Distributed Computing for Real-time Video Processing

Hiroshi Watanabe, Abhay Ghatpande and Hidenori Nakazato
Graduate School of Global Information and Communications Study, Waseda University
29-7 Bldg., 1-3-10 Nishi-Waseda, Shinjuku-ku, Tokyo 169-0051, Japan
E-mail: hiroshi@giti.waseda.ac.jp

1. Introduction

Real-time video processing, such as rendering and/or encoding, is one of the most bandwidth demanding operation in audio-visual applications [1][2]. There have been many efforts at parallelizing video rendering and encoding architecture mainly for VLSI chip design [3][4][5][6][7][8][9][10][11][12][13][14][15][16]. However, not a lot of research has been dedicated in the area of distributed computing for real-time video processing [1][2][17][18][19][20]. Distributed computing on the Internet has mainly been applied to applications in which data could be processed in non-real-time [10]. For visual communication, real-time constraints give additional requirements to data processing in distributed computing. It is necessary to assure the processing time of distributed data since processing period for one frame of video is limited to 1/25 or 1/30 second in most cases. Thus, processing delay is a critical factor for video processing applications especially in the case of non-homogeneous computing environment, such as distributed computing on the Internet. In this paper, we first survey the trend of distributed computing for video processing, followed by introducing some architecture for distributed computing of real-time video application, particularly video encoding.

2. Video Encoding and Distributed Computing

Distributed computing, which requires universal access to high-grade computation facilities, is yet to be achieved. Average users still suffer from a chronic lack of bandwidth and processing power for demanding applications. Computation complexity and bandwidth necessity make video encoding difficult. Thus, parallel and distributed architectures for video encoding have been the subject of research for these ten years. Most successful attempts have unfortunately remained in the dominion of those with high performance computers connected by high-speed networks. Commercial grade video encoding and new, high-quality encoder/decoder are not available to average users. Architecture to distribute and encode video on the Internet would benefit users immensely. Though such a system would be non-real-time for some time to come [10], it would enable Internet users to encode and share high quality videos.

It is beneficial to realize state of the art video coding, such as MPEG-4 and AVC (H.264), by distributed computing architecture. It can also be used for the conventional MPEG-2 and MPEG-1 standards. Another application envisaged is the encoding of HDTV and digital cinema, etc. The purpose is to empower the user community to be able to encode and share high quality video without the associated high cost.

Audio and video exchange continues to dominate the traffic on P2P (Peer to Peer) networks today. Media capture, streaming, download, voice and video chat are important applications for the average Internet user. Storage capacity of magnetic hard disks has increased exponentially over the past few years. CPU processing speed has been improved substantially with special instructions for audio and video processing. The reality, however, is still that users cannot generate high-quality video on their own, primarily because video encoding has very high computation requirements. Though very good video encoder/decoder abound, normally users cannot easily access to them. Thus, users are stuck with grainy videos captured with low-resolution cameras while commercially excellent HDTV resolution video is available.

If there is a “video encoding grid” deployed over the intranet of the company or over the Internet, users can submit the job to the grid, along with some kind of encoding parameters. The video is distributed over the processors available and the encoded video is returned either to the originator or to any other specified machine where it is assembled in to syntactically correct bit stream.

There are very few distributed computing projects that focus on video encoding or processing so far. On the other hand, there have been many researches on parallel processing architecture for designing video encoding chips. Though internet distributed video encoding differs significantly from encoding using a bank of parallel processors, a review of parallel encoding techniques will give us some insight into the issues involved. Currently, bandwidth limitations will force such a video-processing grid to operate in non-real-time. It can be expected that as bandwidth issues diminish in the future, real-time operations will become feasible.

There have been many more efforts at parallelizing video encoding than can be mentioned here. If we disregard massively parallel implementations using an array/grid of tightly integrated microprocessors or specially designed hardware [3] because they fall outside our purview of interest, then [7][10][11][12][13][14] provide a good sample of the efforts in this direction.

The problem of software-based video encoding using parallel processing is non-trivial, and cannot be solved by simply replicating multiple sequential encoders on different processors. The local memory of a single processor is usually not large enough to hold more than a few frames and thus an efficient I/O methodology is required to bring the data in and to take the compressed data out of the processors. Since the video signal can be viewed as a 3D signal, various partitioning schemes are possible. The parallelism can be exploited at several levels of image (Fig. 1), especially based on a size of image data and a block of some images, such as macroblock (a unit of motion compensated interframe prediction), slice (a group of macroblocks in the horizontal direction), frame, and/or "Group of Pictures" (GOP) levels.

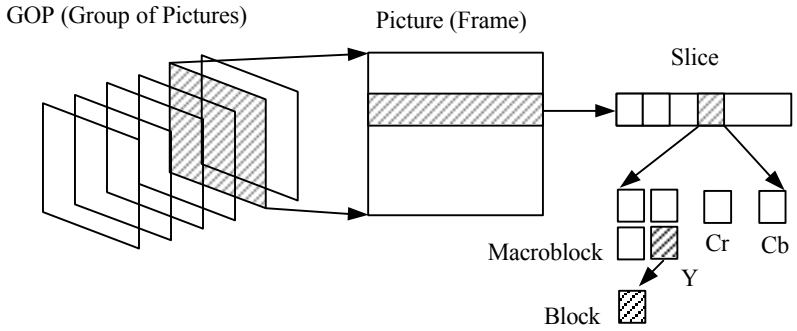

Fig.1 Levels of image based on the data size.

A natural solution to on-line encoding is to partition a frame as it arrives among many processors [7][11]. All processors then concurrently encode their parts of the frame data. The degree of parallelism can be increased by making the granularity as small as possible in terms of the processing data size and the number of available processors.

This is non-trivial though since the computations of individual processors are not independent and the processors need to communicate with each other to exchange certain parameters and data. The encoding times are not the same for all the partitions and hence the global synchronization can incur waiting times at some processors. Additionally, the overhead of data distribution and concatenation of results can saturate the speedup if the number of processors is increased. The drawback of such an approach, in addition to massive parallelism, is that the speedup may saturate since a macroblock is the smallest unit of data that can be reasonably assigned to one processor. Decreasing the granularity beyond a macroblock would incur very heavy inter-processor communication due to motion estimation that requires data beyond the local block [10]. When video data is available on some storage disk, partitioning in the temporal direction may be advantageous and let each processor or a group of processors encode a sequence of frames such as GOP [13].

Many of the schemes outlined above have reported very good results with encoding speedup. Frame rates

of over 41 frames per second for CIF (Common Intermediate Format, 352 x 288 pixels) sequence have been reported [19] and 33 frames per second for ITU-R 601 (Conventional television resolution, 720 x 480 pixels) [20]. The results above would be very encouraging if were not for the fact that they have been carried out on machines such as the Intel Paragon and Touchstone Delta with 100 to 512 processors. These machines are supercomputers with massive amounts of shared memory, data bandwidth, reserved networks, and high performance CPUs. The end result is that the average user is again left without any gain out of these successful experiments.

3. Real-time Grid Design

It would be beneficial to design an architecture, which enables high quality video encoding over open networks like the Internet. This approach would enable common users to make use of high quality video encoder/decoder and encode high-resolution videos irrespective of the bandwidth constraints. To design such architecture, it is necessary to take a look at distributed computing in general and grid computing in particular [17][21][22][23][24][25][26].

It is envisioned to enable communities ("virtual organizations") to share geographically distributed resources as they pursue common goals [27][28]. There is a distinctive absence of:

- Central location,
- Central control,
- Omniscience,
- Existing trust relationships.

The idea is to be able to share resources at will - computers, storage, sensors, networks, etc. This takes the concept beyond standard client-server with distributed data analysis, computation, and collaboration aided by the creation of large or small, static or dynamic, multi-institutional virtual organizations.

It can be easily seen that there are several differences with respect to parallel systems:

- Dynamic system of resources,
- Large heterogeneous and diverse systems,
- Sharing of resources,
- Transparent resource allocation,
- Unreliable network connectivity, link capacity, delay.

What this translates into for distributed video encoding is that the number of processors (machines) available may vary not only from session to session but also during the session itself. Here, one session is the complete encoding of one video sequence. Second, the processors may have varying characteristics, processing capabilities, and instruction sets. Third, there are no guarantees regarding the time that will be taken to complete a job, or whether it will be completed at all. Fourth, the links between the processors may each have different characteristics and capacities.

All these factors may give an impression of utter chaos and complete uncertainty. However, there are several toolkits such as Globus [29], Legion [30], etc. that act as middleware to take care of most of these problems. With this layer of abstraction in place, we can concentrate on the higher-level algorithms for distribution and encoding of the video frames. Considering the special requirements for computation over grids, we can design architecture quite different than what is used for simple parallel encoding on a cluster of networked PCs [17].

4. Research Topics

Input/output (I/O) in parallel computer systems can be a bottleneck in a number of parallel applications. Removal of the I/O bottleneck requires an integrated approach, which addresses the problem at all the levels in the system, including the storage and parallel architecture [31]. In the context of video encoding, the objective is to accomplish the optimal encoding rate, which can be achieved if all the processors are kept busy. This means that their waiting times are zero. This requires a careful I/O strategy that is highly pipelined and always provides data whenever a processor finishes encoding of its previously assigned

data. This, in turn, requires a data layout scheme that can minimize all of the overhead and the yield the desired I/O rate [10]. In a distributed system implemented using cycle stealing mechanism for example, the duration for which each processor is available and the processing capability of the CPU (memory, peripherals, instruction set, etc) may be vastly different. If there is at least some semblance of synchronization to be maintained, it is necessary that the encoding load should be distributed according to the capability of the processor. This means that the video frame may have to be split unevenly, i.e. not fixed number of macroblocks as in parallel system. Other factors such as object-based coding outlined below may also benefit from this non-uniform split of the frame data [32][33][34][35]. We need to implement a strategy to divide the frame data that will take into account all these factors.

Experimental results have proven that motion estimation/compensation is the most computation intensive and consequently most time consuming process of video encoding. Sometimes during parallelization the overhead of data distribution and collection exceeds that of motion estimation/compensation [7][13][14][36]. These two issues are closely related. We need to incorporate an efficient motion compensation strategy into an architecture that will provide high quality prediction and at the same time minimize data to be exchanged between source and each processor, and amongst the processors. Many strategies for parallel motion compensation [37][38][39] have been successful for the static configurations of those systems. However, the same strategies will not work well for a dynamic and changing environment of distributed computing.

Another important factor to be considered is that most of these strategies were used for MPEG-2 or older video coding standards. MPEG-4 and AVC (H.264) for example provide the facility of coding arbitrary shapes (object-based coding) and these algorithms do not provide solutions for this possibility. Any future algorithm should incorporate the facility for motion compensation of arbitrary and changing shapes.

Another issue of prime importance in a video encoder, irrespective of whether it is distributed or not, is the output bitrate control. Accurate bitrate control is not an easy task in distributed computing environment. A multi-level rate control scheme is proposed in [40] which enables reasonably good rate control accuracy in a parallel implementation. However, issues similar to those outlined above for motion compensation are applicable to rate control too. The frame may not be divided into equal parts for distribution. In that case, it would be difficult to estimate the bit allocation for each part. The other point is that we wish to support arbitrary shape encoding means that we have to use bitrate control algorithms for object-based coding.

In short, some of the issues to be addressed can be stated as follows:

- How to split the frame into regions for distribution?
- How to monitor the processors available, their processing capability?
- How to monitor link connectivity, performance?
- Strategy to allocate the regions to respective processors for encoding?
- Strategy to limit the search area for motion compensation?
- How to finish the entire job the fastest?
- How to minimize processing time and idle time of processors?
- How to optimize the utilization of available resources?
- Reallocation/Recovery strategy in case of processor/link failure?
- Strategy for bitrate control?
- Strategy to support arbitrary shape encoding? (subset of frame splitting)

All items listed in this section have not yet been solved. One possible solution is to start from sketching from high-level functions of a system, which consists of mainly two subsystems - the video pre-processor, and the encoding subsystem. An example of such system is shown in Fig. 2.

The pre-processor performs two important functions:

- Identifies regions with similar feature characteristics (color, texture, motion)
- Provides approximate motion data of these regions

This means that the output of the pre-processors is a list of different sized regions in the frame that exhibit

similar characteristics and can be encoded together. We can consider the image to be "tagged" at block or macroblock level and each tag identifies the region to which the block belongs, the approximate motion vector for the block, and the characteristic that identifies that block. A lot of details have been glossed over in this sketch. It presents just an idea for segmenting the video frame into manageable and correlated pieces. Though difficult, this sketch could be broken down by some algorithms and implemented with reasonable effort.

5. Conclusion

In this paper, we introduced some architecture for distributed computing of real-time video encoding after surveying the trend of distributed computing for image processing. Processing delay is a critical factor for video processing applications especially in the case of non-homogeneous computing environment, such as distributed computing on the Internet. Recent researches in this area have been lead by Prof. M. L. Liu of Hong Kong University of Science and Technology, Prof. R. Taniguchi of Kyushu University, and Prof. E. J. Delp of Purdue University. These researchers have been tackling a lot of difficult problems. However, there are still many issues to be solved if we wish to realize distributed computing for state of the art object-based video encoding on the Internet.

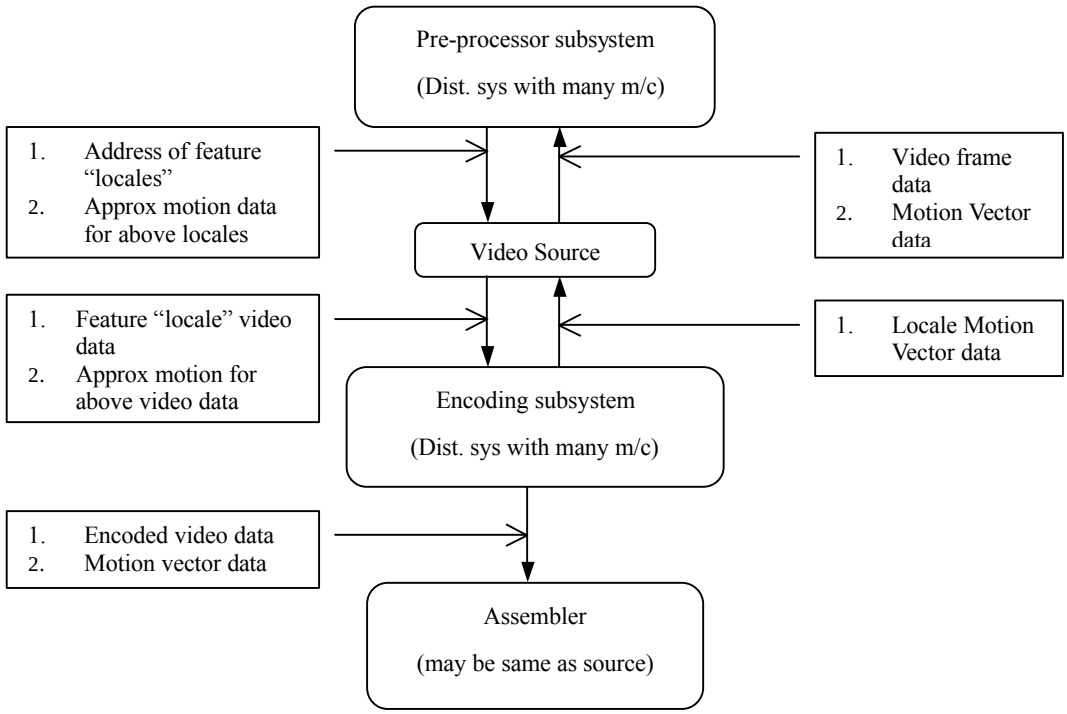

Fig. 2 High-level functions of distributed video encoding system.

References

[1] T. D. Nguyen, C. Peery and J. Zahorjan: "DDDDRRaW: A prototype toolkit for distributed real-time rendering on commodity clusters," In Proc. of International Parallel and Distributed Processing Symposium (IPDPS'01), 2001.
[2] H. Yoshimoto, D. Arita and R. Taniguchi: "Real-time communication for distributed vision processing based on imprecise computation model," In Proc. of International Parallel and Distributed Processing Symposium (IPDPS'02), 2002
[3] K. Shen, G. Cook, L. Jamieson, and E. J. Delp: "An overview of parallel processing approaches to

image and video compression"; In Proc. of SPIE'94 on Image and Video Compression, Vol.2186, pp.197-208, Feb. 1994.
[4] R. Taniguchi, Y. Makiyama, N. Tsuruta and S. Yonemoto and D. Arita: "Software platform for parallel image processing and computer vision," In Proc. of SPIE'97 on Parallel and Distributed Methods for Image Processing, Vol.3166, pp.2-10, Sep. 1997.
[5] I. Ahmad, Y. He and M. L. Liou: "Video compression with parallel processing," Parallel Computing, Volume 28, Issues 7-8, pp.1039-1078, August 2002.
[6] A. Bonhomme, and L. Prylli: "Performance evaluation of a distributed video storage system," In Proc. of International Parallel and Distributed Processing Symposium (IPDPS'02), 2002.
[7] S. M. Akramullah, I. Ahmad and M. L. Liou: "Parallelization of MPEG-2 video encoder for parallel and distributed computing systems'", In Proc. of 38th Midwest Symposium on Circuits and Systems, Vol. 2, pp. 834-837, Rio de Janeiro, Brazil, Aug. 1995.
[8]] Y. He, I. Ahmad and M. L. Liou: "A software-based MPEG-4 video encoder using parallel processing," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 8, No. 7, pages 909-920, Sep. 1998.
[9] Y. Kwok, K. Karlapalem, I. Ahmad and N. M. Pun: "Data and evaluation of data allocation algorithms for distributed multimedia database systems," IEEE Journal on Selected Areas in Communication, Vol. 14, No. 7, pp. 506-521, Sep. 1996.
[10] I. Ahmad, S. Akramullah, M. L. Liou and M. Kafil: "A Scalable off-line MPEG-2 video encoding scheme using a multiprocessor system," Parallel Computing, Vol. 27, No. 6, pp. 823-846, May 2001.
[11] T. Olivares, P. Cuenca, F. Quiles, and A. Garrido; "Parallelization of the MPEG coding algorithm over a multicomputer. A proposal to evaluate its interconnection network," IEEE Pacific Rim Conference on Communications, Computer and Signal Processing (PACRIM'97). 1997
[12] T. Olivares, P. Cuenca, F. Quiles, L. Orozco-Barbosa, and I. Ahmad; "Study of Data Distribution Techniques for the Implementation of an MPEG-2 Video Encoder"; In Proc. of Parallell and Distributed Computing Systems'99, pp. 537-542, Nov. 1999.
[13] D. Barbosa, J. P. Kitajima and W. Meira Jr.: "Parallelizing MPEG video encoding using multiprocessors," XII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI '99), pp.215-222, Oct. 1999.
[14] D. Farin, N. Mache and P. de With: "SAMPEG, a scene adaptive parallel MPEG-2 software encoder," in Proc. of SPIE'01 on Visual Communications and Image Processing (VCIP'01), Vol. 4310, pp. 272-283, Jan 2001.
[15] S. Bozoki, S. J. P. Westen, R. L. Lagendijk and J. Biemond: "Parallel algorithms for MPEG video compression with PVM," In Proc. of EuroSIM'96, pp.315-326, June 1996.
[16] S. M. Akramullah, I. Ahmad and M. L. Liou: "Performance of a software-based MPEG-2 video encoder on parallel and distributed systems," IEEE Transactions on Circuits and Systems for Video Technology, Vol.7, No.4, pp. 687-695, Aug. 1997.
[17] H. Yoshimoto, D. Arita, and R. Taniguchi: "Real-time image processing on IEEE1394-based PC cluster," In Proc. of International Parallel and Distributed Processing Symposium (IPDPS'03), 2003.
[18] S. M. Akramullah, I. Ahmad and M. L. Liou: "A Data-Parallel Approach for Real-Time MPEG-2 Video Encoding," Journal of Parallel and Distributed Computing, Vol. 30, No. 2, pp. 129-146, November 1995.
[19] K. Shen, L. A. Rowe, and E. J. Delp: "A parallel implementation of an MPEG-1 encoder: Faster than real-time"; In Proc. of SPIE'95 on Digital Video Compression: Algorithms and Technologies, Vol.2419, pp. 407-418, Feb. 1995.
[20] K. Shen, and E. J. Delp: "A Spatio-Temporal parallel approach for real-time MPEG video compression," In Proc. of IEEE International Conference on Parallel Processing (ICPP'96), Vol. 2, pp.100-107, Aug. 1996.
[21] Michael Gerndt: "Grid Computing Challenges and Techniques," International Workshop on Parallel Numerics (PARNUM 2002), Oct. 2002.
[22] G. Allen, T. Goodale, M. Russell, E. Seidel and J. Shalf: "Classifying and Enabling Grid Applications"; Concurrency—Practice and Experience, 2000.
[23] D. Kranzlmuller, G. Kurka, P. Heinzlreiter and J. Volkert: "Optimizations in the grid visualization kernel," In Proc. of International Parallel and Distributed Processing Symposium (IPDPS'02), 2002.

[24] Y. M. Teo, S. C. Low, S. C. Tay and J. P. Gozali: “Distributed geo-rectification of satellite images using grid computing,” In Proc. of International Parallel and Distributed Processing Symposium (IPDPS’03), 2003.
[25] I. Taylor, M. Shields, I. Wang, and R. Philip: “Distributed P2P computing within Triana: A galaxy visualization test case,” In Proc. of International Parallel and Distributed Processing Symposium (IPDPS’03), 2003.
[26] S. Basu, S. Adhikari, R. Kumar, Y. Yan, R. Hochmuth and B. E. Blaho: “mmGrid: Distributed resource management infrastructure for multimedia applications”, In Proc. of International Parallel and Distributed Processing Symposium (IPDPS’03), 2003.
[27] I. Foster, C. Kesselman, and S. Tuecke; “The anatomy of the grid - enabling scalable virtual organizations,” International Journal of High Performance Computing Applications, Vol.15, No.3, pp. 200-222, Fall 2001.
[28] M. Romberg: “The unicore architecture: Seamless access to distributed resources,” In Proc. of ACM SIGGRAPH’95, 1995.
[29] I. Foster and C. Kesselman: “Globus: A metacomputing infrastructure toolkit,” International Journal of Supercomputer Applications, Vol. 11, No. 2, pp. 15-128, Summer 1997.
[30] A. Grimshaw, A. Ferrari, F. Knabe, and M. Humphrey; “Wide-area computing: Resource sharing on a large scale,” IEEE Computer, Vol. 32, No. 5, pp.29-37, May 1999.
[31] R. Jain, K. Somalwar, J. Werth and J. C. Browne: “Heuristics for scheduling I/O operations,” IEEE Trans. on Parallel and Distributed Systems, No. 8, Vol. 3, pp. 310-320, Mar 1997.
[32] T. Meier and K. Ngan: “Video Segmentation for Content-Based Coding,” IEEE Trans. on Circuits and Systems for Video Technology, Vol. 9, No. 8, pp. 1190–1203, Dec. 1999.
[33] M. Kim, J. G. Choi, D. Kim, H. Lee, M. H. Lee, C. Ahn and Y. S. Ho: “A VOP generation tool: Automatic segmentation of moving objects in image sequences based on spatio-temporal information,” IEEE Trans. on Circuits and Systems for Video Technology, Vol. 9, No. 8, pp. 1216-1226, Dec. 1999.
[34] C. T. Hsu and Y. C. Tsan: “Mosaics of video sequences with moving objects,” in Proc. of IEEE International Conference on Image Processing (ICIP’01), pp387-390, Oct. 2001.
[35] T. Kruger, J. Wickel and K. F. Kraiss: “Parallel and distributed computing for an adaptive visual object retrieval system,” in Proc. of International Parallel and Distributed Processing Symposium (IPDPS'03), p.232a, Apr. 2003.
[36] F. Vermaut, Y. Deville, X. Marichal and B. Macq: “A distributed adaptive block matching algorithm: Dis-ABMA,” Signal Processing: Image Communication, Vol. 16, No. 5, pp. 431-444, Jan. 2001.
[37] T. S. Gunawan, S. S. Tandjung and C. M. Nang: “An efficient parallelization scheme of motion estimation algorithms on Myrinet-connected workstations,” in Proc. of IEEE 6th International Conference on Control, Automation, Robotics and Computer Vision (ICARCV'2000) Dec. 2000.
[38] Fulvio Moschetti: “A Statistical Approach to Motion Estimation,” Ph.D Thesis, Swiss Federal Institute of Technology/Lausanne, 2001.
[39] P. Baglietto, M. Maresca, A. Migliaro and M. Migliardi: “Parallel implementation of the full search block matching algorithm for motion estimation,” in Proc. of The International Conference on Application Specific Array Processors (ASAP'95), pp.182-192, Jul. 1995.
[40] K. Nakamura, M. Ikeda, T. Yoshitome and T. Ogura: “Global rate control scheme for MPEG-2 HDTV parallel encoding system,” in Proc. of IEEE The International Conference on Information Technology: Coding and Computing (ITCC'00), pp.195-200, Mar. 2000.