)

(73) Assignee: **Koninklijke Philips Electronics N.V.**, Eindhoven (NL)

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

(21) Appl. No.: **09/303,316**

(22) Filed: **Apr. 30, 1999**

(65) **Prior Publication Data**

US 2002/0176500 A1 Nov. 28, 2002

(51) **Int. Cl.**[7] **H04N 7/18**
(52) **U.S. Cl.** **348/415**; 375/240.17
(58) **Field of Search** 375/240.11–240.17

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,412,435 A		5/1995	Nakajima	348/699
5,813,197 A		9/1998	Chan et al.	348/416
5,905,542 A	*	5/1999	Linzer	348/699
5,963,673 A	*	10/1999	Kodama et al.	382/239
6,049,362 A	*	4/2000	Butter et al.	348/699
6,081,622 A	*	6/2000	Carr et al.	382/236
6,144,323 A	*	11/2000	Wise	341/76

FOREIGN PATENT DOCUMENTS

EP	0654946 A1	5/1995	....	H04N/7/13
EP	0658057 A2	6/1995	....	H04N/7/36
EP	0695097 A2	1/1996	....	H04N/7/50
EP	0898426 A1	2/1999	....	H04N/7/30

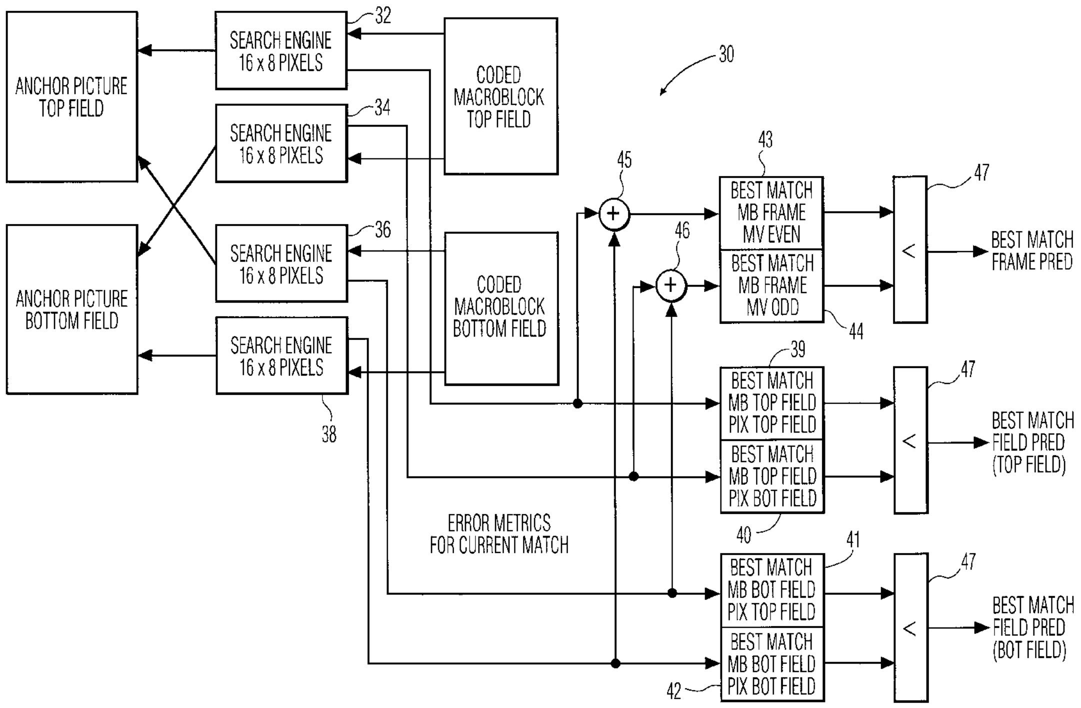
* cited by examiner

Primary Examiner—Andy S. Rao
(74) *Attorney, Agent, or Firm*—Russell Gross

(57) **ABSTRACT**

A method for motion coding an uncompressed digital video data stream such as an MPEG-2 digital video data stream. The method includes the steps of comparing pixels of a first pixel array in a picture currently being coded with pixels of a plurality of second pixel arrays in at least one reference picture and concurrently performing motion estimation for each of a plurality of different prediction modes in order to determine which of the prediction modes is an optimum prediction mode determining which of the second pixel arrays constitutes a best match with respect to the first pixel array for the optimum prediction mode, and, generating a motion vector for the first pixel array in response to the determining step. The method is implemented in a device such as a motion estimation search system of a digital video encoder. In one embodiment, the method and device are capable of concurrently determining performing motion estimation in each of the six different possible prediction modes specified by the MPEG-2 standard.

42 Claims, 8 Drawing Sheets


ANCHOR PICTURE TOP FIELD
ANCHOR PICTURE BOTTOM FIELD
SEARCH ENGINE 16 x 8 PIXELS
SEARCH ENGINE 16 x 8 PIXELS
SEARCH ENGINE 16 x 8 PIXELS
SEARCH ENGINE 16 x 8 PIXELS
32
34
36
38
CODED MACROBLOCK TOP FIELD
CODED MACROBLOCK BOTTOM FIELD
30
43
45
46
BEST MATCH MB FRAME MV EVEN
BEST MATCH MB FRAME MV ODD
44
47
BEST MATCH FRAME PRED
39
BEST MATCH MB TOP FIELD PIX TOP FIELD
BEST MATCH MB TOP FIELD PIX BOT FIELD
40
BEST MATCH FIELD PRED (TOP FIELD)
ERROR METRICS FOR CURRENT MATCH
41
BEST MATCH MB BOT FIELD PIX TOP FIELD
BEST MATCH MB BOT FIELD PIX BOT FIELD
42
BEST MATCH FIELD PRED (BOT FIELD)

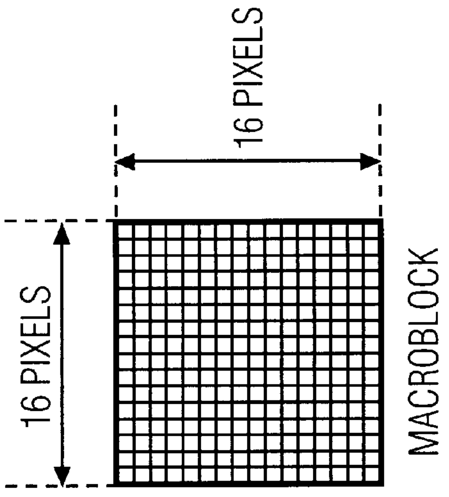


FIG. 1B

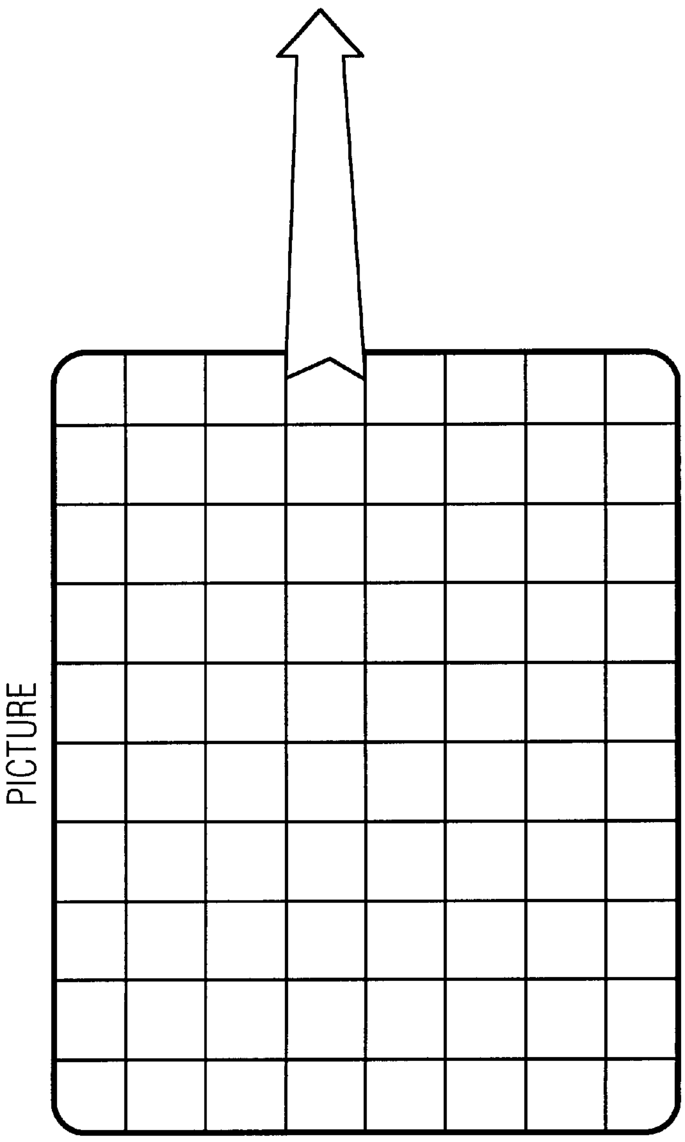


FIG. 1A

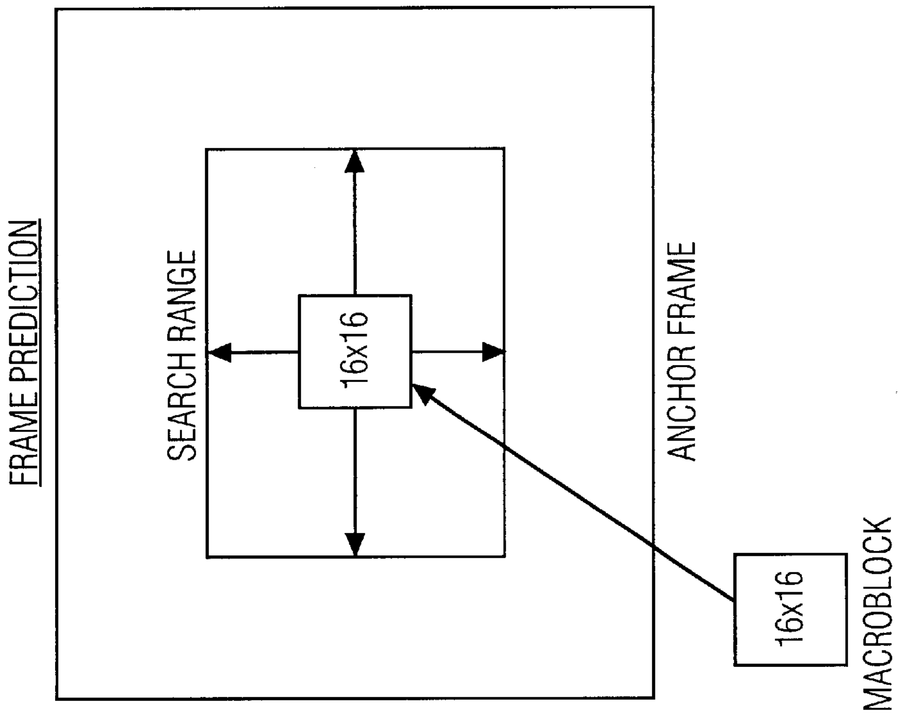

FRAME PREDICTION
SEARCH RANGE
16x16
ANCHOR FRAME
16x16
MACROBLOCK

FIG. 2A

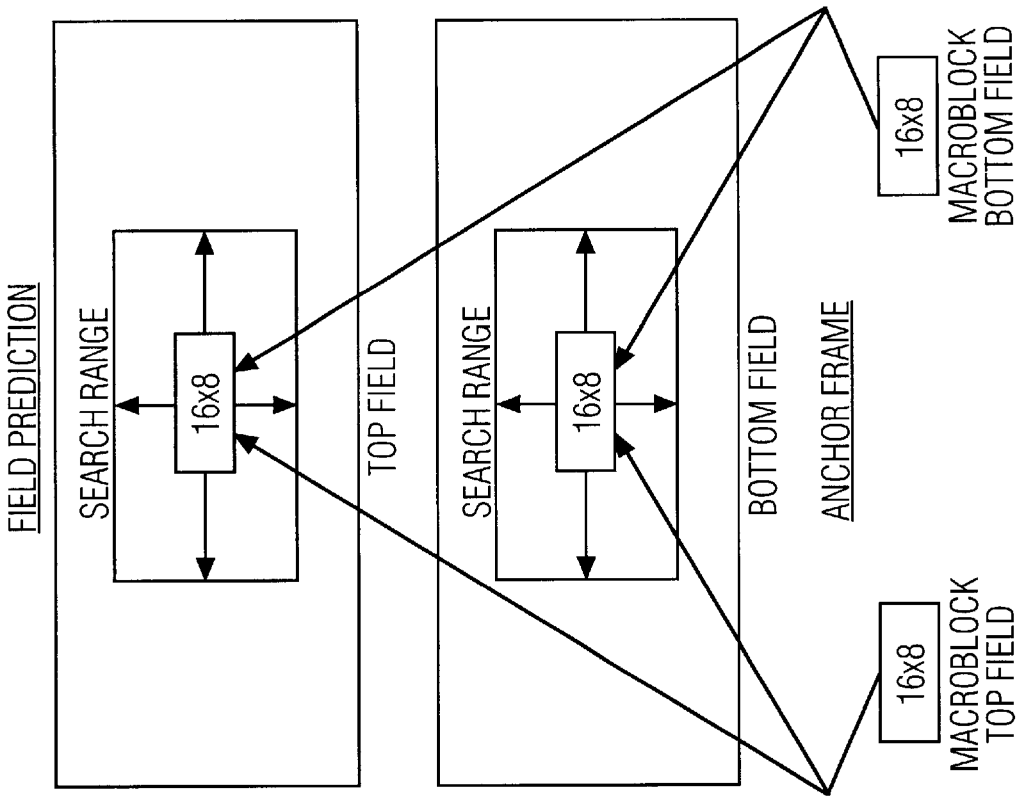

FIELD PREDICTION
SEARCH RANGE
16x8
TOP FIELD
SEARCH RANGE
16x8
BOTTOM FIELD
ANCHOR FRAME
16x8
MACROBLOCK
TOP FIELD
16x8
MACROBLOCK
BOTTOM FIELD

FIG. 2B

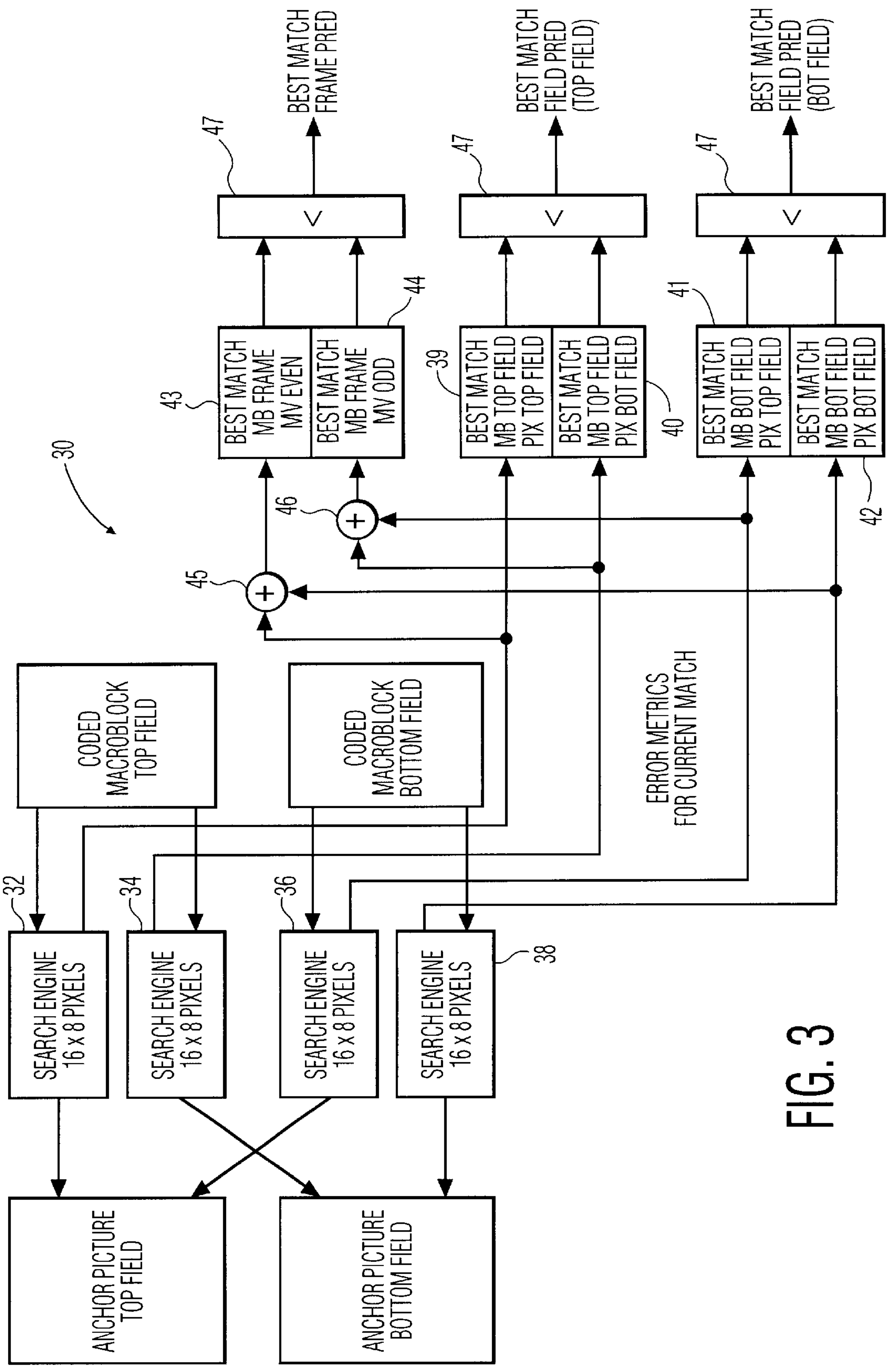

30
BEST MATCH FRAME PRED
BEST MATCH FIELD PRED (TOP FIELD)
BEST MATCH FIELD PRED (BOT FIELD)
47
<
43
BEST MATCH MB FRAME MV EVEN
44
BEST MATCH MB FRAME MV ODD
39
BEST MATCH MB TOP FIELD PIX TOP FIELD
40
BEST MATCH MB TOP FIELD PIX BOT FIELD
41
BEST MATCH MB BOT FIELD PIX TOP FIELD
42
BEST MATCH MB BOT FIELD PIX BOT FIELD
45
46
+
CODED MACROBLOCK TOP FIELD
CODED MACROBLOCK BOTTOM FIELD
ERROR METRICS FOR CURRENT MATCH
32
34
36
38
SEARCH ENGINE 16 x 8 PIXELS
ANCHOR PICTURE TOP FIELD
ANCHOR PICTURE BOTTOM FIELD

FIG. 3

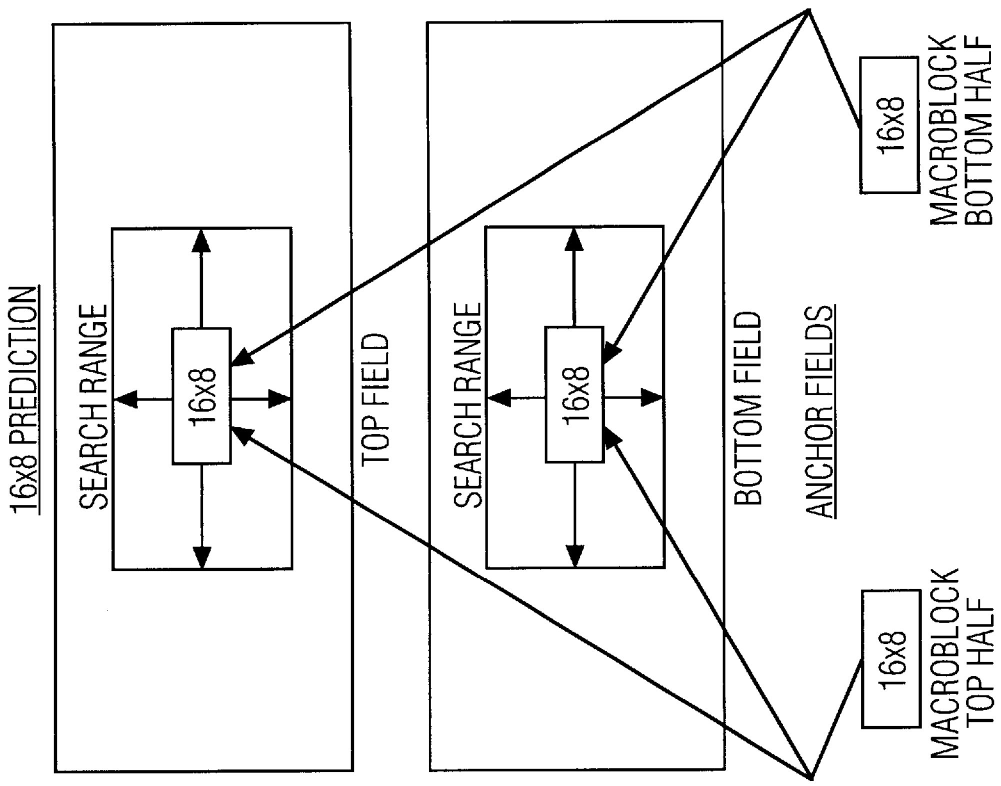

16x8 PREDICTION
SEARCH RANGE
16x8
TOP FIELD
SEARCH RANGE
16x8
BOTTOM FIELD
ANCHOR FIELDS
16x8
MACROBLOCK
BOTTOM HALF
16x8
MACROBLOCK
TOP HALF

FIG. 4B

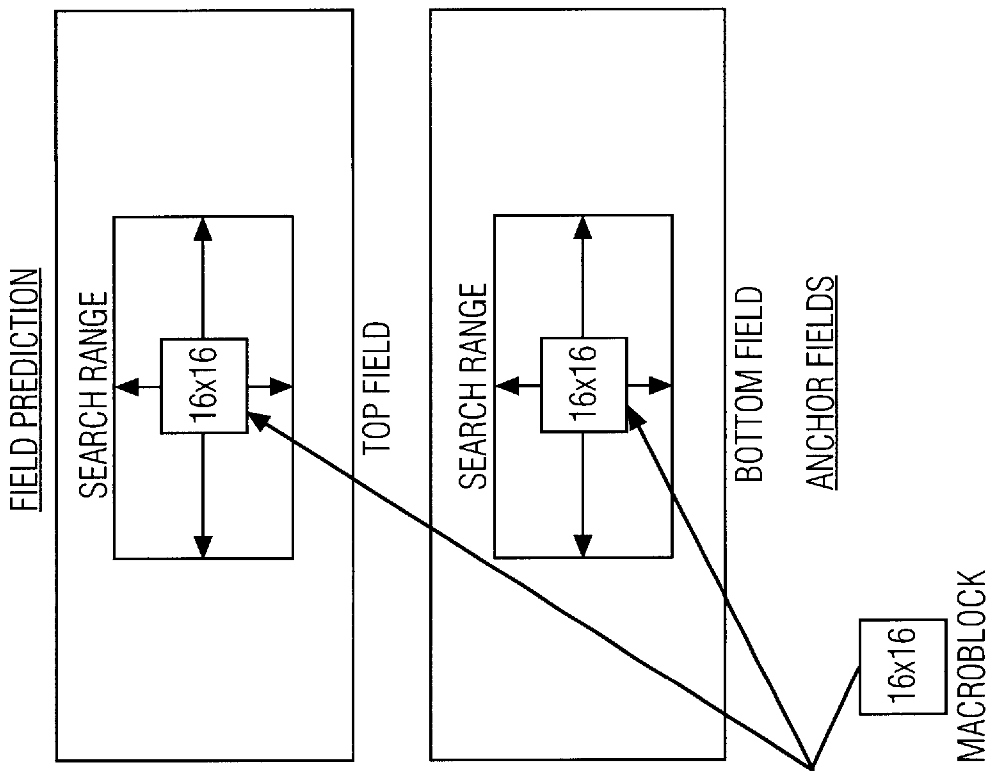
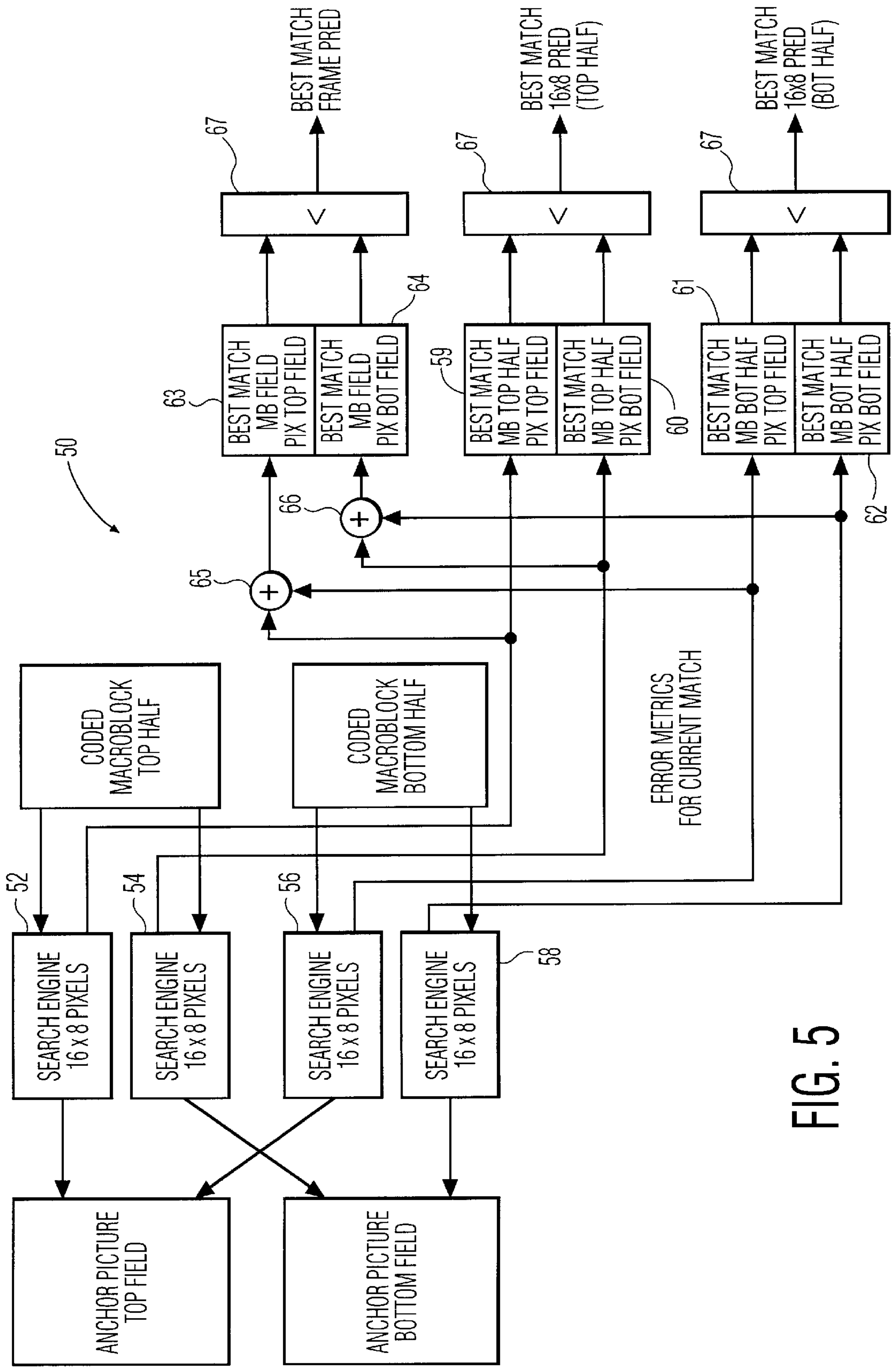

FIELD PREDICTION
SEARCH RANGE
16x16
TOP FIELD
SEARCH RANGE
16x16
BOTTOM FIELD
ANCHOR FIELDS
16x16
MACROBLOCK

FIG. 4A



50
ANCHOR PICTURE TOP FIELD
ANCHOR PICTURE BOTTOM FIELD
SEARCH ENGINE 16 x 8 PIXELS
52
SEARCH ENGINE 16 x 8 PIXELS
54
SEARCH ENGINE 16 x 8 PIXELS
56
SEARCH ENGINE 16 x 8 PIXELS
58
CODED MACROBLOCK TOP HALF
CODED MACROBLOCK BOTTOM HALF
ERROR METRICS FOR CURRENT MATCH
65
66
63
BEST MATCH MB FIELD PIX TOP FIELD
BEST MATCH MB FIELD PIX BOT FIELD
64
59
BEST MATCH MB TOP HALF PIX TOP FIELD
BEST MATCH MB TOP HALF PIX BOT FIELD
60
61
BEST MATCH MB BOT HALF PIX TOP FIELD
BEST MATCH MB BOT HALF PIX BOT FIELD
62
67
<
67
<
67
<
BEST MATCH FRAME PRED
BEST MATCH 16x8 PRED (TOP HALF)
BEST MATCH 16x8 PRED (BOT HALF)

FIG. 5

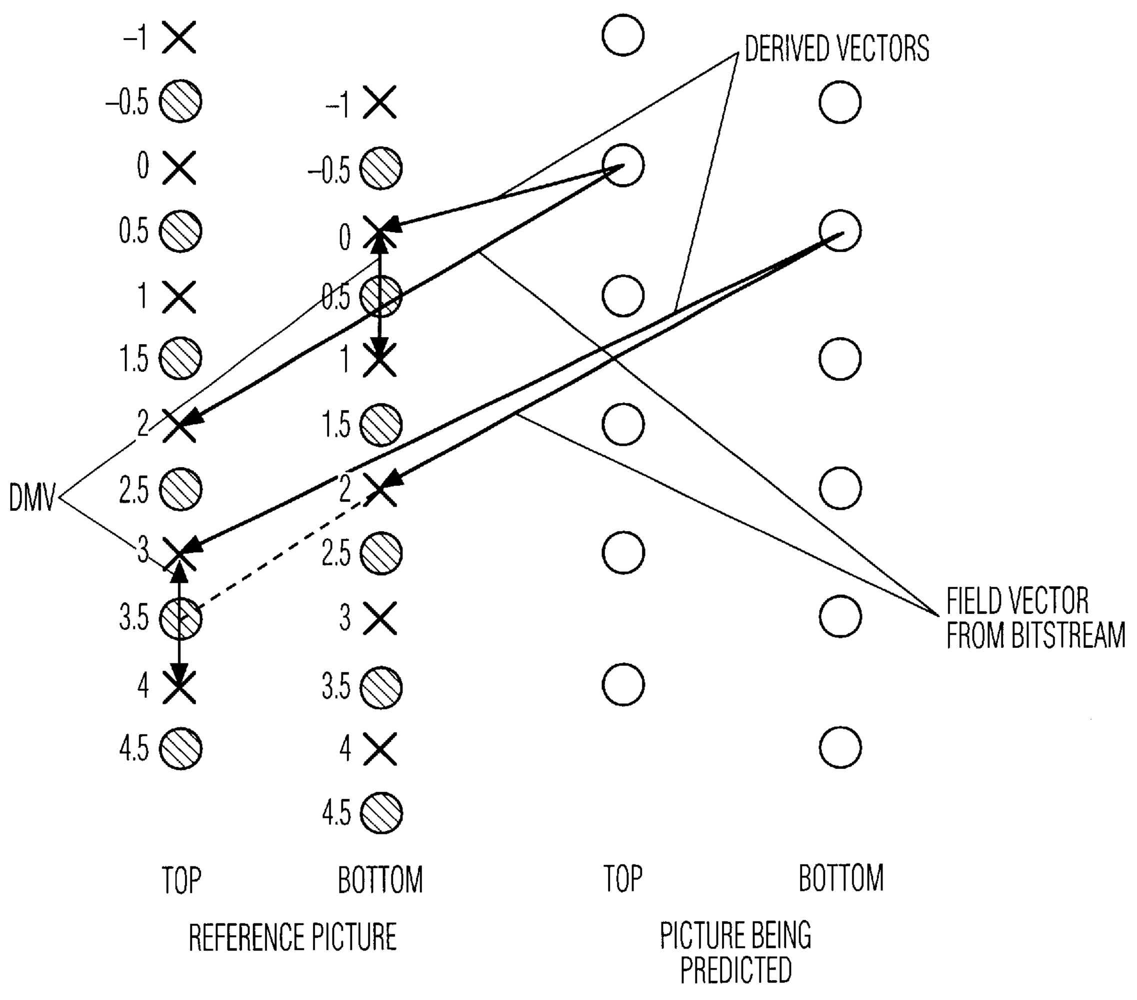

DERIVED VECTORS
DMV
FIELD VECTOR FROM BITSTREAM
-1
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
TOP
BOTTOM
TOP
BOTTOM
REFERENCE PICTURE
PICTURE BEING PREDICTED

FIG. 6

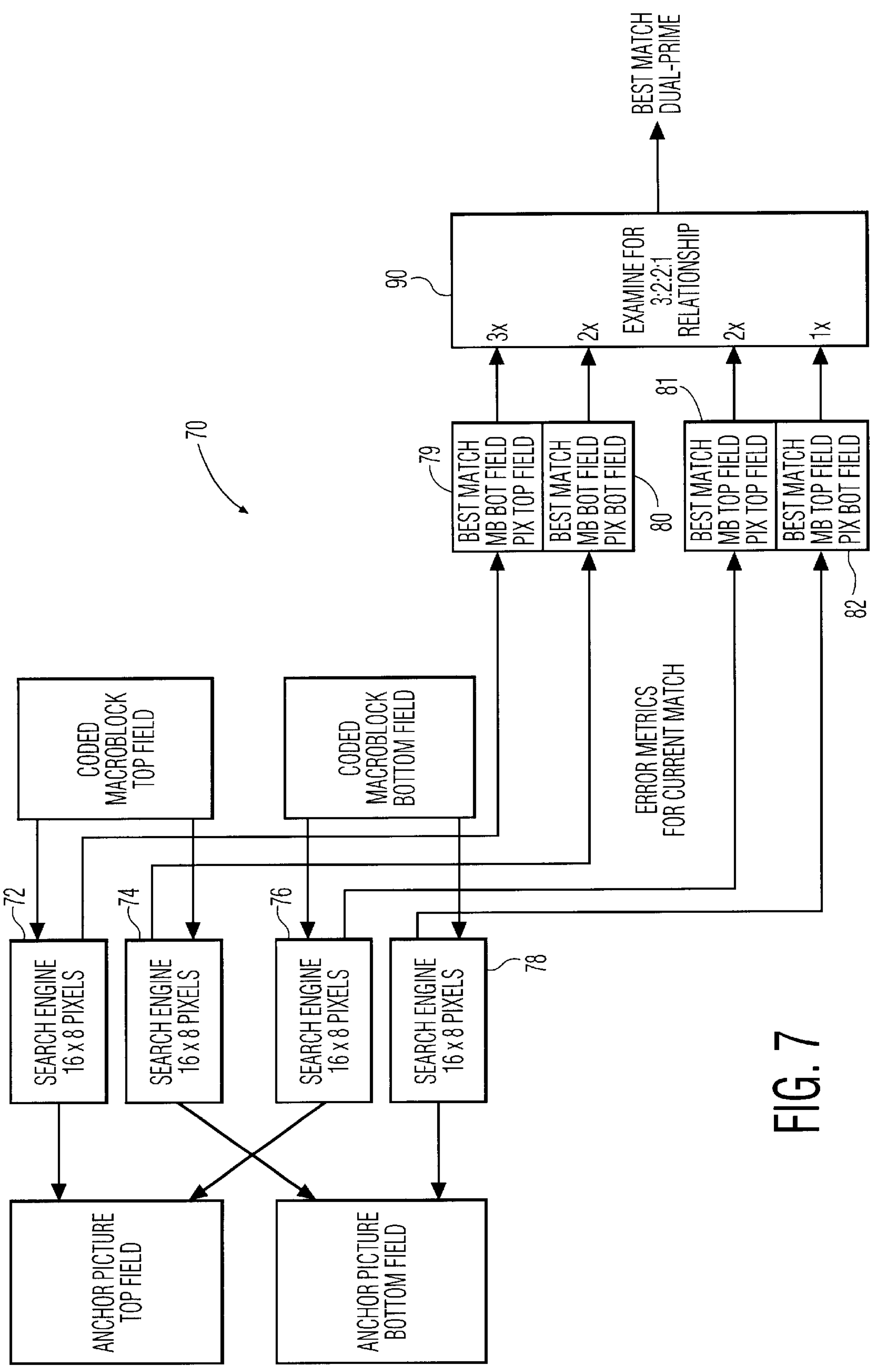
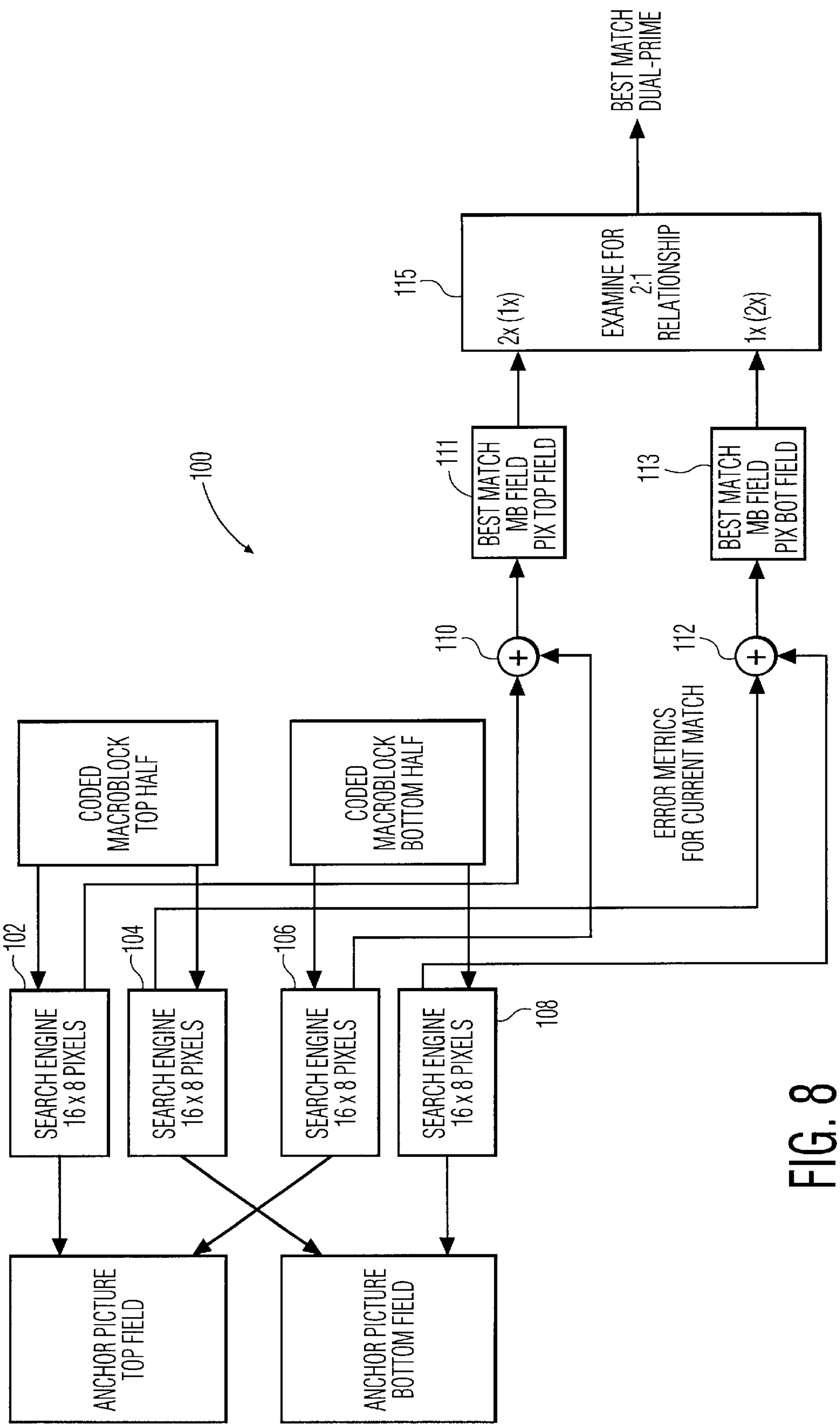

BEST MATCH DUAL-PRIME
90
EXAMINE FOR 3:2:2:1 RELATIONSHIP
3x
2x
2x
1x
70
79
BEST MATCH MB BOT FIELD PIX TOP FIELD
BEST MATCH MB BOT FIELD PIX BOT FIELD
80
81
BEST MATCH MB TOP FIELD PIX TOP FIELD
BEST MATCH MB TOP FIELD PIX BOT FIELD
82
CODED MACROBLOCK TOP FIELD
CODED MACROBLOCK BOTTOM FIELD
ERROR METRICS FOR CURRENT MATCH
72
74
76
78
SEARCH ENGINE 16 x 8 PIXELS
SEARCH ENGINE 16 x 8 PIXELS
SEARCH ENGINE 16 x 8 PIXELS
SEARCH ENGINE 16 x 8 PIXELS
ANCHOR PICTURE TOP FIELD
ANCHOR PICTURE BOTTOM FIELD

FIG. 7



100
CODED MACROBLOCK TOP HALF
CODED MACROBLOCK BOTTOM HALF
102
104
106
108
SEARCH ENGINE 16 x 8 PIXELS
SEARCH ENGINE 16 x 8 PIXELS
SEARCH ENGINE 16 x 8 PIXELS
SEARCH ENGINE 16 x 8 PIXELS
ANCHOR PICTURE TOP FIELD
ANCHOR PICTURE BOTTOM FIELD
ERROR METRICS FOR CURRENT MATCH
110
112
+
+
111
BEST MATCH MB FIELD PIX TOP FIELD
113
BEST MATCH MB FIELD PIX BOT FIELD
115
2x (1x)
EXAMINE FOR 2:1 RELATIONSHIP
1x (2x)
BEST MATCH DUAL-PRIME

FIG. 8

METHOD OF CONCURRENT MULTIPLE-MODE MOTION ESTIMATION FOR DIGITAL VIDEO

BACKGROUND OF THE INVENTION

The present invention relates generally to digital video compression, and, more particularly, to a motion estimation method and search engine for a digital video encoder that is simpler, faster, and less expensive than the presently available technology permits, and that permits concurrent motion estimation using multiple prediction modes.

Many different compression algorithms have been developed in the past for digitally encoding video and audio information (hereinafter referred to generically as "digital video data stream") in order to minimize the bandwidth required to transmit this digital video data stream for a given picture quality. Several multimedia specification committees have established and proposed standards for encoding/compressing and decoding/decompressing audio and video information. The most widely accepted international standards have been proposed by the Moving Pictures Expert Group (MPEG), and are generally referred to as the MPEG-1 and MPEG-2 standards. Officially, the MPEG-1 standard is specified in the ISO/IEC 11172-2 standard specification document, which is herein incorporated by reference, and the MPEG-2 standard is specified in the ISO/IEC 13818-2 standard specification document, which is also herein incorporated by reference. These MPEG standards for moving picture compression are used in a variety of current video playback products, including digital versatile (or video) disk. (DVD) players, multimedia PCs having DVD playback capability, and satellite broadcast digital video. More recently, the Advanced Television Standards Committee (ATSC) announced that the MPEG-2 standard will be used as the standard for Digital HDTV transmission over terrestrial and cable television networks. The ATSC published the *Guide to the Use of the ATSC Digital Television Standard* on Oct. 4, 1995, and this publication is also herein incorporated by reference.

In general, in accordance with the MPEG standards, the audio and video data comprising a multimedia data stream (or "bit stream") are encoded/compressed in an intelligent manner using a compression technique generally known as "motion coding". More particularly, rather than transmitting each video frame in its entirety, MPEG uses motion estimation for only those parts of sequential pictures that vary due to motion, where possible. In general, the picture elements or "pixels" of a picture are specified relative to those of a previously transmitted reference or "anchor" picture using differential or "residual" video, as well as so-called "motion vectors" that specify the location of a 16-by-16 array of pixels or "macroblock" within the current picture relative to its original location within the anchor picture. Three main types of video frames or pictures are specified by MPEG, namely, I-type, P-type, and B-type pictures.

An I-type picture is coded using only the information contained in that picture, and hence, is referred to as an "intra-coded" or simply, "intra" picture.

A P-type picture is coded/compressed using motion compensated prediction (or "motion estimation") based upon information from a past reference (or "anchor") picture (either I-type or P-type), and hence, is referred to as a "predictive" or "predicted" picture.

A B-type picture is coded/compressed using motion compensated prediction (or "motion estimation") based upon

information from either a past and or a future reference picture (either I-type or P-type), or both, and hence, is referred to as a "bidirectional" picture. B-type pictures are usually inserted between I-type or P-type pictures, or combinations of either.

The term "intra picture" is used herein to refer to I-type pictures, and the term "non-intra picture" is used herein to refer to both P-type and B-type pictures. It should be mentioned that although the frame rate of the video data represented by an MPEG bit stream is constant, the amount of data required to represent each frame can be different, e.g., so that one frame of video data (e.g., 1/30 of a second of playback time) can be represented by x bytes of encoded data, while another frame of video data can be represented by only a fraction (e.g., 5%) of x bytes of encoded data. Since the frame update rate is constant during playback, the data rate is variable.

In general, the encoding of an MPEG video data stream requires a number of steps. The first of these steps consists of partitioning each picture into macroblocks. Next, in theory, each macroblock of each "non-intra" picture in the MPEG video data stream is compared with all possible 16-by-16 pixel arrays located within specified vertical and horizontal search ranges of the current macroblock's corresponding location in the anchor picture(s). The MPEG picture and macroblock structure is diagrammatically illustrated in FIG. **1**.

The aforementioned search or "motion estimation" procedure, for a given prediction mode, results in a motion vector(s) that corresponds to the position of the closest-matching macroblock (according to a specified matching criterion) in the anchor picture(s) within the specified search range. Once the prediction mode and motion vector(s) have been determined, the pixel values of the closest-matching macroblock are subtracted from the corresponding pixels of the current macroblock, and the resulting 16-by-16 array of differential pixels is then transformed into 8-by-8 "blocks," on each of which is performed a discrete cosine transform (DCT), the resulting coefficients of which are each quantized and Huffman-encoded (as are the prediction type, motion vectors, and other information pertaining to the macroblock) to generate the MPEG bit stream. If no adequate macroblock match is detected in the anchor picture, or if the current picture is an intra, or "I-" picture, the above procedures are performed on the actual pixels of the current macroblock (i.e., no difference is taken with respect to pixels in any other picture), and the macroblock is designated an "intra" macroblock.

For all MPEG-2 prediction modes, the fundamental technique of motion estimation consists of comparing the current macroblock with a given 16-by-16 pixel array in the anchor picture, estimating the quality of the match according to the specified metric, and repeating this procedure for every such 16-by-16 pixel array located within the search range. The hardware or software apparatus that performs this search is usually termed the "search engine," and there exists a number of well-known criteria for determining the quality of the match. Among the best-known criteria are the Minimum Absolute Error (MAE), in which the metric consists of the sum of the absolute values of the differences of each of the 256 pixels in the macroblock with the corresponding pixel in the matching anchor picture macroblock; and the Minimum Square Error (MSE), in which the metric consists of the sum of the squares of the above pixel differences. In either case, the match having the smallest value of the corresponding sum is selected as the best match within the specified search range, and its horizontal and vertical positions relative to the current macroblock therefore constitute the motion vector. If the resulting minimum sum is nevertheless deemed to large, a suitable match does not exist for the current macroblock, and it is coded as an intra macroblock. For the purposes of the present invention, either of the above two criteria, or any other suitable criterion, may be used.

In accordance with the MPEG-2 standard, any of a number of so-called "prediction modes" may be used for each individual macroblock that is encoded; the optimum prediction mode depends both on the type of picture being encoded and on the characteristics of the portion of the picture in which the given macroblock being encoded is located. Currently known methods of motion coding allow the use of different prediction modes, but generally require one prediction mode to be specified for a given macroblock before an actual motion estimation is performed. Although such a determination can often be made based upon prior knowledge of the picture or image source characteristics, there are many cases where the optimum prediction mode cannot be known unless more than one motion estimation is performed for the macroblock in question. Since motion estimation usually consists of an exhaustive search procedure in which all 256 pixels of two corresponding macroblocks are compared, and which is repeated for a large number of macroblocks, the latter is not a practical option.

Computation of the motion vector(s) for a given macroblock is typically performed by means of an exhaustive search procedure. The current macroblock in question is "compared" with a macroblock-sized pixel array within the anchor picture that is offset by an amount less than specified vertical and horizontal distances, called the "search ranges," and an "error" value is computed for this particular "match" of the macroblock using a specified criterion, or "metric," that gives a measure of how large the error is. This is done for every possible combination of vertical and horizontal offset values within the respective search ranges, and the offset pair that yields the smallest error according to the chosen metric is selected as the motion vector for the current macroblock relative to the anchor picture. Clearly, this procedure is very computationally intensive.

Based on the above and foregoing, it can be appreciated that there presently exists a need in the art that overcomes the disadvantages and shortcomings of the presently available technology. The present invention fulfills this need in the art by performing motion coding of an uncompressed digital video sequence in such a manner that the prediction mode for each individual macroblock is determined as part of the motion estimation process, along with the actual motion vector(s), and need not be specified in advance; only the type of picture currently being coded need be known. Since the latter must be determined at a higher level of video coding than the macroblock layer, this method makes possible a much more efficient, as well as optimal, degree of video compression than would otherwise be possible using conventional methods of motion estimation. Further, the present invention provides a novel scheme for concurrently searching for the optimum macroblock match within the appropriate anchor picture according to each of a plurality of motion prediction modes during the same search operation for the given macroblock, without the need for a separate search to be performed on the same macroblock for each such mode. Since this search procedure is the single most complex and expensive aspect of motion estimation, in both time and hardware, such a method as the present invention will clearly result in a more efficient video image coding and compression than would otherwise be possible given the aforementioned practical limitations of the presently available technology.

Although the present invention was primarily motivated by the specific requirements of the ATSC standard, it can nevertheless be used with any digital video transmission or storage system that employs a video compression scheme, such as MPEG, in which motion coding with multiple prediction modes is used.

SUMMARY OF THE INVENTION

The present invention encompasses a method for motion coding an uncompressed digital video data stream such as an MPEG-2 digital video data stream. The method includes the steps of comparing pixels of a first pixel array in a picture currently being coded with pixels of a plurality of second pixel arrays in at least one reference picture and concurrently performing motion estimation for each of a plurality of different prediction modes in order to determine which of the prediction modes is an optimum prediction mode, determining which of the second pixel arrays constitutes a best match with respect to the first pixel array for the optimum prediction mode, and, generating a motion vector for the first pixel array in response to the determining step. The method is implemented in a device such as a motion estimation search system of a digital video encoder. In one embodiment, the method and device are capable of concurrently determining performing motion estimation in each of the six different possible prediction modes specified by the MPEG-2 standard.

The present invention also encompasses a method for motion coding a digital video data stream comprised of a sequence of pictures having top and bottom fields which includes the steps of comparing pixels of a first portion (e.g., 16-by-8 portion) of a current macroblock (e.g., a 16-by-16 macroblock) of the top field of a current picture with pixels of each of a plurality of correspondingly-sized portions of a macroblock of a top field of an anchor picture in accordance with a prescribed search metric, and producing a first error metric for each comparison; comparing pixels of the first portion (e.g., 16-by-8 portion) of the current macroblock of the top field of the current picture with pixels of each of the plurality of correspondingly-sized portions of a macroblock of a bottom field of the anchor picture in accordance with the prescribed search metric, and producing a second error metric for each comparison; comparing pixels of a second portion (e.g., 16-by-8 portion) of a current macroblock (e.g., a 16-by-16 macroblock) of the bottom field of the current picture with pixels of each of the plurality of correspondingly-sized portions of the macroblock of the top field of the anchor picture in accordance with the prescribed search metric, and producing a third error metric for each comparison; comparing pixels of the:second portion (e.g., a 16-by-8 portion) of the current macroblock of the bottom. field of the current picture with pixels of each of the plurality of correspondingly-sized portions of the macroblock of the bottom field of the anchor picture in accordance with the prescribed search metric, and producing a fourth error metric for each comparison; summing the first and fourth error metrics to produce a first composite error metric; summing the second and third error metrics to produce a second composite error metric; and, determining which of the first, second, third, and fourth error metrics, and first and second composite error metrics has the lowest value, and selecting one a plurality of possible motion estimation prediction modes on the basis of such determination. Preferably and advantageously, all of the comparing steps are performed concurrently, and both of the summing steps are performed concurrently. The plurality of possible motion estimation prediction modes can include frame and field prediction modes for frame pictures in accordance with the MPEG-2 standard.

The present invention also encompasses a method for motion coding a digital video data stream comprised of a sequence of pictures, in which the method includes the steps of comparing pixels of a first portion (e.g., 16-by-8 portion) of a top half of a current macroblock (e.g., a 16-by-16 macroblock) of a current picture with pixels of each of a plurality of correspondingly-sized portions of a macroblock of a top field of an anchor picture in accordance with a prescribed search metric, and producing a first error metric for each comparison; comparing pixels of the first portion (e.g., 16-by-8 portion) of the top half of the current macroblock of the current picture with pixels of each of the plurality of correspondingly-sized portions of a macroblock of a bottom field of the anchor picture in accordance with the prescribed search metric, and producing a second error metric for each comparison; comparing pixels of a second portion (e.g., 16-by-8 portion) of a bottom half of a current macroblock (e.g., a 16-by-16 macroblock) of the current picture with pixels of each of the plurality of correspondingly-sized portions of the macroblock of the top field of the anchor picture in accordance with the prescribed search metric, and producing a third error metric for each comparison; comparing pixels. of the second portion (e.g., a 16-by-8 portion) of the bottom half of the current macroblock of the current picture with pixels of each of the plurality of correspondingly-sized portions of the inacroblock of the bottom field of the anchor picture in accordance with the prescribed search metric, and producing a fourth error metric for each comparison; summing the first and third error metrics to produce a first composite error metric; summing the second and fourth error metrics to produce a second composite error metric; and, determining which of the first, second, third, and fourth error metrics, and first and second composite error metrics has the lowest value, and selecting one a plurality of possible motion estimation prediction modes on the basis of such determination. Preferably and advantageously, all of the comparing steps are performed concurrently, and both of the summing steps are performed concurrently. The plurality of possible motion estimation prediction modes can include field and 16×8 prediction modes for field pictures in accordance with the MPEG-2 standard.

The present invention also encompasses a method for motion coding a digital video data stream comprised of a sequence of pictures having top and bottom fields which includes the steps of comparing pixels of a first portion (e.g., 16-by-8 portion) of a current macroblock (e.g., a 16-by-16 macroblock) of the top field of a current picture with pixels of each of a plurality of correspondingly-sized portions of a macroblock of a top field of an anchor picture in accordance with a prescribed search metric, and producing a first error metric for each comparison; comparing pixels of the first portion (e.g., 16-by-8 portion) of the current macroblock of the top field of the current picture with pixels of each of the plurality of correspondingly-sized portions of a macroblock of a bottom field of the anchor picture in accordance with the prescribed search metric, and producing a second error metric for each comparison; comparing pixels of a second portion (e.g., 16-by-8 portion) of a current macroblock (e.g., a 16-by-16 macroblock) of the bottom field of the current picture with pixels of each of the plurality of correspondingly-sized portions of the macroblock of the top field of the anchor picture in accordance with the prescribed search metric, and producing a third error metric for each comparison; comparing pixels of the second portion (e.g., a 16-by-8 portion) of the current macroblock of the bottom field of the current picture with pixels of each of the plurality of correspondingly-sized portions of the macroblock of the bottom field of the anchor picture in accordance with the prescribed search metric, and producing a fourth error metric for each comparison; producing first, second, third, and fourth motion vectors on the basis of the first, second, third, and fourth error metrics, respectively; and, examining the first, second, third, and fourth motion vectors to determine whether a prescribed relationship between them is present, and, if so, selecting a frame picture dual-prime motion estimation prediction mode. Preferably and advantageously, all of the comparing steps are performed concurrently.

The present invention also encompasses a method for motion coding a digital video data stream comprised of a sequence of pictures, in which the method includes the steps of comparing pixels of a first portion (e.g., 16-by-8 portion) of a top half of a current macroblock (e.g., a 16-by-16 macroblock) of a current picture with pixels of each of a plurality of correspondingly-sized portions of a macroblock of a top field of an anchor picture in accordance with a prescribed search metric, and producing a first error metric for each comparison; comparing pixels of the first portion (e.g., 16-by-8 portion) of the top half of the current macroblock of the current picture with pixels of each of the plurality of correspondingly-sized portions of a macroblock of a bottom field of the anchor picture in accordance with the prescribed search metric, and producing a second error metric for each comparison; comparing pixels of a second portion (e.g., 16-by-8 portion) of a bottom half of a current macroblock (e.g., a 16-by-16 macroblock) of the current picture with pixels of each of the plurality of correspondingly-sized portions of the macroblock of the top field of the anchor picture in accordance with the prescribed search metric, and producing a third error metric for each comparison; comparing pixels of the second portion (e.g., a 16-by-8 portion) of the bottom half of the current macroblock of the current picture with pixels of each of the plurality of correspondingly-sized portions of the macroblock of the bottom field of the anchor picture in accordance with the prescribed search metric, and producing a fourth error metric for each comparison; summing the first and third error metrics to produce a first composite error metric; summing the second and fourth error metrics to produce a second composite error metric; producing first and second motion vectors on the basis of the first and second composite error metrics, respectively; and, examining the first and second motion vectors to determine whether a prescribed relationship between them is present, and if so, selecting a field picture dual-prime motion estimation prediction mode. Preferably and advantageously, all of the comparing steps are performed concurrently, both of the summing steps are performed concurrently, and both of the producing steps are performed concurrently.

The present invention further encompasses a device such as a motion estimation search system for a digital video encoder that concurrently implements any of the above-described methods of the present invention in any combination thereof.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other objects, features, and advantages of the present invention will be readily understood from the following detailed description taken in conjunction with the accompanying drawings, in which:

FIG. **1** is a diagram that illustrates the MPEG picture and macroblock structure;

FIG. **2** is a diagram that illustrates motion estimation for frame pictures using frame and field prediction;

FIG. **3** is a block diagram of a motion estimation search system constructed in accordance with an exemplary embodiment of the present invention for concurrently performing motion estimation for frame prediction mode and field prediction modes for frame pictures;

FIG. **4** is a diagram that illustrates motion estimation for field (16×16) and 16×8 prediction modes for field pictures;

FIG. **5** is a block diagram of a motion estimation search system constructed in accordance with an exemplary embodiment of the present invention for performing motion estimation for field prediction and 16×8 prediction modes for field pictures;

FIG. **6** is a diagram that illustrates motion estimation using dual-prime prediction;

FIG. **7** is a block diagram of a motion estimation search system constructed in accordance with an exemplary embodiment of the present invention for performing frame picture dual-prime motion estimation; and,

FIG. **8** is a block diagram of a motion estimation search system constructed in accordance with an exemplary embodiment of the present invention for performing field picture dual-prime motion estimation.

DETAILED DESCRIPTION OF THE INVENTION

FIG. **2** diagrammatically depicts the MPEG-2 motion estimation process for frame pictures using the frame and field prediction modes, respectively. In frame prediction, the composite anchor frame is treated as a contiguous picture, and the composite macroblock is treated as a contiguous 16-by-16 pixel array. The motion estimation procedure is performed in the manner described hereinabove.

In field prediction, however, the current macroblock is partitioned into one 16-by-8 array consisting of lines from the top field (even-numbered lines, starting with 0), and a second, 16-by-8 array consisting of lines from the bottom field (odd-numbered lines, starting with 1). The anchor frame picture is also partitioned into a top-field picture (even-numbered lines) and a bottom-field picture (odd-numbered lines). The top-field 16-by-8 array is then matched, in a manner analogous to that described hereinabove, with every 16-by-8 pixel array within the search range in the top-field anchor. picture, in order to find the best match.

The procedure is then repeated, using the same top-field 16-by-8 array, in the bottom-field picture to find the best match. The two resulting matches are then compared, and the better of the two is selected as the best match for the top field of the macroblock. The match is represented by specifying the anchor field picture (top or bottom) in which it was found, along with the corresponding motion vector.

The entire procedure is repeated for the bottom-field 16-by-8 array, once again using both the top and bottom fields of the anchor frame in the manner described above to determine which of the two fields contains the better optimum match and to give its corresponding motion vector. The final result is an anchor field selector, motion vector pair for each of the top and bottom field 16-by-8 arrays of the current macroblock.

If the current picture is a predicted, or "P," picture, forward coding is used exclusively. In the case of a bidirectional, or "B," picture, however, the prediction may be forward, backward, or bidirectional. In the first two cases, the above motion estimation is performed using the forward or backward anchor picture, respectively, as required. In the case of bidirectional. coding, however, the same motion estimation must be performed for both the forward and the backward anchor picture. In a B picture, the prediction direction(s) is (are) specified individually for each macroblock of the current picture.

In all known motion estimation methods, the prediction mode must be specified for every macroblock before the motion estimation, with its constituent search, is performed. However, in accordance with the present invention, in one of its aspects, the motion estimation may be performed, in a frame picture, for both frame and field prediction modes simultaneously, during the same search of the anchor picture.

The observation that, for the same horizontal and vertical offset, the sum of the motion estimation match criterion, or metric, for the top-field 16-by-8 array in the top field of the anchor frame and that of the bottom-field 16-by-8 array in the bottom field of the anchor frame (in both cases using field prediction) is equal to the corresponding metric for the composite 16-by-16 macroblock array in the composite anchor frame (using frame prediction) illustrates how it is possible to perform motion estimation for more than one prediction mode during a single search. In order to accomplish this, the optimal match must be determined for each of the top- and bottom-field 16-by-8 arrays in each of the top- and bottom-field anchor pictures. If all searches are performed such that, at any given time, the horizontal and vertical offsets of the current attempted match are the same (a reasonable assumption in light of the fact that, in a practical motion estimation system, anchor picture pixels correspond to memory locations, which in conventional memory technologies are typically assessed only one at a time), a metric value is generated for each of the four attempted matches. If the current metric for the top-field 16-by-8 array in the top-field anchor picture is added to that for the bottom-field 16-by-8 array in the bottom-field anchor picture, the result, in the case of an even-numbered vertical offset is equal to the current metric for the composite 16-by-16 macroblock in the composite anchor frame. Just as the optimum metric values are determined for each of the four field prediction searches over the specified search range, the optimum metric value for frame prediction can also be determined from the above sum. In the case of an odd-numbered vertical offset, the top-field 16-by-16 pixel array is matched in the bottom field anchor picture, and the bottom-field 16-by-16 pixel array is matched in the top field anchor picture; the vertical pixel locations within the respective anchor field pictures will also differ by 1 in this case.

A motion estimation search system **30** that implements the above-described motion estimation method of the present invention is depicted in FIG. **3**, and will now be described. More particularly, the motion estimation search system **30** includes four parallel search engines **32**, **34**, **36**, and **38** that compare respective portions of the coded macroblock top and bottom fields with appropriate portions of the anchor picture top and bottom fields in the manner described hereinabove, in accordance with a prescribed search metric, e.g., Minimum Absolute Error (MAE). The search engines **32**, **34**, **36**, and **38** produce respective error metrics for each comparison operation they perform. In particular, the error metrics produced by the search engine **32** are applied to an input of a logic element **39** that determines which of the anchor picture top field macroblocks constitutes the best match with respect to the coded macroblock top field, and then and then produces the best match results at its output. The error metrics produced by the search engine **34** are applied to an input of a logic element **40** that determines which of the anchor picture bottom field macroblocks constitutes the best match with respect to the coded macroblock top field, and then produces the best match results at its output. The error metrics produced by the search engine **36** are applied to an input of a logic element **41** that determines which of the anchor picture top field macroblocks constitutes the best match with respect to the coded macroblock bottom field, and then produces the best match results at its output. The error metrics produced by the search engine **38** are applied to an input of a logic element **42** that determines which of the anchor picture bottom field macroblocks constitutes the best match with respect to the coded macroblock bottom field, and then produces the best match results at its output. The error metrics produced by the search engines **32** and **38** are combined by an adder circuit **45**, and the resultant composite error metric is applied to an input of a logic element **43** that determines which of the anchor picture macroblocks constitutes the best match with respect to the coded macroblock for the case of an even-numbered vertical offset, and then produces the best match results at its output. The error metrics produced by the search engines **34** and **36** are combined by an adder circuit **46**, and the resultant composite error metric is applied to an input of a logic element **44** that determines which of the anchor picture macroblocks constitutes the best match with respect to the coded macroblock for the case of an odd-numbered vertical offset, and then produces the best match results at its output. Parallel comparison logic elements **47** compare the best match results generated by the logic elements **39**–**44**, and then determine which of the prediction modes (i.e., the field or frame prediction mode for frame pictures) is optimum for the coded macroblock on the basis thereof. The corresponding motion vector for the best match produced by the selected prediction mode is then output for further processing by the motion estimation search system.

The ATSC standard, which corresponds to the MPEG-2 main profile at high-level, allows as many as six different prediction modes. Of these, two were considered in the above description, namely the frame and field prediction modes in frame pictures, respectively. For field pictures, there are two analogous modes, namely field prediction, in which a 16-by-16 pixel macroblock in the current field picture is matched in one of the two previous anchor field pictures in a manner similar to that used for frame prediction in frame pictures; and 16-by-8 prediction, in which the upper 16-by-8 pixel half of the current macroblock is matched in either of the previous two anchor field pictures (and/or the following two anchor pictures in the case of backward coding in B pictures), and the lower half of the same macroblock is independently matched in either of the two previous anchor field pictures, this time in a manner similar to that used for field prediction in frame pictures. These two prediction modes for field pictures are illustrated diagrammatically in FIG. **4**.

As before, all searches are performed such that, at any given time, the horizontal and vertical offsets of the four current attempted matches are the same, and a metric value is generated for each one. Since the relative offset for the upper half of the current macroblock with respect to the upper half of the attempted matching macroblock in either anchor field is the same as the relative offset for the lower half of the current macroblock with respect to the lower half of the same attempted match in either anchor field, separate metrics can be computed, during the full macroblock search, for the upper and lower halves of the current macroblock. If the metric value for the upper 16-by-8 array in the top-field anchor picture is added to that for the lower 16-by-8 array in the top-field anchor picture, the result is equal to the metric value for the composite 16-by-16 macroblock in the top-field anchor picture. The same holds true for the bottom-field anchor picture. Just as the optimum metric values are determined for each of the four 16-by-8 prediction searches over the specified search range, the optimum metric values for each of the two field prediction searches can also be determined from the above sums.

A motion estimation search system **50** that implements the above-described motion estimation method of the present invention is depcited in FIG. **5**, and will now be described. More particularly, the motion estimation search system **50** includes four parallel search engines **52**, **54**, **56**, and **58** that compare respective portions of the coded macroblock top and bottom halves with appropriate portions of the anchor picture top and bottom fields in the manner described hereinabove, in accordance with a prescribed search metric, e.g., Minimum Absolute Error (MAE). The search engines **52**, **54**, **56**, and **58** produce respective error metrics for each comparison operation they perform. In particular, the error metrics produced by the search engine **52** are applied to an input of a logic element **59** that determines which of the anchor picture top field macroblocks constitutes the best match with respect to the coded macroblock top half, and then and then produces the best match results at its output.

The error metrics produced by the search engine **54** are applied to an input of a logic element **60** that determines which of the anchor picture bottom field macroblocks constitutes the best match with respect to the coded macroblock top half, and then produces the best match results at its output.

The error metrics produced by the search engine **56** are applied to an input of a logic element **61** that determines which of the anchor picture top field macroblocks constitutes the best match with respect to the coded macroblock bottom half, and then produces the best match results at its output. The error metrics produced by the search engine **48** are applied to an input of a logic element **62** that determines which of the anchor picture bottom field macroblocks constitutes the best match with respect to the coded macroblock bottom half, and then produces the best match results at its output.

The error metrics produced by the search engines **52** and **56** are combined by an adder circuit **65**, and the resultant composite error metric is applied to an input of a logic element **63** that determines which of the top field anchor picture macroblocks constitutes the best match with respect to the coded macroblock, and then produces the best match results at its output.

The error metrics produced by the search engines **54** and **58** are combined by an adder circuit **66**, and the resultant composite error metric is applied to an input of a logic element **64** that determines which of the bottom field anchor picture macroblocks constitutes the best match with respect to the coded macroblock, and then produces the best match results at its output.

Parallel comparison logic elements **67** compare the best match results generated by the logic elements **59**–**64**, and then determine which of the prediction modes (i.e., the field or 16-by-8 prediction mode for field pictures) is optimum for the coded macroblock on the basis thereof The corresponding motion vector for the best match produced by the selected prediction mode is then output for further processing by the motion estimation search system.

The final remaining MPEG-2 motion prediction mode is the so-called "dual-prime" mode, which may be used in cases where the source video is interlaced and where only I and P pictures are used in the encoding process (i.e., B pictures are not allowed). In this mode, which may be used in either frame or field pictures that meet the aforementioned criteria, advantage is taken of the physical properties of object motion within an interlaced video sequence to represent a plurality of motion vectors (four in the case of frame pictures, two in the case of field pictures) using just one encoded motion vector. This allows less information to be transmitted or stored per macroblock and, thereby results in more efficient video compression.

In interlaced video, each complete frame is partitioned into two separate fields, the first of which is designated the top field and consists of all even-numbered lines of the composite video frame (starting with 0), and the second of which is designated the bottom field and consists of all odd-numbered lines (starting with 1). In this mode of operation, the top-field image of a given frame is generated in its entirety, and the bottom field image of the same frame is subsequently generated, also in its entirety. The procedure is then repeated for the following frame, and then for all subsequent frames. In a video system with a specified frame rate (for example, 30 frames per second, with the NTSC standard, which is used in the United States), the corresponding field rate will be precisely twice this frame rate (60 fields per second in the case of the NTSC standard). This means that the time difference between two consecutive video fields is exactly half the time difference between two consecutive frames. Since most conventional video sources, such as cameras and recorders, generate lines of video in a sequential, raster-scan format, the time difference between corresponding lines (i.e., lines having the same vertical position) in consecutive fields will always have the same value, even if the times associated with different lines in the same field are different due to the constant vertical rate of the raster scan.

In a video sequence, an object that is moving with a uniform velocity will move by a finite distance within the image, vertically and horizontally, during the time interval between two consecutive frames. In the time between two consecutive fields, however, it will move by precisely half the aforementioned distance, according to the principles explained in the previous paragraph. In the more general case where the motion is not uniform, however, the small value of the time interval between subsequent frames (and the even smaller interval between subsequent fields), generally ensures that the second and higher-order derivatives of the object motion may be ignored, and that, over this small interval, the motion can safely be treated as uniform. This means that the above relationships concerning the distances of object motion between consecutive frames and that between consecutive fields, as well as the constancy of the motion between such fields, will effectively hold true even for non-uniform motion. The dual-prime mode of motion prediction capitalizes extensively on the above relationships.

The MPEG-2 specification for the dual-prime prediction modes in both frame and field pictures is diagrammatically depicted in FIG. 6. As can be seen from the illustration, motion vectors for fields of a given parity (e.g., top field) relative to the previous field of the same parity have a certain length. In a frame picture, where each of the two constituent fields is motion-coded relative to each of the two constituent fields of the previous anchor frame, the top-field to top-field and bottom-field to bottom-field vectors are seen to have the same length. This is expected according to the above analysis, since, in both cases, they represent the distance traveled by the object in the course of two video intervals. The motion vectors for fields of one parity relative to the opposite parity, however, will represent the distance traveled in one field interval (in the case of that for a top field relative to the previous bottom field), or three field intervals (in the case of that for a bottom field relative to the previous top field). In the former case, the motion vector will have a length of one-half the value of the above two motion vectors; in the latter case, it will have three-halves of this value.

In a field picture, which is motion-coded relative to the two previous anchor fields, the motion vector for the field of the same parity, once again, represents the distance traveled by the object in the course of two video field intervals, and has a certain length. The motion vector for the field of opposite parity, however, always refers to the previous field and, therefore, represents the distance traveled in one field interval; it will thus have one-half of the value of the above motion vector.

Upon initial examination, it appears that different fields must be searched for matches located at different horizontal and vertical offsets relative to the current macroblock in order to determine whether the above criteria for dual-prime representation are satisfied. Consideration of the fact that, in an interlaced video source, these criteria arise naturally from the properties of motion in a two-dimensional image, leads to the conclusion that, if all of the appropriate searches are performed, using the field prediction mode, for the current macroblock in the required anchor pictures, the resulting optimal motion vectors should automatically have the relative relationships required for dual-prime representation; that is, motion vectors corresponding to fields of the same parity should have a length of one-half or three-halves that of the above motion vector, depending upon the specific relationship between the fields. It is, therefore, only necessary to perform the conventional motion estimations for field prediction on either a frame picture or a field picture, and then examine the resulting motion vectors to determine whether the relative relationships required for dual-prime representation are present. If they are, the macroblock is simply encoded using the dual-prime prediction mode; if not, the most optimal of the other prediction modes is chosen instead.

In either a frame picture or a field picture, it is possible, due either to nonuniformity of motion or simply to spatial quantization of the image, that the relative relationships required for the motion vectors are very nearly, but not exactly, met. In addition, there always is a one-line vertical offset between the top and bottom fields of a video frame due to the nature of interlacing. The MPEG-2 standard accommodates the first of these situations by allowing a so-called "differential motion vector" for each of the vertical and horizontal components of the encoded vector, which is restricted to the three values −1, 0, and +1. It also accommodates the second situation by always providing a vertical correction for all derived motion vectors, which always predicts a field of a given parity relative to that of the opposite parity. In the event that the required relationships are still not exactly met, it is always possible to choose a slightly different motion vector value for the case that does not conform; although not precisely optimal, the overall superiority of dual-prime coding may nevertheless make this preferable in such a situation.

Dual-prime prediction for a frame picture consists of field prediction for the current macroblock relative to both fields of the previous anchor frame. This means that the top-field portion of the current macroblock is matched with both the top and bottom fields of the anchor frame (in contrast with conventional field prediction of frame pictures, where only the anchor field yielding the better prediction is chosen), and the same is done for the bottom-field portion of the current macroblock. Four motion vectors are therefore needed. The motion estimation system **30** depicted in FIG. **3**, when used for field prediction, was designed to determine the optimum motion vectors for precisely the four matches required for dual-prime prediction in a frame picture. Consequently, the resultant four motion vectors need only be examined to determine whether the required relative relationships given in the above discussion holds among the four vectors. The same architecture used to simultaneously perform frame and field prediction in a frame picture, and select the better mode, can thus implement the dual-prime prediction mode and choose it over the other two prediction modes if superior to them as well. The resulting architecture of a motion estimation system **70** for motion estimation and coding of frame pictures is depicted in FIG. **7**.

With specific reference to FIG. **7**, the motion estimation system **70** includes four parallel search engines **72**, **74**, **76**, and **78** that compare respective portions of the coded macroblock top and bottom fields with appropriate portions of the anchor picture top and bottom fields in the manner described hereinabove, in accordance with a prescribed search metric, e.g., Minimum Absolute Error (MAE). The search engines **72**, **74**, **76**, and **78** produce respective error metrics for each comparison operation they perform. In particular, the error metrics produced by the search engine **72** are applied to an input of a logic element **79** that determines which of the anchor picture top field macroblocks constitutes the best match with respect to the coded macroblock top field, and then produces the corresponding motion vector at its output. The error metrics produced by the search engine **74** are applied to an input of a logic element **80** that determines which of the anchor picture bottom field macroblocks constitutes the best match with respect to the coded macroblock top field, and then produces the corresponding motion vector at its output. The error metrics produced by the search engine **76** are applied to an input of a logic element **81** that determines which of the anchor picture top field macroblocks constitutes the best match with respect to the coded macroblock bottom field, and then produces the corresponding motion vector at its output. The error metrics produced by the search engine **78** are applied to an input of a logic element **82** that determines which of the anchor picture bottom field macroblocks constitutes the best match with respect to the coded macroblock bottom field, and then produces the corresponding motion vector at its output. The motion vectors produced by the logic elements **79**–**82** are examined by a logic circuit **90** for a 3:2:2:1 relationship, and if such a relationship is determined to exist between these motion vectors, then a frame picture dual-prime motion estimation prediction mode is selected and the corresponding motion vector generated.

Dual-prime prediction for a field picture consists of field prediction for the current macroblock relative to the two previous anchor fields. This means that the current macroblock is matched with the previous top and bottom anchor fields (in contrast with conventional field prediction of field pictures, where only the anchor field yielding the better prediction is chosen). Two motion vectors are therefore needed. The motion estimation system **50** depicted in FIG. **5**, when used for field prediction, was designed to determine the optimum motion vectors for precisely the two matches required for dual-prime prediction in a field picture. Consequently, these two motion vectors need only be examined to determine whether the required relative relationships given in the above discussion holds among the two vectors. The same architecture used to simultaneously perform field

and 16-by-8 prediction in a field picture, and select the better mode, can thus implement the dual-prime prediction mode, and choose it over the other two modes if superior to them as well. The resulting architecture of a motion estimation system **100** for motion estimation and coding of field pictures is depicted in FIG. **8**.

With specific reference to FIG. **8**, the motion estimation system **100** includes four parallel search engines **102**, **104**, **106**, and **108** that compare respective portions of the coded macroblock top and bottom halves with appropriate portions of the anchor picture top and bottom fields in the manner described hereinabove, in accordance with a prescribed search metric, e.g., Minimum Absolute Error (MAE). The search engines **102**, **104**, **106**, and **108** produce respective error metrics for each comparison operation they perform. In particular, the error metrics produced by the search engine **102** are applied to a first input of a first adder **110**, and the error metrics produced by the search engine **106** are applied to a second input of the first adder **110**, which produces at its output the sum of the error metrics applied to its first and second inputs as a first composite error metric. The first composite error metric is applied to a logic element **111** that determines which of the anchor picture top field macroblocks constitutes the best match with. respect to the coded macroblock, and then produces the corresponding motion vector at its output. The error metrics produced by the search engine **104** are applied to a first input of a second adder **112**, and the error metrics produced by the search engine **108** are applied to a second input of the second adder **112**, which produces at its output the sum of the error metrics applied to its first and second inputs as a second composite error metric. The second composite error metric is applied to a logic element **113** that determines which of the anchor picture bottom field macroblocks constitutes the best match with respect to the coded macroblock, and then produces the corresponding motion vector at its output. The motion vectors produced by the logic elements **111** and **113** are examined by a logic circuit **115** for a 2:1 relationship, and if such a relationship is determined to exist between these motion vectors, then a field picture dual-prime motion estimation prediction mode is selected and the corresponding motion vector generated.

The similarities between the techniques and architectures described for motion estimation of frame pictures and field pictures immediately suggests that a unified architecture can be implemented which supports all three prediction modes allowed for frame pictures as well as all three prediction modes allowed for field pictures. Combining all of the techniques previously described, such an architecture requires knowledge only of the picture structure (frame or field) and type (I, P, or B) to determine the optimal prediction mode (i.e., the mode that yields the smallest value of the error metric) and its corresponding motion vector(s) for any macroblock, and need only perform a single search operation to do so. When implemented using custom hardware, as required for real-time video (e.g., a live broadcast), motion estimation is the most hardware-intensive and expensive operation in a digital video coding system. When implemented in computer software, as is usually done when the coding need not be performed in real-time (e.g., the coding of a DVD), the motion estimation algorithm is the most computationally complex and intensive part of the digital video coding algorithm. In either case, the methods and architectures of the present invention result in a means of significantly improving the video compression efficiency and, hence, the resulting picture quality, without the need for either greater hardware costs or higher computational complexity.

Although preferred embodiments of the present invention have been described in detail hereinabove, it should be clearly understood that many variations and/or modifications of the basic inventive concepts taught herein that may appear to those skilled in the pertinent art will still fall within the spirit and scope of the present invention, as defined in the appended claims.

What is claimed is:

1. A method for motion coding an uncompressed digital video data stream, including the steps of:

comparing pixels of a first pixel array in a picture currently being coded with pixels of a plurality of second pixel arrays in at least one reference picture and concurrently performing motion estimation for each of a plurality of different prediction modes in order to determine which of the prediction modes is an optimum prediction mode;

determining which of the second pixel arrays constitutes a best match with respect to the first pixel array for the optimum prediction mode; and,

generating a motion vector for the first pixel array in response to the determining step.

2. The method as set forth in claim **1**, wherein the first and second pixel arrays each have a size and structure defined by an MPEG standard.

3. The method as set forth in claim **1**, wherein the method is implemented using a motion estimation search engine of a digital video encoder.

4. The method as set forth in claim **3**, wherein the digital video encoder is an MPEG-2 digital video encoder.

5. The method as set forth in claim **1**, wherein the motion coding is performed in accordance with an MPEG standard.

6. The method as set forth in claim **5**, further comprising the initial step of providing information identifying a picture type of the first pixel array and using this information in the comparing step.

7. The method as set forth in claim **5**, wherein the different prediction modes are:

frame prediction mode for frame pictures;

field prediction mode for frame pictures;

field prediction mode for field pictures;

16×8 prediction mode for field pictures;

dual-prime prediction mode for field pictures; and,

dual-prime prediction mode for frame pictures.

8. The method as set forth in claim **1**, wherein the different prediction modes are:

frame prediction mode for frame pictures; and,

field prediction mode for frame pictures.

9. The method as set forth in claim **1**, wherein the different prediction modes are:

field prediction mode for field pictures; and,

16×8 prediction mode for field pictures.

10. The method as set forth in claim **8**, wherein the different prediction modes further include a dual-prime prediction mode for frame pictures.

11. The method as set forth in claim **9**, wherein the different prediction modes further include a dual-prime prediction mode for field pictures.

12. The method as set forth in claim **1**, wherein the different prediction modes are:

frame prediction mode for frame pictures;

field prediction mode for frame pictures;

field prediction mode for field pictures; and,

16×8 prediction mode for field pictures.

13. A device that implements the method set forth in claim **1**.

14. A device that implements the method set forth in claim **7**.

15. A device that implements the method set forth in claim **8**.

16. A device that implements the method set forth in claim **9**.

17. A method for motion coding a digital video data stream comprised of a sequence of pictures having top and bottom fields, the method including the steps of:

comparing pixels of a first portion of a current macroblock of the top field of a current picture with pixels of each of a plurality of correspondingly-sized portions of a macroblock of a top field of an anchor picture in accordance with a prescribed search metric, and producing a first error metric for each comparison;

comparing pixels of the first portion of the current macroblock of the top field of the current picture with pixels of each of the plurality of correspondingly-sized portions of a macroblock of- a bottom field of the anchor picture in accordance with the prescribed search metric, and producing a second error metric for each comparison;

comparing pixels of a second portion of a current macroblock of the bottom field of the current picture with pixels of each-of the plurality of correspondingly-sized portions of the macroblock of the top field of the anchor picture in accordance with the prescribed search metric, and producing a third error metric for each comparison;

comparing pixels of the second portion of the current macroblock of the bottom field of the current picture with pixels of each of the plurality of correspondingly-sized portions of the macroblock of the bottom field of the anchor picture in accordance with the prescribed search metric, and producing a fourth error metric for each comparison;

summing the first and fourth error metrics to produce a first composite error metric;

summing the second and third error metrics to produce a second composite error metric; and

determining which of the first, second, third, and fourth error metrics, and first and second composite error metrics has the lowest value, and selecting one of a plurality of possible motion estimation prediction modes on the basis of such determination.

18. The method as set forth in claim **17**, wherein all of the comparing steps are performed concurrently, and both of the summing steps are performed concurrently.

19. The method as set forth in claim **18**, wherein:

the first portion of the current macroblock of the top field of the current picture comprises a first half-portion of the current macroblock of the top field of the current picture; and,

the second portion of the current macroblock of the bottom field of the current picture comprises a second half-portion of the current macroblock of the bottom field of the current picture.

20. The method as set forth in claim **19**, wherein the dimensions of each of the first and second portions of the current macroblock of the top and bottom fields of the current picture are 16-by-8 pixels.

21. The method as set forth in claim **20**, wherein the digital video data stream comprises an MPEG-2 digital video data stream.

22. The method as set forth in claim **18**, wherein the plurality of possible motion estimation prediction modes includes frame and field prediction modes for frame pictures in accordance with the MPEG-2 standard.

23. The method as set forth in claim **19**, wherein the plurality of possible motion estimation prediction modes includes frame and field prediction modes for frame pictures in accordance with the MPEG-2 standard.

24. A device that implements the method set forth in claim **19**.

25. A method for motion coding a digital video data stream comprised of a sequence of pictures, the method including the steps of:

comparing pixels of a first portion of a top half of a current macroblock of a current picture with pixels of each of a plurality of correspondingly-sized portions of a macroblock of a top field of an anchor picture in accordance with a prescribed search metric, and producing a first error metric for each comparison;

comparing pixels of the first portion of the top half of the current macroblock of the current picture with pixels of each of the plurality of correspondingly-sized portions of a macroblock of a bottom field of the anchor picture in accordance with the prescribed search metric, and producing a second error metric for each comparison;

comparing pixels of a second portion of a bottom half of a current macroblock of the current picture with pixels of each of the plurality of correspondingly-sized portions of the macroblock of the top field of the anchor picture in accordance with the prescribed search metric, and producing a third error metric for each comparison;

comparing pixels of the second portion of the bottom half of the current macroblock of the current picture with pixels of each of the plurality of correspondingly-sized portions of the macroblock of the bottom field of the anchor picture in accordance with the prescribed search metric, and producing a fourth error metric for each comparison;

summing the first and third error metrics to produce a first composite error metric;

summing the second-and fourth error metrics to produce a second composite error metric; and

determining which of the first, second, third, and fourth error metrics, and first and second composite error metrics has the lowest value, and selecting one of a plurality of possible motion estimation prediction modes on the basis of such determination.

26. The method as set forth in claim **25**, wherein all of the comparing steps are performed concurrently, and both of the summing steps are performed concurrently.

27. The method as set forth in claim **26**, wherein:

the first portion of the top half of the current macroblock of the current picture comprises a first half-portion of the top half of the current macroblock of the current picture; and,

the second portion of the bottom half of the current macroblock of the current picture comprises a second half-portion of the bottom half of the current macroblock of the current picture.

28. The method as set forth in claim **27**, wherein the dimensions of each of the first and second portions of the top and bottom halves of the current macroblock of the current picture are 16-by-8 pixels.

29. The method as set forth in claim **28**, wherein the digital video data stream comprises an MPEG-2 digital video data stream.

30. The method as set forth in claim **25**, wherein the plurality of possible motion estimation prediction modes

includes field and 16×8 prediction modes for field pictures in accordance with the MPEG-2 standard.

31. The method as set forth in claim **26**, wherein the plurality of possible motion estimation prediction modes includes field and 16×8 prediction modes for field pictures in accordance with the MPEG-2 standard.

32. A device that implements the method set forth in claim **26**.

33. A method for motion coding a digital video data stream comprised of a sequence of pictures having top and bottom fields, the method including the steps of:

comparing pixels of a first portion of a current macroblock of the top field of a current picture with pixels of each of a plurality of correspondingly-sized portions of a macroblock of a top field of an anchor picture in accordance with a prescribed search metric, and producing a first error metric for each comparison;

comparing pixels of the first portion of the current macroblock of the top field of the current picture with pixels of each of the plurality of correspondingly-sized portions of a macroblock of a bottom field of the anchor picture in accordance with the prescribed search metric, and producing a second error metric for each comparison;

comparing pixels of a second portion of a current macroblock of the bottom field of the current picture with pixels of each of the plurality of correspondingly-sized portions of the macroblock of the top field of the anchor picture in accordance with the prescribed search metric, and producing a third error metric for each comparison;

comparing pixels of the second portion of the current macroblock of the bottom field of the current picture with pixels of each of the plurality of correspondingly-sized portions of the macroblock of the bottom field of the anchor picture in accordance with the prescribed search metric, and producing a fourth error metric for each comparison;

producing first, second, third, and fourth motion vectors on the basis of the first, second, third, and fourth error metrics, respectively; and,

examining the first, second, third, and fourth motion vectors to determine whether a prescribed relationship between them is present, and if so, selecting a frame picture dual-prime motion estimation prediction mode.

34. The method as set forth in claim **33**, wherein all of the comparing steps are performed concurrently.

35. A device that implements the method set forth in claim **34**.

36. A method for motion coding a digital video data stream comprised of a sequence of pictures, the method including the steps of:

comparing pixels of a first portion of a top half of a current macroblock of a current picture with pixels of each of a plurality of correspondingly-sized portions of a macroblock of a top field of an anchor picture in accordance with a prescribed search metric, and producing a first error metric for each comparison;

comparing pixels of the first portion of the top half of the current macroblock of the current picture with pixels of each of the plurality of correspondingly-sized portions of a macroblock of a bottom field of the anchor picture in accordance with the prescribed search metric, and producing a second error metric for each comparison;

comparing pixels of a second portion of a bottom half of a current macroblock of the current picture with pixels of each of the plurality of correspondingly-sized portions of the macroblock of the top field of the anchor picture in accordance with the prescribed search metric, and producing a third error metric for each comparison;

comparing pixels of the second portion of the bottom half of the current macroblock of the current picture with pixels of each of the plurality of correspondingly-sized portions of the macroblock of the bottom field of the anchor picture in accordance with the prescribed search metric, and producing a fourth error metric for each comparison;

summing the first and third error metrics to produce a first composite error metric;

summing the second and fourth error metrics to produce a second composite error metric;

producing first and second motion vectors on the basis of the first and second composite error metrics, respectively; and,

examining the first and second motion vectors to determine whether a prescribed relationship between them is present, and if so, selecting a field picture dual-prime motion estimation prediction mode.

37. The method as set forth in claim **36**, wherein all of the comparing steps are performed concurrently, both of the summing steps are performed concurrently, and both of the producing steps are performed concurrently.

38. A device that implements the method set forth in claim **37**.

39. A motion estimation search system that concurrently performs motion estimation using each of a plurality of different motion estimation prediction modes and then selects the prediction mode that produces the optimum result.

40. The motion estimation search system as set forth in claim **39**, wherein the motion estimation search system is included in an MPEG-2 digital video encoder.

41. A method, including the steps of:

concurrently performing motion estimation using each of a plurality of different motion estimation prediction modes;

comparing the results produced using each different prediction mode; and,

selecting the prediction mode that produced an optimum result; and,

generating one or more motion vectors using the selected prediction mode.

42. The method as set forth in claim **41**, wherein the prediction modes include at least a plurality of the prediction modes specified by the MPEG-2 standard.

* * * * *

EXHIBIT D

US006470345B1

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

(10) **Patent No.: US 6,470,345 B1**
(45) **Date of Patent: Oct. 22, 2002**

(54) **REPLACEMENT OF SUBSTRINGS IN FILE/DIRECTORY PATHNAMES WITH NUMERIC TOKENS**

(75) Inventors: **Edward Doutre**, Cary; **John Christian Fluke**, Raleigh, both of NC (US)

(73) Assignee: **International Business Machines Corporation**, Armonk, NY (US)

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

(21) Appl. No.: **09/477,771**

(22) Filed: **Jan. 4, 2000**

(51) **Int. Cl.**^{7} **G06F 17/30**
(52) **U.S. Cl.** **707/100**; 707/102; 707/200; 707/501.1
(58) **Field of Search** 707/3–5, 100–102, 707/501.1, 1, 9, 10, 104.1; 709/219, 220, 102, 106, 201, 223, 224; 713/2; 704/9; 706/45, 49, 58; 710/36; 717/113, 121

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,325,091	A		6/1994	Kaplan et al.	341/51
5,325,531	A	*	6/1994	McKeeman et al.	717/112
5,475,743	A	*	12/1995	Nixon et al.	379/114.15
5,497,492	A	*	3/1996	Zbikowski et al.	709/321
5,525,982	A		6/1996	Cheng et al.	341/51
5,574,903	A	*	11/1996	Szymanski et al.	707/1
5,577,249	A		11/1996	Califano	707/100
5,608,901	A	*	3/1997	Letwin	707/205
5,652,876	A	*	7/1997	Ashe et al.	703/26
5,659,755	A		8/1997	Strohacker	708/203
5,666,114	A		9/1997	Brodie et al.	341/50
5,740,353	A	*	4/1998	Kreulen et al.	714/33
5,778,255	A		7/1998	Clark et al.	710/68
5,778,361	A	*	7/1998	Nanjo et al.	707/2
5,873,118	A	*	2/1999	Letwin	707/205
6,021,433	A	*	2/2000	Payne et al.	340/7.29
6,105,027	A	*	8/2000	Schneider et al.	707/10
6,185,575	B1	*	2/2001	Orcutt	707/200
6,195,689	B1	*	2/2001	Bahlmann	709/217
6,199,068	B1	*	3/2001	Carpenter	340/870.03
6,266,678	B1	*	7/2001	McDevitt et al.	707/10
6,366,988	B1	*	4/2002	Skiba et al.	707/203
6,374,250	B2	*	4/2002	Ajtai et al.	341/50

OTHER PUBLICATIONS

"Separation of file/directory pathname canonicalization from validation", Research Disclosure, IBM Corporation, Nov. 1999, p. 1.*
Peterson, Larry "The Profile Naming Service", ACM Transactions on Computer Systems, vol. 6, No. 4, Nov. 1988, pp. 341–364.*
Santry, Douglas J. et al., "Elephant: The File System that Never Forgets", Proceedings of Seventh Workshop on Hot Topics in Operating Systems, Mar. 29–30, 1999, pp. 2–7.*
Bach, M.J., "Design of the Unix Operating System," pp. 76–88, Prentice–Hall, Inc., 1986.

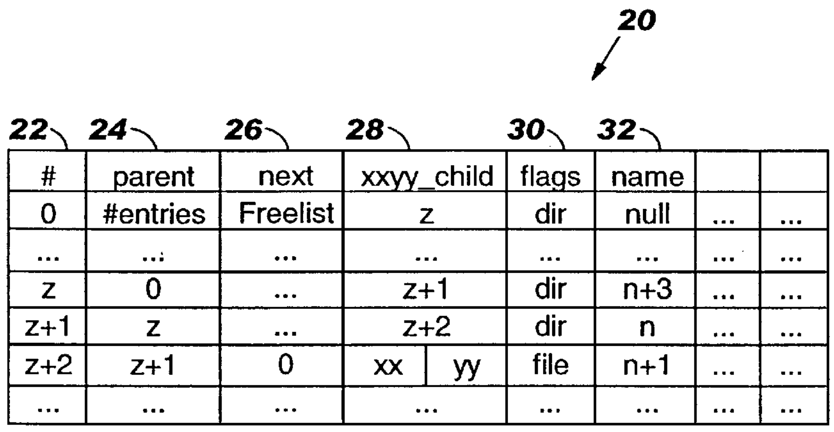
(List continued on next page.)

Primary Examiner—Jean M. Corrielus
Assistant Examiner—Shahid Alam
(74) *Attorney, Agent, or Firm*—J. Bruce Schelkopf

(57) **ABSTRACT**

A method and system for replacing substrings in file and directory pathnames with numeric tokens. A name string to be converted is first read; the current working directory and name string are canonicalized to form a pathname containing the substrings. The pathname is parsed and each substring is searched in a string dictionary to locate a corresponding numeric token. The string dictionary that is created associates token values with substrings, so that there is a one-to-one correspondence. The returned list of tokens for the parsed pathname are validated through a lookup process in a directory table. If the parsed pathname is successfully validated, the tokens are then used in subsequent file operations such as create, delete, open, rename and compare files.

48 Claims, 7 Drawing Sheets


20
22
24
26
28
30
32

#	parent	next	xxyy_child		flags	name		
0	#entries	Freelist	z		dir	null	...	...
...	...	...	...		...	...	...	...
z	0	...	z+1		dir	n+3	...	...
z+1	z	...	z+2		dir	n	...	...
z+2	z+1	0	xx	yy	file	n+1	...	...
...	...	...	...		...	...	...	...

OTHER PUBLICATIONS

"Complexity of Preprocessor in MPM Data Compression System," Kiefer et al., Proceedings DCC '98 Data Compression Conference, Mar. 30–Apr. 1 1998, p. 554 (abstract only).

"An LR Substring Parser Applied in a Parallel Environment," Clarke, G, et al., Journal of Parallel and Distributed Computing, vol. 35, No. 1, May 25, 1996, pp. 2–17, (abstract only).

* cited by examiner

FIG. 1

10

12	14
...	...
n	ThisIsALongDirectoryName
n+1	ThisIsALongFileName.ThisIsALongFileExtension
n+2	ADXLXZTN
n+3	VFS
...	...

FIG. 2

20

22	24	26	28		30	32		
#	parent	next	xxyy_child		flags	name		
0	#entries	Freelist	z		dir	null	...	...
...	...	...	...		...	...	...	...
z	0	...	z+1		dir	n+3	...	...
z+1	z	...	z+2		dir	n	...	...
z+2	z+1	0	xx	yy	file	n+1	...	...
...	...	...	...		...	...	...	...

FIG. 3

h0:vfs/~~00/	files:	00, 01, 02, ... FE,FF
h0:vfs/~~01/	files:	00, 01, 02, ..., FE, FF
h0:vfs/~~...	files:	00, 01, 02, ..., FE, FF
h0:vfs/~~FE/	files:	00, 01, 02, ..., FE, FF
h0:vfs/~~FF/	files:	00, 01, 02, ..., FE, FF

FIG. 4

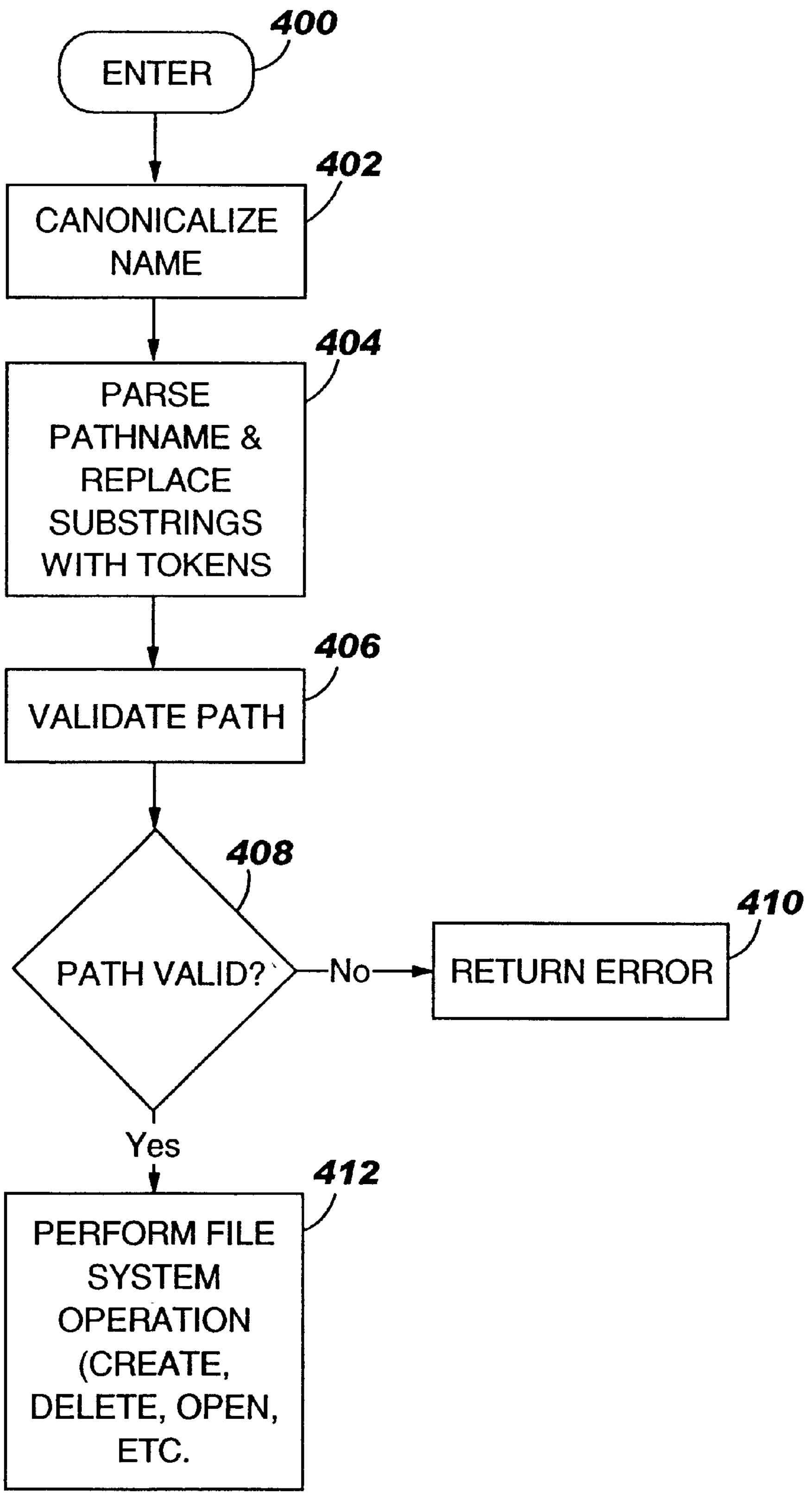

400
ENTER
402
CANONICALIZE NAME
404
PARSE PATHNAME & REPLACE SUBSTRINGS WITH TOKENS
406
VALIDATE PATH
408
PATH VALID?
No
410
RETURN ERROR
Yes
412
PERFORM FILE SYSTEM OPERATION (CREATE, DELETE, OPEN, ETC.

FIG. 5

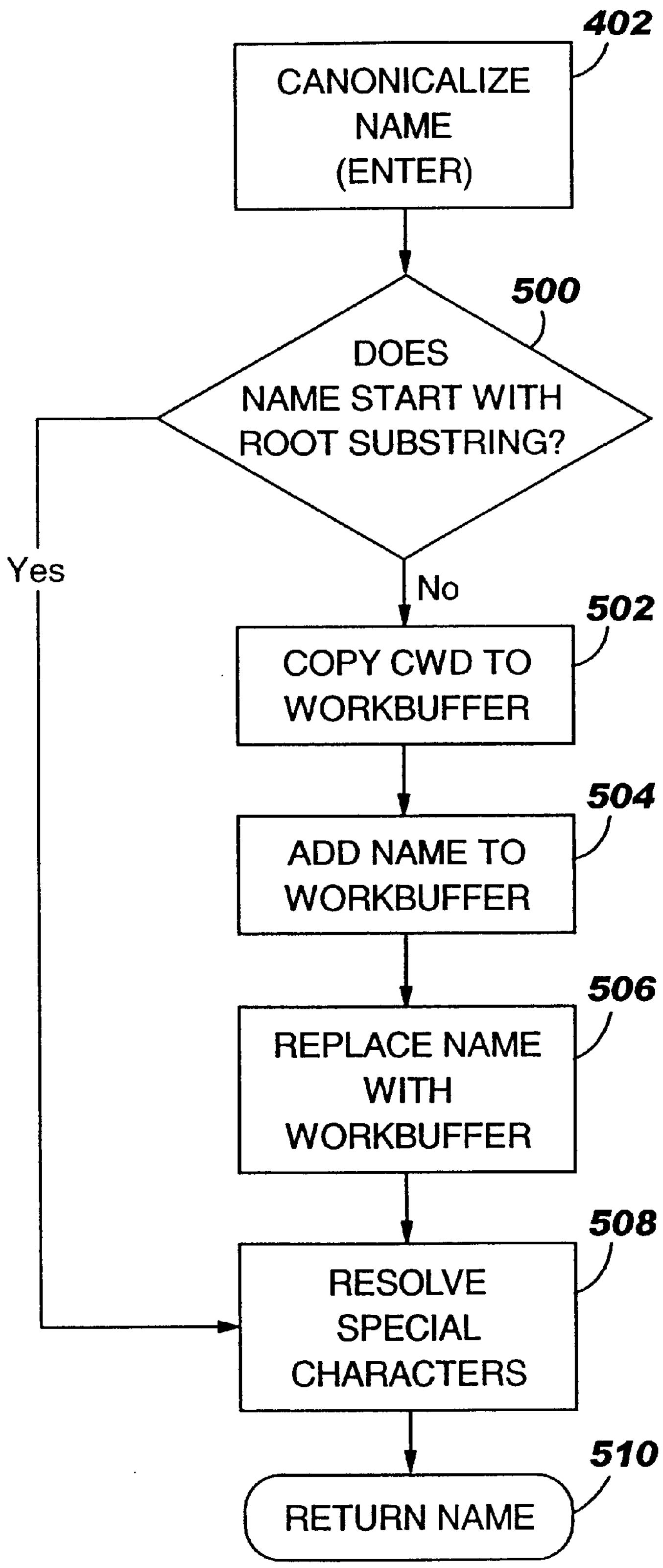

402
CANONICALIZE NAME (ENTER)
500
DOES NAME START WITH ROOT SUBSTRING?
Yes
No
502
COPY CWD TO WORKBUFFER
504
ADD NAME TO WORKBUFFER
506
REPLACE NAME WITH WORKBUFFER
508
RESOLVE SPECIAL CHARACTERS
510
RETURN NAME

FIG. 6

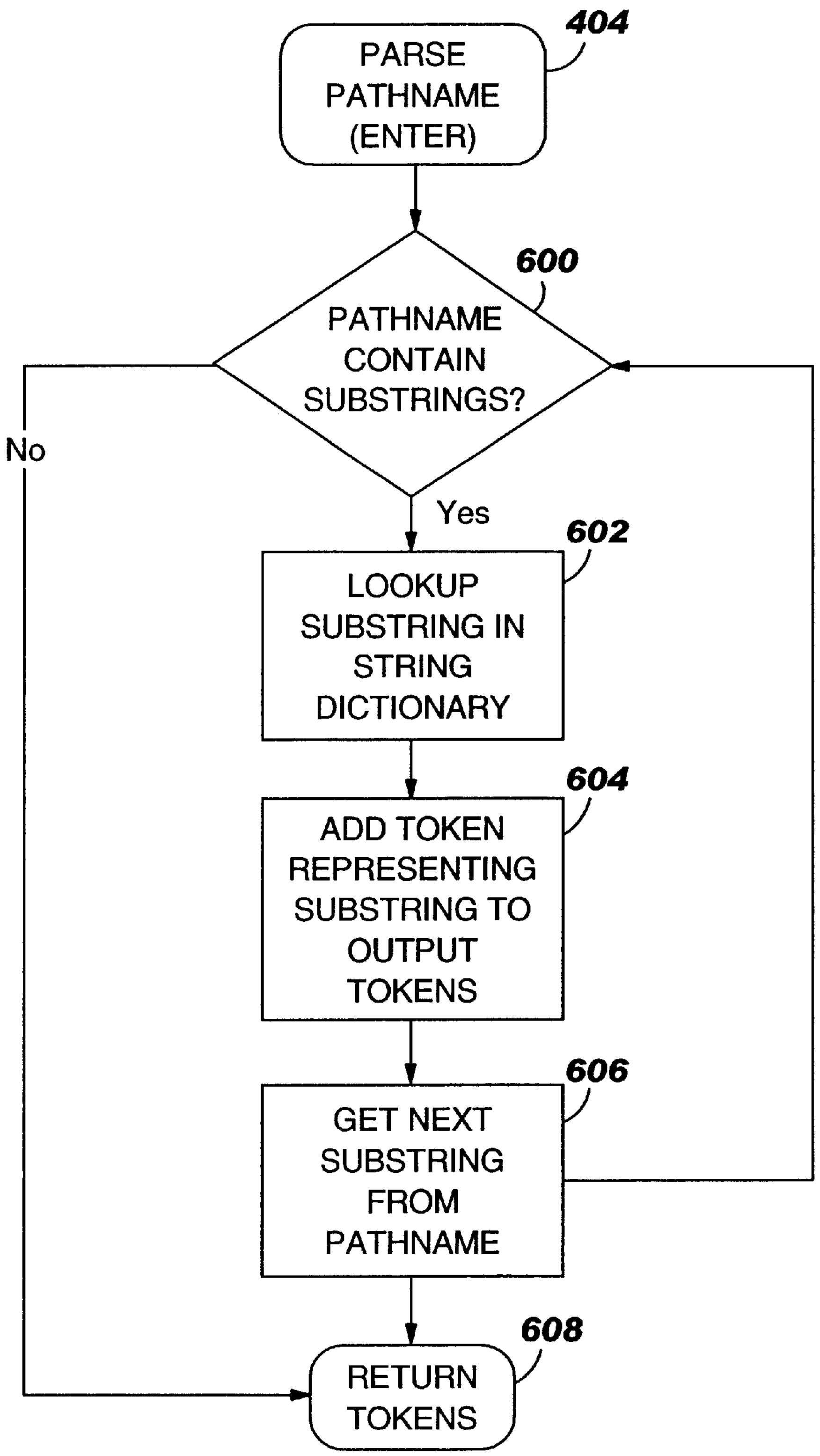

404
PARSE PATHNAME (ENTER)
600
PATHNAME CONTAIN SUBSTRINGS?
No
Yes
602
LOOKUP SUBSTRING IN STRING DICTIONARY
604
ADD TOKEN REPRESENTING SUBSTRING TO OUTPUT TOKENS
606
GET NEXT SUBSTRING FROM PATHNAME
608
RETURN TOKENS

FIG. 7

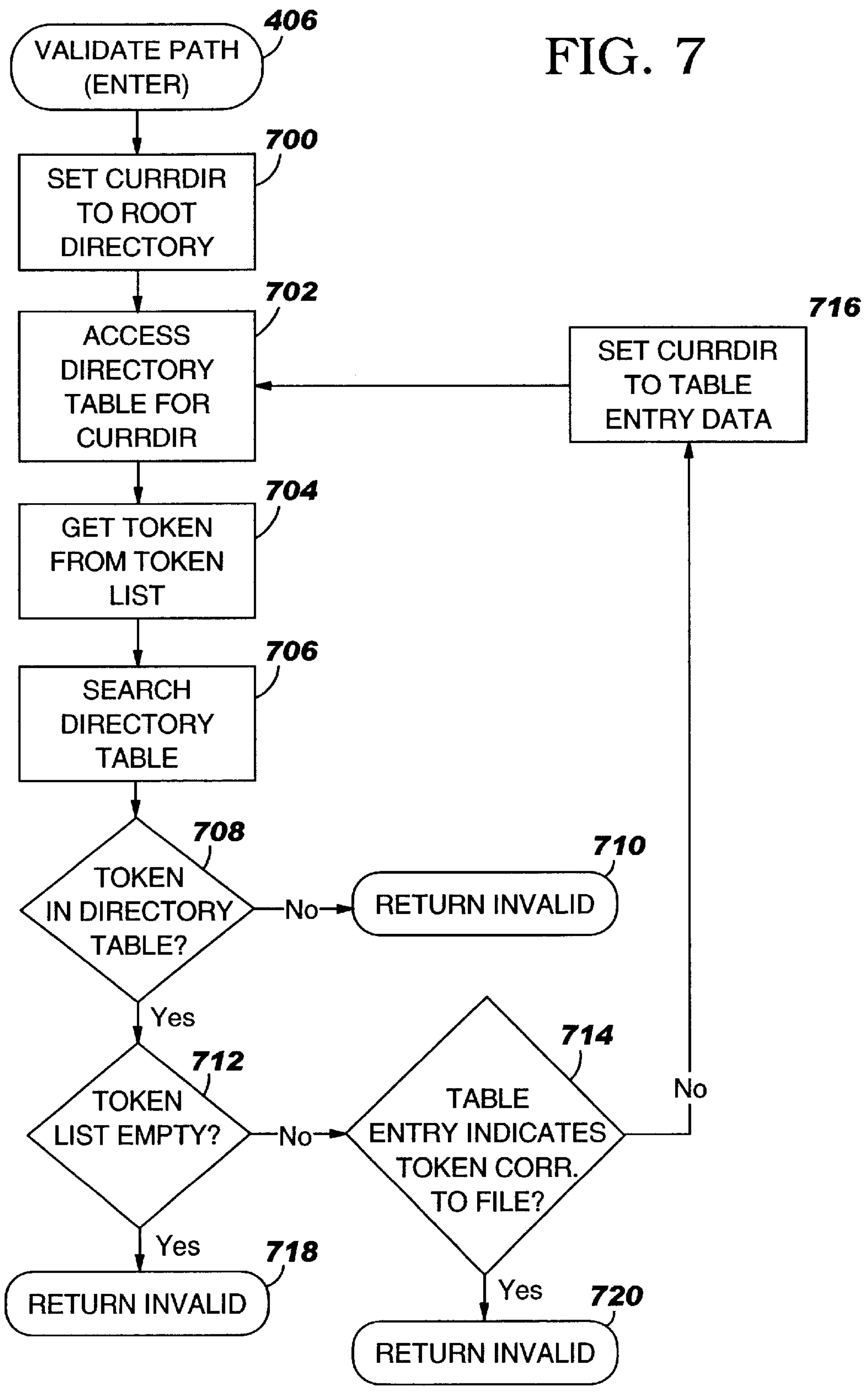

406
VALIDATE PATH (ENTER)
700
SET CURRDIR TO ROOT DIRECTORY
702
ACCESS DIRECTORY TABLE FOR CURRDIR
704
GET TOKEN FROM TOKEN LIST
706
SEARCH DIRECTORY TABLE
708
TOKEN IN DIRECTORY TABLE?
No
710
RETURN INVALID
Yes
712
TOKEN LIST EMPTY?
No
714
TABLE ENTRY INDICATES TOKEN CORR. TO FILE?
No
716
SET CURRDIR TO TABLE ENTRY DATA
Yes
718
RETURN INVALID
Yes
720
RETURN INVALID

FIG. 8A

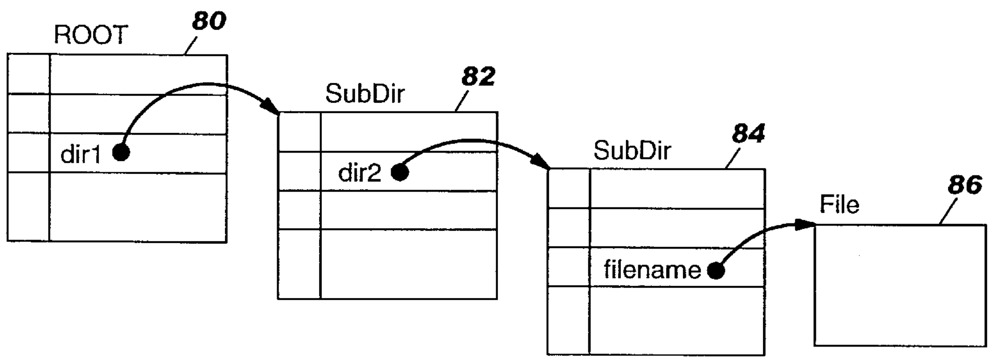
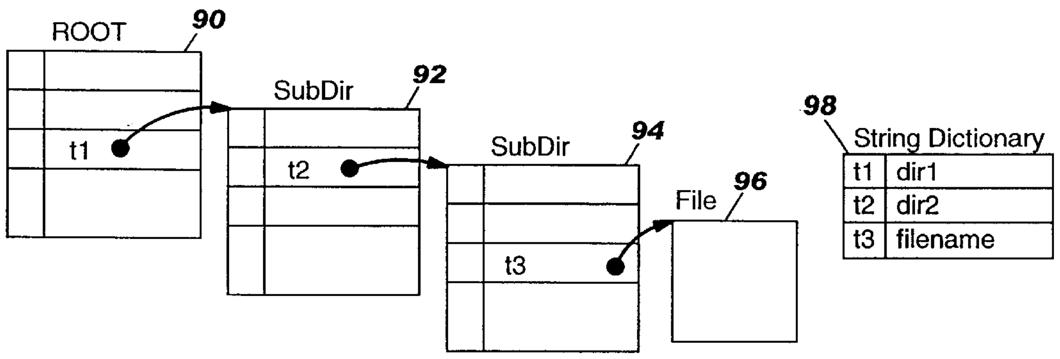


FIG. 8B



t1	dir1
t2	dir2
t3	filename

FIG. 9

100

102 104 106

token	substring	size
t1	"Test_1"	6
t2	"Source"	6
t3	"filename1"	9
t4	"text"	4
t5	"filename2"	9
t6	"Output"	6
t7	"binary"	6

REPLACEMENT OF SUBSTRINGS IN FILE/DIRECTORY PATHNAMES WITH NUMERIC TOKENS

BACKGROUND OF THE INVENTION

The present invention is generally related to data processing systems; and more particularly is related to a method and system for the replacement of substrings in file and directory pathnames with numeric tokens.

Most file systems will complete a partial file or directory specification by using the current working directory information along with whatever partial information is given. This process of creating a complete, syntactically correct specification (the canonical form) is sometimes referred to as "canonicalization". This canonical form is important, since it completely and uniquely identifies the file system resource, whether a file, directory or some other type of resource.

Another important task is the semantic validation of a path, made up of the root, intermediate directories, and file or directory specification. All intermediate directories must be valid for a pathname to refer to a valid file system resource. The exception is that the final term, whether a file, directory or other name, might not exist at the time of validation, since the operation requested of the file system may be to create, or indeed, to check whether it exists.

These two tasks are often intertwined in a single function or set of functions. This makes sense in some file systems, such as UNIX's file system (UFS), where all resources are local and creations, modifications and deletions are all within the same data scope of an operating system process and can be easily synchronized.

The combination of these two functions can also effect some savings by being more efficient. If the current working directory for a given process is taken to be always valid (which assumes some method to prevent other processes from modifying that file system information while a process is "in it"), then validation of a path can start with the partial information specified by the user of the file system.

However useful this method of combining these two functions can be, it should always be remembered that these are two separate tasks. Severe performance penalties can be the cost of forgetting this. During recent development of a Virtual File System (VFS) and related network file system (NFS) work by the inventors, it was found that some NFS clients were sending remote procedure call (RPC) requests to validate each intermediate part of the path (via NFS_LOOKUP) instead of sending the full path as far as it was thought to be valid. This means in many cases 12 to 15 RPCs instead of a single RPC.

In the design of the file system that is structured on a client/server split, where the client portion keeps track of the current working directory and therefore has to perform the canonicalization, the path validation can often only be efficiently done by the server. The inventors' research has shown that in most cases even where there is no client/server split, it is advantageous to separate canonicalization from validation and perform these two operations in a close sequence, but not interleaving validation of intermediate path information with a forming of a canonical name. This results in a simpler implementation and superior performance, especially in a network environment.

SUMMARY OF THE INVENTION

In a network of computers, there is often a need to extend some operating systems' file systems to accommodate file

and directory names that are not supported natively. When implementing Java Virtual Machines (JVMs) on file systems that only support "8.3" names (up to eight characters for the name and up to three characters for extension or type) this becomes very apparent. A trivial example is: "SomeJavaApplication.class", which violates both the eight character name and the three character extension limits. Special characters, DBCS (Double Byte Character Set), uppercase and lowercase letters, spaces within names and a host of other limitations can cause problems that limit the usefulness of an otherwise desirable file system.

A virtual file system (VFS) has been implemented that allows clients to map many names that use these problem characters and can far exceed the length of the file or directory name or total length of a "path". In general, a VFS is an indirection layer that handles the file-oriented system calls and calls the necessary functions in the physical file system code to perform input/output. The VFS consists of a Name Space Server accessed via TCP/IP sockets and a run-time VFS client. In a sense the run-time client intercepts names that are allowed to exceed the limits of the native file system and sends them to the Name Space Server to be converted into names that are supported natively.

In dealing with file/directory pathnames, the number of sometimes quite lengthy strings poses a significant problem, especially when these are broken into substrings which then are constantly compared to other substrings. By parsing the strings into their semantically correct substrings and replacing those substrings with unique numeric tokens, a significant improvement is realized in the storage of the strings as well as better performance in comparing those substrings. Since each substring (typically a subdirectory, filename or extension) is replaced with a numeric value, these numeric values can be arithmetically compared (e.g., is a ==b) instead of string compared (i.e., are all characters the same, what about uppercase vs. lowercase, etc.). This alone represents a substantial improvement in performance. In addition, by keeping a string dictionary, which the token uniquely indexes, only one copy is kept of any substring. This too can represent a substantial savings in the amount of storage needed to implement a file system.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is better understood by reading the following detailed description of the preferred embodiment in conjunction with the accompanying drawings, wherein:

FIG. **1** illustrates an example of the partial format of a string dictionary used in the preferred embodiment of the present invention;

FIG. **2** illustrates an example of the format of a mapping database used in the preferred embodiment of the present invention;

FIG. **3** illustrates the structure of a physical directory file layout;

FIG. **4** illustrates a high level flowchart of the functions of the token replacement mechanism of the present invention;

FIG. **5** illustrates a flowchart of the canonicalization process of the present invention;

FIG. **6** illustrates a flowchart of the parsing process of the present invention;

FIG. **7** illustrates a flowchart of validation process of the present invention;

FIGS. **8A–8B** depict the prior art method of storing directory and file names and the mechanism used in the present invention to store directory and file names using numeric tokens; and

FIG. 9 illustrates an example of a string dictionary used in the preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Due to a need for directory and file names that are greater than can currently be supported in the existing File Allocation Table (FAT) file system, a Virtual File System (VFS) can be designed and implemented that can support much longer path lengths and completely avoid the 8.3 directory and file name limitations.

The VFS can be accessed by programs and users by referring to a virtual drive, e.g., 'VFS:'. The Virtual File Names (VFNs) are of the form:

node::VFS:/directory/ . . . /filename

where 'node::' is the machine node network name; and the 'directory' and 'filename' specifiers are permitted to each be of up to 255 characters, with a total path length of 270 characters.

Underlying the VFS is the normal physical file system with all of the usual operating system features and limitations. The Real File Names (RFNs) will be placed under a system sub-directory on the controller or file server (h0:/VFS/ is the current name) and are of the form:

node::device_name:VFS/~~X_1X_2/Y_1Y_2

where 'node::' is the machine network node name; the 'device_name:' is the usual 'h0:' or 'h1:' pertaining to a specific implementation (IBM 4690 OS); hard disk device name: 'VFS/' is the system sub-directory under which all VFS system data files and all of the RFNs lie; '~~X_1X_2/' are special sub-directories created to hold the actual files that are in VFS ('X_1,' and 'X_2' are the ASCII representations of the hexadecimal digits from '0' to 'F', i.e. one of '00', '01', '02', . . . 'FE', 'FF'); 'Y_1Y_2' gives the physical name of the file (again the ASCII representation of two hex digits). The implementation of the invention; however, is not limited to any particular operating system platform, and those skilled in the art will readily appreciate that the invention can be implemented on many other platforms including Microsoft Corp.' s Windows 95/98 and Windows NT, IBM Corp.'s OS/2 as well as Sun Microsystems' Java Virtual Machine (JVM).

This scheme provides for 256*256 (64K) RFNs in a VFS logical volume (i.e. the 'VFS:' drive). All access to these physical files is through VFS code via their Virtual File Names (VFNs) except for certain system routines and utilities. An application or user will never see an RFN.

While actual files are represented as physical files, directories are only represented in VFS database files, along with information on the physical files within those directories. Information is kept as to distribution attributes as well as which nodes in the networked system actually have the physical files.

The VFS is implemented using a client/server split, where each user of the VFS accesses the files and directories by calling functions on their local machine. This is true for controllers as well as terminals. The client functions open a TCP/IP socket to the currently active controller in the system (perhaps even on the same physical machine) and make requests of the VFS server.

The function of this VFS Name Space Server is analogous to that of the TCP/IP network Domain Name Server (DNS) or bind server which maps IP addresses and node names back and forth. However, the VFS Name Space Server is involved in many simultaneous operations performed by numerous applications and therefore has some severe performance requirements and also does not have the IP name restrictions that limit the total number of usable names. The typical DNS implementation also does not face the dynamic creation and deletion of names and the attendant management problems that entails. Some of these differences are readily apparent when given some examples. A typical DNS would map the IP node name "mymachine.mysubnet.mydomain.net" to an IP address of the (for example) 9.67.5.1 and another IP node name with the same "mysubnet.my domain.net" would only differ in the least significant number (the ".1" might for instance become ".2"). These restrictions are documented in various Request For Comments (RFCs) documents (for example, RFCs 1034/2535) and are tightly controlled because of their impact on IP routers, which depend on these addresses to deliver packets to the correct sub-network.

In the VFS Name Space Server, Virtual File Names (VFNs) are reusable, in part, as long as different subdirectories make them unique and the real file names (RFNs) that they map to can be distinguished in some fashion. In addition, the VFNs can be very long in that they may contain a large number of subdirectories and there is no direct correlation between these intermediate components and the allocation of RFNs. As an example, the VFN "nodename::devicename:\dir**1**\dir**2**\dir**3**\dir**4**\dir**5**\file.name" can be mapped to the RFN "node name:: C:\xx\yz" where "xx" is a subdirectory (used to improve performance) and "yz" is an actual file. At the same time, a VFN of the "othernode::otherdevice:\otherdirectory\filename.extension" may map to RFN "othernode::otherdevice:\xx\yz", where "z" differs from "y" by only one character. In other words, the allocation of RFNs is totally independent of the logical proximity of the Virtual File Name.

The VFS Name Space Server is composed of three distinct parts: the string heap, the mapping database and the RFN allocation subsystem. The parsing of a canonical file/directory name into its individual component substrings and replacement with numeric tokens involves the specification of the VFN as either a file name (where file name is of the form "node::device:\directory\. . . \filename.extension") or a directory name (where directory name is of the form "node::device:\directory\. . . "). The string is decomposed by a parser which recognizes the subdirectory delimiters '\' or '/' and replaces each component of the path with a numeric token which uniquely specifies the original substring. The VFN is then located in the mapping database which associates with each VFN, in the case of an actual file, an RFN that is obtained from the RFN allocation subsystem. Virtual directories are also placed into the mapping database, but do not map to an RFN since directory VFNs are virtual.

The VFS server creates and maintains a database to provide information for all clients about the files and directories in the VS volume. The most frequent request is the 'find' operation which returns a physical name (i.e., an RFN) to allow the client code to create, open, delete and otherwise manipulate the file. Some requests are done completely on the server side (e.g., the 'rename' function) and some require more than one interaction between the client code and the server.

However, the application software does not normally use these client functions directly. Instead, the system-provided runtime libraries, such as PORTLIB.DLL for 32-bit 'C' programs, the Visual Age runtimes for 32-bitC/C^{++} programs, as well as 16-bit link86able.LIB routines, call the VFS client functions "under-the-covers," providing transparent access to files and directories in the VFS volume.

Certain 'trusted' code, such as the Java Virtual Machine (JVM) and the command interpreter (the 'command prompt' and .BAT file processing code) can use the functions directly due to performance and possibly other reasons.

The client portion of the VFS consists of several functions that access the VFS server via TCP/IP sockets. The typical interaction between the client and server consists of a non-binding socketopen call by the client to the VFS server. If a socket is successfully opened, the client performs a send operation to transmit a request block and the client performs a receive operation to wait for the server's reply. If the non-binding open fails, then error recovery is needed to attempt to find another controller or to attempt in some other way to handle the problem. If the error turns out to be unrecoverable, then an appropriate error return is required.

The client code does more than just route the function requests to the server code. It also maintains the unique information that each client process uses, such as the current drive and working directory. The canonicalization process which turns each partial path/file name into a complete, fully qualified path/file name is done on the client, though the verification must be done on the canonical form at the server (the TransPath function).

The run-time VFS client contains APIs to deal with files and directories and passes requests to deal with native file system compliant names through the native operating system. The VFS hooks route the VFNs to the VFS Name Space Server via an RPC layer similar in some ways to the network file system (NFS) as described in various documents including RFC 1813. Note that file data transfer is not a function of this mechanism, only virtual file names (VFNs), their associated real file names (RFNs) and 'stat ( )' type information is transferred. All requests that deal with the data within a file are passed through the native file system.

The client code provides the following functions to users of the VFS:

int VFS_isinVFS(VFN * name);
int VFS_find(VFN *name, RFN *realname)
int VFS_create(VFN *name, RFN *realname)
int VFS_delete(VFN *name, RFN *realname)
int VFS_rename(VFN *from, VFN *to)
int VFS_stat(VFN *name, stat_struct *pstat)
int VFS_mkdir(VFS *dirname)
int VFS_rmdir(VFS *dirname)
int VFS_chdir(VFS *dirname)
char VFS_getcwd(VFN *dirname)
int VFS_readdir(VFSDIR *dir)

Each of these functions is invoked on the client and most interact with their server counterparts via a TCP/IP socket-based transport that conveys the request and the parameters to the server and the response from the server back to the client. A socket is opened from the client to the server for the duration of the call. This primitive RPC method is sufficient for all users of the service. The default port that the server "listens" on is currently "5555", but may change and this will be reflected in the Services TCP/IP configuration file.

Additional function is provided by the client code in several cases. A canonicalization process first performed on all VFNs, using the client's current working directory information to ensure a well-formed fully qualified path and file name is always provided to the server. Some functions are mostly or entirely implemented in the client code. The VFS_getcwd( )function simply returns the current working directory string to the caller. The VFS_chdir( ) function sets the client-maintained current working directory information after verifying with the VFS server that the new directory is valid.

Each of the functions are described as follows:

int VFS_isin VFS(VFN *name) returns either true or false, depending on whether the VFN contains one of the valid drive letters that indicate whether the specified file or directory name is in the VFS.

int VFS_find(VFN *name, RFN *realname) returns a string pointer (RFN *realname) that is the actual FAT/8.3 compliant name of the file. If the VFN passed in is not a file, but is a directory, an error code is returned that indicates this. If the VFN does not exist in the mapping database, an appropriate error code is returned.

int VFS_create(VFN *name, RFN *realname) allocates an RFN and associates it with the VFN passed in. It then returns a string pointer (RFN *realname) that is the actual FAT/8.3 compliant name of the file. No physical file is created. The client uses the normal runtime to actually open the file with the appropriate flags. Appropriate error codes are returned for conditions such as file already exists, no room available in file system (64 k file/dir limit) and so forth.

int VFS_delete(VFN *name, RFN *realname) deallocates the RFN and removes the VFN from the mapping database. It then returns a string pointer (RFN realname) that was the actual FAT/8.3 compliant name of the file. The physical file is not deleted. The client code uses the normal runtime to actually remove the physical file.

int VFS_rename (VFN *from, VFN *to) locates the first VFN (VFN *from) in the mapping database, validates that the second VFN (VFN *to) is valid and changes the VFN in the mapping database to reflect the changed name. If the name of a higher level directory in the *from or *to names (i.e. the path) is different, the file is moved from the first directory to the second. Error codes include: (1) error directory does not exist; (2) error *from file does not exist; (3) error *to file already exists; (4) error no room available in the file system (64 k file/dir limit). Conflicts in from/to names across nodes or involving DDA are considered errors as well and the system will not perform the rename. Alternatively an enhancement enables the client code to do a sequence of create/copy/delete in those instances. Rename request across different drive letters (e.g. 'rename adxlxztn::h0:\foo adxlxztn::vfs:/foo ') are also not permitted in the preferred embodiment but can be added in alternate embodiments in the same manner.

int VFS_stat(VFN *name, stat_struct *pstat) works only on virtual files and directories. The client sees the regular file system for file stat( ) information. The VFS server provides the information for directories. Error codes include file/directory name does not exist, etc. The server returns the RFN of a file to the client code if and only if the VFN denotes a file in the VFS, however, the length of the RFN field is used to return the mapping database index (used as the inode of directories) and a null RFN for directories. This is because there is no "normal" file system information for virtual directories (i. e., no RFN exists).

int VFS_mkdir(VFN *dirname) takes the fully qualified directory name (VFN *dirname) and adds it to the mapping database. Error codes include error directory already exists, error in pathname (a higher level directory does not exist), and error no room available in file system (64 k file/dir limit).

int VFS_rmdir(VFN *dirname) removes the filly qualified directory name (VFN *dirname) from the mapping database. Error codes include error in pathname (some directory in *dirname does not exist) and the directory is not empty.

int VFS_chdir(VFN *dirname) is not a server-side operation. The client-side code maintains the concepts of a current drive and working directory and this function allows the application code to manipulate the process-level current working directory. Verification is performed on the target directory name (VFN *dirname) and error codes include error target directory does not exist, etc.

char *VFS_getcwd(VFN *dirname) is not a server-side operation. Like the VFS_chdir( ) function it deals with the per-process current working directory. This function simply returns a pointer to the character string that contains the current working directory. No error codes are supported.

int VFS_readdir(VFSDIR *dir) takes a valid directory structure which includes a valid directory VFN, and on the first call will return the structure with information about the first file/directory within the target directory. Subsequent calls with the same VFSDIR structure returns updated information on the next directory entry within the specified target directory. Error codes include error directory does not exist.

The VFS server maintains the database of information and the actual underlying physical files. Communications between the client and server portions are via IP sockets using a custom RPC protocol which allows for a synchronous response from the server code to the client code for each request.

As requests are received for access to files and directories in the VFS, as indicated by a virtual device/drive (e.g., 'VFS:', they are routed to a file name hashing and sub-allocation scheme. This comprises three (3) parts as follows:

1. A string 'dictionary' that contains all strings for directory and file name identification.
2. A mapping database associating virtual file names (VFNs) with real file names (RFNs).
3. An RFN allocation scheme that keeps track of what real files currently exist.

The assumption is made that all files placed on this virtual drive are subject to name hashing and attendant sub-allocation, even if the names are valid in the 8.3 FAT file system. In addition, all directories created on this virtual drive will have entries in the mapping database. This solves several problems, including the fast location of files and certain directory operations.

In order to allow access to the data structures and algorithms, the following higher level functions are available on the server:

VFS_find( ) given a VFN, locates and returns an RFN.

VFS_create( ) given a VFN, allocates an RFN, updates the mapping database information and returns the RFN. Further calls to normal file system function are needed to actually create and open the real file.

VFS_delete( ) given a VFN, deallocates the associated RFN and cleans up the mapping database information.

VFS_rename( ) given two (2) VFNs, changes the mapping database information to reflect the new name.

VFS_stat( ) given a VFN, locates and returns information about VFS directories.

VFS_mkdir( ) given a VFN, creates the directory information in the mapping database.

VFS_rmdir( ) given a VFN, removes the directory information from the mapping database.

VFS_readdir( ) given a VFN in the passed in VFSDIR structure, on the first call to this function, locates the first directory entry and returns the structure with that directory information as well as the directory's first VFN/RFN and other information. On subsequent calls to this function given the same VFS directory information structure, returns the structure with the next VFN/RFN and other information

These functions will be used (sometimes in combination) to provide to clients (both terminals and controllers) all file system services.

The string 'dictionary' contains all strings that are part of a fully qualified path and file name, i.e. they are 'canonical'. For example:

"ADXLXZTN::VFS:\ThisIsALongDirectoryName\ThisIsALongFileName.ThisIsALongFileExtension."

When this string is parsed into its component parts it appears in the dictionary **10** as indicated in FIG. **1** where "n" is simply the index in the dictionary table where the strings are inserted. String dictionary **10** depicts an index field **12** and a corresponding string field **14**. The entries in the dictionary are not assumed to be sequential, but may be inserted in whatever order space is available. If a string already exists in the table, it is not inserted again, i.e., all strings in the table are unique in the dictionary. There is no need to delete entries from the dictionary as file and directory names may be used in many places and are often created and deleted repeatedly.

While the case of letters is not significant, it is preserved as whatever the file or directory name creation indicated. This allows an efficient hashing function algorithm. An easily implementable configurable option would be to consider case significant.

The implementation of this layer uses a string hashing scheme, the current version of which depends on a 4096 entry hash vector and a string summation and shifting hash function. This gives a very good distribution over the test sample which consists of 65,534 file and directory names from the HotJava browser, Jigsaw web server, and the Java Compatibility Kit (JCK) 1.1.6a. There are 35,369 unique strings in this test sample which average 11.5 characters each, producing a 410,191 byte dictionary. The hash function produces only 54 empty buckets and a maximum bucket size of 40 items. The simple arithmetic average of all non-empty buckets is 6 items and the weighted arithmetic mean is 13 items. An additional enhancement keeps the items in buckets sorted in a decreasing frequency of use order that also helps retrieval performance.

The string dictionary is used by the mapping database to keep very short entries that fully describe the unique, canonical path and file name associated with a particular file (virtual name), without having to keep the complete canonical path and file name in the mapping table.

The mapping database connects a logical name with a physical name. An entry in the mapping database consists of a fixed length structure that contains six 16-bit values that are the indices to other mapping table entries, RFN indicators, flags and indices into the string dictionary. Following the example provided for the string dictionary, an example for the mapping database **20** is illustrated in FIG. **2**. The mapping database **20** depicts a number of fields including mapping table entry number **22**, parent entry **24**, next entry **26**, xx|yy field **28** (described below), flags field **30** and name (index) field **32**. In FIG. **2**, "0, z, z+1 and z+2" are simply mapping table entry numbers and "n, n+1, n+3" are the indices (also referred to as tokens) into the dictionary table where the complete strings are kept and xx, yy are two 1-byte fields that represent the physical directory and file that the virtual file name is mapped to. If a table entry is a directory, then the xx, yy field is reused to indicate the mapping table entry that contains the first file in the directory. A free-entry (deleted) list and the number of entries in the table are maintained to avoid having to reorder table entries. This linked list is pointed to by the '.next' field in the root entry of the mapping database (i.e. the zero-th entry) and the number of entries in use is maintained in the '.parent' field of the same entry.

The virtual "device", in this case drive "VFS:", is added to the table and pointed to by the root entry. This enables expansion of this scheme to include other virtual drives.

The 'real' file name (RFN) allocation scheme is tied closely into the mapping database. It consists of a bit-map, logically 256×256 bits (i.e. 256×32 bytes, or 8192 bytes), where each bit represents a physical file. An example of the layout on the disk of the physical file allocation scheme is illustrated in FIG. **3**.

There are up to 256 sub-directories under the home (/vfs) directory, each with a 4 character name from '~~00' to '~~FF' (i.e., ASCII representation of hex 0×00 through 0xff), and up to 256 files in each sub-directory with names (similar to the sub-directories) from '00' through 'FF'.

As files are created, the bits representing the corresponding files are set to '1' and when the file is deleted, reset back to '0'. There is no mapping of virtual sub-directories to physical sub-directories, the physical sub-directories exist only to keep performance optimal. Virtual directories are only retained in the mapping database and have no physical counterpart.

As RFNs are allocated with this scheme, an extended attribute file with the same name, but with the added ".A" extension (e.g. "h0:vfs/~~ab/cd.A", where ab and cd are the ASCII representation of a two digit hexadecimal number). The full, canonical virtual pathname of the file is written into this extended attribute file to provide both a simple way of mapping from an RFN back to the associated VFN and a way of associating other extended attributes with this file (e.g. icons).

Each time a change is made in one of the in-memory database tables, a corresponding change is required to the version of that database table kept on the hard disk. This serves several purposes, the primary being the reliability of the VFS function in the operating system. If a machine is rebooted or a power loss occurs during a directory or file creation, deletion or rename, unforseeable errors can occur if the database was being changed and was temporarily invalid.

Therefore, all VFS directory and file name, create, delete and rename functions are processed in the following sequence:

1. a request is received for a create, delete, mkdir, rmdir or rename function;
2. the transaction log is opened and the request information is written into it;
3. the request starts being processed by the server; and
4. as a change is made to the in-memory database tables, the corresponding change to the database files is determined;
5. the file changes are written into the transaction log file;
6. after all changes have been made to the in-memory version and all changes to the file version have been determined and written to the transaction log, (1) the result and return code are written into the transaction log, and (2) the result and return code is sent back to the client;
7. the transaction log is processed, making all changes necessary to all database files and after all changes are complete; and
8. the transaction log is reset to zero length and closed.

If a power loss occurs at anytime during this process, either the transaction can still be cancelled without problems, or all information is securely written to the disk so the server can recover when power is restored.

This should allow reasonably stateless file updates to occur after server function is restored by simply writing all data in the transaction log to the correct files and therefore setting the database to a known state. This state would then agree with all clients who have received results back from calls to the server.

If a transaction log is incomplete (the final results and return code are not in the file), when server function is restored, the transaction log is truncated to zero length and processing continues.

The format in the transaction log of the VFS client requests are:

1. client node name (null terminated character array);
2. client VFS request (16-bits);
3. length of VFN #1 (16-bits: value is up to MAXVFNLEN bytes);
4. Virtual File Name #1 (null terminated character array);
5. length of VFN #2 (16-bits: value is up to MAXVFNLEN bytes);
6. Virtual File Name #2 (null terminated character array);

This format derives directly from the client request block used to transport the request from the client to the server.

The format in the transaction log of the database file updates are:

1. database file name manifest (32-bits);
2. file offset at which to write data (32-bits);
3. data block size (32-bits);
4. data block (variable length);

The format in the transaction log of the VFS server reply will be:

1. return code (16-bits);
2. length of returned RFN (if any. 16-bits: value is up to MAXRFNLEN);
3. Real File Name (null terminated character array);

The above format derives directly from the server reply block used to transport the reply from the server back to the client.

FIG. **4** illustrates a high level flowchart of the token replacement process of the present invention. The process starts in entry block **400** in which the current working directory and filename (e.g., current-work-dir=\dir1\dir2; name=filename) are input to the canonicalization process as indicated by logic block **402**. This action results in the canonical form such as pathname=\dir1\dir2\filename. This is followed in logic block **404** with parsing of the pathname and replacement of substrings with tokens. The substrings in this small example are "dir1", "dir2", and "filename". The result of this action are tokens t**1**, t**2**, and t**3**. The validation of the path is the next act in the process as indicated by logic block **406**. From this act the process continues in decision block **408** with a determination of the validity of the path. If the path is found to be invalid an error is returned as indicated by termination block **410**. Otherwise, the path is found to be valid and a file system operation is performed as indicated in logic block **412**.

FIG. **5** illustrates the specific acts of the canonicalization process **402** of FIG. **4**. It begins in decision block **500** with a determination if the name starts with a root substring. If it does, then processing jumps to logic block **508** for resolution of special characters in the name. If the name does not start with a root substring, then in logic block **502** the current working directory is copied to a work buffer. The content of the work buffer at this point in the process is \dir1\dir2. Next, the name (i.e., filename) is added to the work buffer as indicated in logic block **504**. The content of the work buffer at this point is \dir1\dir2\filename. In logic block **506**, the name is replaced with the work buffer contents. The process concludes in logic block **508** with the resolution of special characters such as ".." or ".". The canonicalization process exits back to the many processing logic in termination block **510**.

FIG. **6** illustrates a flowchart of the parsing process **404** of the present invention. It commences with the entry of decision block **600** which initiates an iterative routine to perform as long as the pathname contains substrings. The iterative routine begins in logic block **602** in which a substring is looked up in the string dictionary. If the substring does not exist then a new token is created to represent that substring. In logic block **604**, the token representing the substring is added to a list of output tokens for the pathname. The next act is to get the next substring from the pathname as indicated in logic block **606**. The iterative routine loops back to decision block **600**. After the entire pathname has been parsed into substrings and replaced with tokens (DONE indication out of decision block **600**), the parsing process retuns the tokens found as indicated in termination block **608**.

FIG. **7** illustrates a flowchart of the validation process **406** of the present invention. The token list is input to logic block **700** in which the current directory is set to the root directory. In logic block **702**, the directory table is accessed for the current directory. This is followed in logic block **704** with the act of getting a token from the token list. Next, in logic block **706**, a search is performed to locate the token in the directory table. In decision block **708**, a test is made to determine if the token was found in the directory table. If the search failed, then an invalid pathname indication is returned to the main processing logic via termination block **710**. If the search was successful, processing continues in decision block **712**, in which a test is made to determine if the token list is empty. If not, the processing continues in decision block **714** in which a determination is made as to whether or not the directory table entry found is for a file (rather than for a directory). If the directory table entry is for a directory, then processing continues in logic block **716** in which the current directory is set to the table entry data; processing then returns to logic block **702**. If the directory table entry found in decision block **714** is for a file, then processing ends in termination block **720** with an invalid pathname indication. If, in decision block **712**, the token list was found to be empty (i.e., all tokens have been processed) then processing exits in termination block **718** with the return of an valid pathname.

FIGS. **8A–8B** indicate both the prior art and the inventive method of storing directory and file names on a storage device, such as a disk. FIG. **8A** shows a linked list structure with dir1 stored in root block memory location **80**, dir2 stored in subdirectory block memory location **82**, the filename stored in subdirectory block memory location **84**, and the actual file stored at memory location **86**. FIG. **8B** indicates the method of storing directory and pathnames according to the present invention. Token t**1** is stored in root block memory location **90**, token t**2** is stored in subdirectory block memory location **92**, token t**3** is stored in subdirectory block memory location **94** which contains a pointer to the file stored at memory location **96**. Also shown in FIG. **8B** is the string dictionary **98** corresponding to this simple example.

A simple example of the use of the invention demonstrating its advantages is described below:

The filenames

String**1**=\test_**1**\Source\filename**1**.text

String**2**=\test_**1**\Source\filename**2**.text

String**3**=\test_**1**\Source\filename**1**.Output

String**4**=\test_**1**\Source\filename**2**.Output

String**5**=\test_**1**\Output\filename**1**.binary

String**6**=\test_**1**\Output\filename**2**.binary

contain 7 unique semantically significant substrings: "Test_**1** ", "Source", "filename**1**", "filename**2** ", "text", "output" and "binary".

If placed into a table (or dictionary) as illustrated in FIG. **9**, it is easy to see that a representation of the original substrings based on their position in the table would be (given the assumption that a "." is inserted in place of the "\" in front of the final token):

String**1**={t**1**, t**2**, t**3**, t**4** }

String**2**={t**1**, t**2**, t**5**, t**4** }

String**3**={t**1**, t**2**, t**3**, t**6** }

String**4**={t**1**, t**2**, t**5**, t**6** }

String**5**={t**1**, t**6**, t**3**, t**7** }

String**6**={t**1**, t**6**, t**5**, t**7** }

A simple comparison of the amount of storage to hold this information is as follows:

	Traditional method	New Method
String 1 =	6 + 6 + 9 + 4 = 25 bytes	8 bytes
String 2 =	6 + 6 + 9 + 4 = 25 bytes	8 bytes
String 3 =	6 + 6 + 9 + 6 = 27 bytes	8 bytes
String 4 =	6 + 6 + 9 + 6 = 27 bytes	8 bytes
String 5 =	6 + 6 + 9 + 6 = 27 bytes	8 bytes
String 6 =	6 + 6 + 9 + 6 = 27 bytes	8 bytes
	158 total bytes	48 total bytes

However, this greater than 3 to 1 comparison ratio is not quite entirely complete in that there is an "overhead" of 81 bytes to store the substrings in a dictionary (as null-terminated strings) along with the pointers to locate them. This overhead, while not negligible, is not as significant as the savings in replacing substrings with 2-byte numeric tokens.

The difference in speed of comparison is not quite so readily calculated. It is clear that comparing a new string:

StringN=\Test_**1**\Output\filename**2**.binary.NEW

with String**6**, character by character, would involve 32 comparisons of single bytes until a mismatch is found. A simple comparison of the two strings using the token-scheme would require four comparisons of 2-byte tokens.

Again, this 8 to 1 ratio is not entirely complete in that the conversion of the strings into substrings and proper insertion into the table require some overhead, but in a file system where locating information is much more frequent than inserting, removing or renaming it, this overhead is not as significant as the savings in numeric comparisons verus string comparisons.

A third advantage that is usually involved whenever data compression is present is the additional security for a file system that uses the new method. Several schemes could be easily applied to prevent the string dictionary from being accessed even though the file and directory names may be available. This is the "shared-secret" type of security and is the most difficult to decrypt. While the substrings themselves can also be encrypted, it would be easier to take advantage of the clean split between the semantic information embodied in the tokens and the human-readable form of the strings to deter someone from locating secure information in a file system.

The fourth advantage is that of the additional flexibility that tokenizing the substrings provides. Since the actual substrings are stored in a separate place from the directory and file information in the native file system, limits on the length of a substring, overall length of a path (composed of many substrings) as well as the permissible characters in any substring can be much different than those imposed by the native file system. As long as the sequence of tokens can be uniquely mapped to a native file system resource practically any string can be accommodated. The tokens are used only to uniquely represent the substrings, wherever they may be used in a file system name. A clear example is the above use of "Output" as both a sub-directory name and as a file "extension" in String**3** and String**5** for instance.

The file/directory pathnames token replacement mechanism of the present invention has been described as a computer program that can be resident on one or more host computers such as a workstation, a network device, or a server device. As such, the token replacement mechanism can stored as an application on any network device. It is important to note, however, that those skilled in the art will appreciate that the mechanisms of the present invention are capable of being distributed as a program product in a variety of forms, and that the present invention applies equally regardless of the particular type of signal bearing media utilized to actually carry out the distribution. Examples of signal bearing media include, without limitation, recordable type media such as cassettes or CD ROMS and transmission type media such as analog or digital communication links.

Additionally, corresponding structures, materials, acts, and equivalents of all means plus function elements in the claims below are intended to include any structure, material, or acts for performing the functions in combination with other claimed elements as specifically claimed.

While the invention has been particularly shown and described with reference to a preferred embodiment thereof, it will be understood by those skilled in the art that various changes in form and detail may be made without departing from the spirit and scope of the present invention.

What is claimed is:

1. A method for replacing substrings in file and directory pathnames with tokens in a computer-implemented file system, comprising the acts of:

reading a name string to be converted into a list of tokens;

canonicalizing a current working directory and the name string to form a pathname containing a plurality of substrings;

parsing the pathname and replacing each substring with an associated token; and

validating the parsed pathname containing the list of tokens.

2. The method for replacing substrings in file and directory pathnames with tokens of claim **1** wherein the act of canonicalizing includes the acts of:

determining if the read name string starts with a root substring;

if the act of determining indicates that the read name string does not begin with a root substring, performing the additional acts of:

copying the current working directory to a working buffer;

adding the name string to the working buffer; and

replacing the read name string with the contents of the working buffer.

3. The method for replacing substrings in file and directory pathnames with tokens of claim **2** wherein the act of canonicalizing further includes the act of resolving any special characters contained in the name string.

4. The method for replacing substrings in file and directory pathnames with tokens of claim **1** wherein the act of parsing the pathname includes the acts of:

dissecting the pathname into a plurality of substrings;

for each substring in the pathname, performing the additional acts of:

searching for the substring in a string dictionary; and

adding a token corresponding to the substring to the list of output tokens representing the pathname; and

returning the list of tokens for further processing in the act of validating the parsed pathname.

5. The method for replacing substrings in file and directory pathnames with tokens of claim **4** wherein the act of parsing the pathname further includes the act of creating a new token for any substring that is not found in the search of the string dictionary.

6. The method for replacing substrings in file and directory pathnames with tokens of claim **1** wherein the act of validating includes the acts of:

setting the current directory to a root directory;

accessing a directory table to locate the current directory;

getting a token from the list of output tokens;

searching the directory table for the output token; and

while the list of output tokens is not empty and the table entry data indicates that the token does not correspond to a file name, setting the current directory to the table entry data and repeating the acts of accessing, getting and searching.

7. The method for replacing sub strings in file and directory pathnames with tokens of claim **6** wherein the act of validating further includes the act of returning an invalid pathname if the token is not found in the directory table, or the list of output tokens is not empty and the table entry data indicates that the token corresponds to a file name.

8. The method for replacing substrings in file and directory pathnames with tokens of claim **6** wherein the act of validating further includes the act of returning a pathname valid indication after each token from the list of output tokens has been found in the directory table.

9. A method for enhancing performance related to a selected file system operation in a computer-implemented file system, comprising the acts of:

reading a name string on which to conduct a file system operation;

canonicalizing a current working directory and the name string to form a pathname containing a plurality of substrings;

parsing the pathname and replacing each substring with an associated token;

validating the parsed pathname containing a list of tokens; and

performing the selected file system operation on the parsed pathname.

10. The method for enhancing performance related to a selected file system operation of claim **9** wherein the act of canonicalizing includes the acts of:

determining if the read name string starts with a root substring;

if the act of determining indicates that the read name string does not begin with a root substring, performing the additional acts of:

copying the current working directory to a working buffer;

adding the name string to the working buffer; and

replacing the read name string with the contents of the working buffer.

11. The method for enhancing performance related to a selected file system operation of claim **10** wherein the act of canonicalizing further includes the act of resolving any special characters contained in the name string.

12. The method for enhancing performance related to a selected file system operation of claim **9** wherein the act of parsing the pathname includes the acts of:

dissecting the pathname into a plurality of substrings;

for each substring in the pathname, performing the additional acts of:

searching for the substring in a string dictionary; and

adding a token corresponding to the substring to a list of output tokens representing the pathname; and

returning the list of tokens for further processing in the act of validating the parsed pathname.

13. The method for enhancing performance related to a selected file system operation of claim **12** wherein the act of parsing the pathname further includes the act of creating a new token for any substring that is not found in the search of the string dictionary.

14. The method for enhancing performance related to a selected file system operation of claim **9** wherein the act of validating includes the acts of:

setting the current directory to a root directory;

accessing a directory table to locate the current directory;

getting a token from a list of output tokens;

searching the directory table for the output token; and

while the list of output tokens is not empty and the table entry data indicates that the token does not correspond to a file name, setting the current directory to the table entry data and repeating the acts of accessing, getting and searching.

15. The method for enhancing performance related to a selected file system operation of claim **14** wherein the act of validating further includes the act of returning an invalid pathname if the token is not found in the directory table, or the list of output tokens is not empty and the table entry data indicates that the token corresponds to a file name.

16. The method for enhancing performance related to a selected file system operation of claim **14** wherein the act of validating further includes the act of returning a pathname valid indication after each token from the list of output tokens has been found in the directory table.

17. A computer readable medium containing a computer program product for replacing substrings in file and directory pathnames with tokens in a computer-implemented file system, comprising:

program instructions that read a name string to be converted into a list of tokens;

program instructions that canonicalize a current working directory and the name string to form a pathname containing a plurality of substrings;

program instructions that parse the pathname and replace each substring with an associated token; and

program instructions that validate the parsed pathname containing the list of tokens.

18. The computer program product of claim **17** wherein the program instructions that canonicalize include:

program instructions that determine if the name string read starts with a root substring;

program instructions that copy the current working directory to a working buffer;

program instructions that add the name string to the working buffer; and

program instructions that replace the read name string with the contents of the working buffer.

19. The computer program product of claim **18** wherein the program instructions that canonicalize further include program instructions that resolve any special characters contained in the name string.

20. The computer program product of claim **17** wherein the program instructions that parse the pathname include:

program instructions that dissect the pathname into a plurality of substrings;

program instructions that search for each substring in a string dictionary;

program instructions that add a token corresponding to the substring to the list of output tokens representing the pathname; and

program instructions that return the list of tokens for further processing by the program instructions that validate the parsed pathname.

21. The computer program product of claim **20** wherein the act of parsing the pathname further include program instructions that create a new token for any substring that is not found in the search of the string dictionary.

22. The computer program product of claim **17** wherein the program instructions that validate include:

program instructions that set the current directory to a root directory;

program instructions that access a directory table to locate the current directory;

program instructions that get a token from the list of output tokens;

program instructions that search the directory table for the output token; and

while the list of output tokens is not empty and the table entry data indicates that the token does not correspond to a file name, program instructions that set the current directory to the table entry data and repeat the program instructions that access, get and search.

23. The computer program product of claim **22** wherein the program instructions that validate further include program instructions that returns an invalid pathname if the token is not found in the directory table, or the list of output tokens is not empty and the table entry data indicates that the token corresponds to a file name.

24. The computer program product of claim **22** wherein the program instructions that validate further include the program instructions that return a pathname valid indication after each token from the list of output tokens has been found in the directory table.

25. A computer readable medium containing a computer program product for enhancing performance related to a selected file system operation in a computer-implemented file system, comprising:

program instructions that read a name string on which to conduct a file system operation;

program instructions that canonicalize a current working directory and the name string to form a pathname containing a plurality of substrings;

program instructions that parse the pathname and replace each substring with an associated token;

program instructions that validate the parsed pathname containing a list of tokens; and

program instructions that perform the selected file system operation on the parsed pathname.

26. The computer program product of claim **25** wherein the program instructions that canonicalize include:

program instructions that determine if the name string read starts with a root substring;

program instructions that copy the current working directory to a working buffer;

program instructions that add the name string to the working buffer; and

program instructions that replace the read name string with the contents of the working buffer.

27. The computer program product of claim **26** wherein the program instructions that canonicalize further include program instructions that resolve any special characters contained in the name string.

28. The computer program product of claim **25** wherein the program instructions that parse the pathname includes:

program instructions that dissect the pathname into a plurality of substrings;

program instructions that search for each substring in a string dictionary;

program instructions that add a token corresponding to the substring to a list of output tokens representing the pathname; and

program instructions that return the list of tokens for further processing by the program instructions that validate the parsed pathname.

29. The computer program product of claim **28** wherein the program instructions that parse the pathname further include program instructions that creates a new token for any substring that is not found in the search of the string dictionary.

30. The computer program product of claim **25** wherein the program instructions that validate include:

program instructions that set the current directory to a root directory;

program instructions that access a directory table to locate the current directory;

program instructions that get a token from a list of output tokens;

program instructions that search the directory table for the output token; and

while the list of output tokens is not empty and the table entry data indicates that the token does not correspond to a file name, program instructions that set the current directory to the table entry data and repeat the program instructions that access, get and search.

31. The computer program product of claim **25** wherein the program instructions that validate further include program instructions that returns an invalid pathname if the token is not found in the directory table, or the list of output tokens is not empty and the table entry data indicates that the token corresponds to a file name.

32. The computer program product of claim **25** wherein the program instructions that validate further include program instructions that returns a pathname valid indication

after each token from the list of output tokens has been found in the directory table.

33. A system for replacing substrings in file and directory pathnames with tokens in a computer-implemented file system, comprising:

an input module that reads a name string to be converted into a list of tokens;

a module that canonicalizes a current working directory and the name string to form a pathname containing a plurality of substrings;

a module that parses the pathname and replaces each substring with an associated token; and

a module that validates the parsed pathname containing the list of tokens.

34. The system for replacing substrings in file and directory pathnames with tokens of claim **33** wherein the module that canonicalizes includes:

a module that determines if the name string read starts with a root substring;

a module that copies the current working directory to a working buffer;

a module that adds the name string to the working buffer; and

a module that replaces the read name string with the contents of the working buffer.

35. The system for replacing substrings in file and directory pathnames with tokens of claim **34** wherein the module that canonicalizes further includes a module that resolves any special characters contained in the name string.

36. The system for replacing substrings in file and directory pathnames with tokens of claim **33** wherein the module that parses the pathname includes:

a module that dissects the pathname into a plurality of substrings;

a module that searches for the substring in a string dictionary;

a module that adds a token corresponding to the substring to the list of output tokens representing the pathname; and

a module that returns the list of tokens for further processing in the module that validates the parsed pathname.

37. The system for replacing substrings in file and directory pathnames with tokens of claim **36** wherein the module that parses the pathname further includes a module that creates a new token for any substring that is not found in the search of the string dictionary.

38. The system for replacing substrings in file and directory pathnames with tokens of claim **33** wherein the module that validates includes:

a module that sets the current directory to a root directory;

a module that accesses a directory table to locate the current directory;

a module that gets a token from the list of output tokens;

a module that searches the directory table for the output token; and

while the list of output tokens is not empty and the table entry data indicates that the token does not correspond to a file name, a module that sets the current directory to the table entry data and causes a return to the module that accesses.

39. The system for replacing substrings in file and directory pathnames with tokens of claim **38** wherein the module that validates further includes a module that returns an invalid pathname if the token is not found in the directory table, or the list of output tokens is not empty and the table entry data indicates that the token corresponds to a file name.

40. The system for replacing substrings in file and directory pathnames with tokens of claim **38** wherein the module that validates further includes a module that returns a pathname valid indication after each token from the list of output tokens has been found in the directory table.

41. A system for enhancing performance related to a selected file system operation in a computer-implemented file system, comprising:

a module that reads a name string on which to conduct a file system operation;

a module that canonicalizes a current working directory and the name string to form a pathname containing a plurality of substrings;

a module that parses the pathname and replaces each substring with an associated token;

a module that validates the parsed pathname containing a list of tokens; and

a module that performs the selected file system operation on the parsed pathname.

42. The system for enhancing performance related to a selected file system operation of claim **41** wherein the module that canonicalizes includes:

a module that determines if the entered name string starts with a root substring;

a module that copies the current working directory to a working buffer;

a module that adds the name string to the working buffer; and

a module that replaces the entered name string with the contents of the working buffer.

43. The system for enhancing performance related to a selected file system operation of claim **42** wherein the module that canonicalizes further includes a module that resolves any special characters contained in the name string.

44. The system for enhancing performance related to a selected file system operation of claim **41** wherein the module that parses the pathname includes:

a module that dissects the pathname into a plurality of substrings;

a module that searches for each substring in a string dictionary;

a module that adds a token corresponding to the substring to a list of output tokens representing the pathname; and

a module that returns the list of tokens for further processing in the module that validates the parsed pathname.

45. The system for enhancing performance related to a selected file system operation of claim **44** wherein the module that parses the pathname further includes a module that creates a new token for any substring that is not found in the search of the string dictionary.

46. The system for enhancing performance related to a selected file system operation of claim **41** wherein the module that validates includes:

a module that sets the current directory to a root directory;

a module that accesses a directory table to locate the current directory;

a module that gets a token from a list of output tokens;

a module that searches the directory table for the output token; and

while the list of output tokens is not empty and the table entry data indicates that the token does not correspond to a file name, a module that sets the current directory to the table entry data and causes a return to the module that accesses.

47. The system for enhancing performance related to a selected file system operation of claim **46** wherein the module that validates further includes a module that returns an invalid pathname if the token is not found in the directory table, or the list of output tokens is not empty and the table entry data indicates that the token corresponds to a file name.

48. The system for enhancing performance related to a selected file system operation of claim **46** wherein the module that validates further includes a module that returns a pathname valid indication after each token from the list of output tokens has been found in the directory table.

* * * * *