

Correspondence

Adding Lossless Video Compression to MPEGs

Jianmin Jiang and Guoqiang Xiao

Abstract—In this correspondence, we propose to add a lossless compression functionality into existing MPEGs by developing a new context tree to drive arithmetic coding for lossless video compression. In comparison with the existing work on context tree design, the proposed algorithm features in 1) prefix sequence matching to locate the statistics model at the internal node nearest to the stopping point, where successful match of context sequence is broken; 2) traversing the context tree along a fixed order of context structure with a maximum number of four motion compensated errors; and 3) context thresholding to quantize the higher end of error values into a single statistics cluster. As a result, the proposed algorithm is able to achieve competitive processing speed, low computational complexity and high compression performances, which bridges the gap between universal statistics modeling and practical compression techniques. Extensive experiments show that the proposed algorithm outperforms JPEG-LS by up to 24% and CALIC by up to 22%, yet the processing time ranges from less than 2 seconds per frame to 6 seconds per frame on a typical PC computing platform.

Index Terms—Context tree, lossless video compression, statistics modeling.

I. INTRODUCTION

Although research in lossless video compression is limited, a number of attempts were reported in the literature [1]–[5] working toward removal of both spatial and temporal redundancies. Memon and Sayood [1] described one of the earliest work on lossless video compression based on old JPEG lossless image compression standard. They proposed a number of prediction and error modeling schemes to exploit the spatial and temporal correlations. Since then, research on lossless still image compression has made significant progress and thus the state-of-the-art in lossless compression techniques is now represented by prediction schemes developed in JPEG-LS [9], [10] and CALIC [11] etc. Recently Brunello *et al.* [4] published their work on lossless video compression by investigating issues in three-dimensional prediction, where not only pixels from the current frame but also those pixels from previous frames are taken into consideration in constructing the predictive values. Such prediction schemes are primarily developed out of the still image compression area, where investigations were conducted along the lines of prediction and statistics modeling. Along with the research in still image compression, most of the existing work is limited to investigations on prediction schemes, intending to improve on either its accuracy or its optimization in terms of some cost function measurements [1]–[5], [8]. To this end, we adopted a different line of research to extend the concept of universal statistics modeling by context trees

[6], [7] to fine tune the statistics modeling for efficient and effective lossless video compression, and achieve an appropriate balance among speed, complexity, and performances.

The remaining part is organized into two sections, where Section II describes the proposed algorithm design, and Section III reports our experimental results benchmarked by the best still image compression algorithms, JPEG-LS and CALIC. Some concluding remarks are also included in this section.

II. DESIGN OF LOSSLESS COMPRESSION FUNCTIONALITY FOR MPEGs

Given a sequence of video frames, its structure can be characterized by dividing the sequence into GoPs as such that, apart from the first frame being an I-frame, all others are designated as P-frames only. Each GoP is designed to have 30 frames unless a shot cut is detected [12], where shot boundary is always arranged as the boundary of GoPs in order to maintain the consistency of statistics within a GoP. Following that, motion estimation and compensation is carried out via macro-blocks in the same way as that of MPEGs to remove the temporal redundancy.

To maximize the entropy coding efficiency, we propose a new context tree to gather statistics of input videos based on the Algorithm Context by Rissanen [6] and the universal context modeling by Weinberger [7]. Given a sequence of pixels $x^i = x_1 x_2 \dots x_{i-1} x_i$, the Algorithm Context constructs a context tree by recursively inserting each individual pixel into its previous tree to estimate its statistics and represent all the contexts by those internal nodes inside the tree. Assuming that the current tree T_i has been constructed for the pixel sequence $x^i = x_1 x_2 x_3 \dots x_i$ and the next input pixel is x_{i+1} , the Algorithm Context will insert x_{i+1} into T_i to construct T_{i+1} by traversing the T_i along the path defined by the past sequence $x_i x_{i-1} \dots x_2 x_1$. As a result, the context tree is essentially a suffix tree since the most recent pixel is always visited first. To estimate the probabilities for each pixel inserted in the tree, its occurrence count needs to be incremented and the tree updated accordingly. As lossless compression is essentially achieved by maximizing the probability estimation assigned to each pixel to be encoded and such estimation should be conducted under the context of its previously occurred pixels, selection of which context inside the tree to encode the next pixel becomes an important issue. At present, this is always addressed via the principle that the context that has sequentially assigned the largest probability for its pixel occurrences in the past is most likely to achieve the largest probability again for the next pixel. To this end, various optimization algorithms have been proposed and reported in the literature, including backward pruning from the leaves to the root via dynamic programming [5], and efficiency index stored and updated at each node [7], which only compares the two adjacent nodes (one node and its parent node) to determine the optimality of each node visited.

For such a universal context tree, significant drawbacks have been identified by Weinberger *et al.* [7] as such that, although the context tree is capable of achieving optimal statistics modeling theoretically and eventually, its practical performance on compressing grey scale images is very poor. This is because that the universality of the context modeling is unable to exploit those unique features embedded inside images, including correlation between adjacent pixels, smoothness of regional textures, tighter finite-state models illustrated by auto-regression etc. As a result, Weinberger *et al.* [7] proposed a range of ideas to improve the universal context tree modeling characterized in PCM domain and DPCM domain. Such improvements are mainly motivated by the intention to

Manuscript received April 20, 2005; revised August 26, 2005. This work was supported in part by the Chinese Natural Science Foundation and by the the EU Framework-6 Programme under Contract IST-4-027312.. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Yo-Sung Ho.

J. Jiang is with the Faculty of Informatics & Computing, Southwest China University, Chongqing, China and also with the School of Informatics, University of Bradford, Bradford BD7 1DP, U.K. (e-mail: j.jiang1@Bradford.ac.uk).

G. Xiao is with the Faculty of Informatics & Computing, Southwest China University, Chongqing, China (e-mail: gqxiao@swnu.edu.cn).

Digital Object Identifier 10.1109/TMM.2006.870721

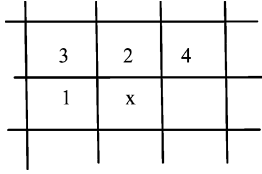


Fig. 1. Context structure.

develop a practically usable context modeling technique, which should lead to a compression algorithm with low cost and low complexity. Indeed, such proposed changes also originated the latter JPEG lossless compression standard, JPEG-LS. In summary, their proposed improvements can be highlighted by 1) parametric statistics modeling, in which the occurrence counts for every pixel are replaced by discretised Laplacian probability density function: $f^{(s)}(x) = (\gamma^{(s)}/2)e^{-\gamma^{(s)}|x-\mu|}$. In this way, the large alphabets of pixels can be reduced to two parameters, γ and μ , which are estimated by previously encoded pixels within certain boundary of contexts; 2) context construction out of quantized pixels to partially lump together those siblings inside the context tree. As a result, when the quantization is implemented along the binary representation of each pixel value, starting from the most significant bit to the second most significant bit, etc. the context tree becomes a binary tree; and 3) construction of contexts from pixel differences rather than pixels themselves (DPCM domain).

While the above modification exploits those unique features inside the still images, it may not achieve the full potential of those features inside MPEG videos. Our experiments on MPEG videos reveal that 1) due to the motion estimation and compensation, the predictive errors illustrate less correlation with those remote siblings inside the context tree. In other words, the correlation tends to converge inside closer neighborhoods, and thus finer tune of statistics modeling becomes more desirable; 2) after the motion estimation and compensation, the predictive errors distribute themselves within a much smaller data range, i.e., the alphabet of the error values is reduced to around 30 instead of 256 for 8 bit grey levels. As a result, the observations given above justifies that the original context tree should be modified in a different direction to take into considerations those unique features inside MPEG videos. Specifically, the first observation indicates that lumping those remote siblings together may no longer be a desirable option for the context tree, and the second observation indicates that parameterization of probability estimation should also be reconsidered. To overcome those drawbacks of original context tree as identified by Weinberger *et al.* [7] in light of the MPEG videos, we propose the following schemes to replace what were proposed in [7]:

Firstly, instead of parametric statistical modeling, we reduce the context modeling complexity by restricting the context to a maximum size of four symbols (i.e., four motion compensated errors in the case of MPEG videos). As a result, the context modeling template is fixed to four previously encoded errors and their order is also fixed as illustrated in Fig. 1. Correspondingly, at each internal node, the probabilities for all its branches will be estimated by their occurrence counts or frequency counts rather than by parameterised Laplacian density distribution function;

Secondly, we set up a threshold to quantize the context value distribution into two sections, as the motion compensated errors mostly distribute themselves within the range of [0,30]. Specifically, given a context sequence: $context = e_1e_2e_3e_4$, the value of each error inside the context is redefined as follows when the context tree is searched and updated:

$$e_i = \begin{cases} e_i & \text{if } e_i < T_c \\ T_c & \text{otherwise} \end{cases} \quad (1)$$

where T_c is a context threshold, which plays the role to reduce the size of the context tree and thus improve the compression speed. In our algorithm design, T_c is empirically designed as 15, based on the fact that most of the error values after motion estimation and compensation are limited within the range of [0, 30].

From (1), it can be seen that, essentially, what we proposed is 1) for those contexts with small error values, their statistics modeling is fine tuned by frequency counts; and 2) for those contexts with large error values, their statistics is lumped together via the context threshold to exploit the points made by Weinberg *et al.* [7];

Thirdly, the tree will still be traversed by symbols rather than by their binary representations, since lumping together those remote siblings needs to be fine tuned and its spirit is already exploited by (1). To resolve the optimization problem on selecting the best possible context for probability estimation, we change the context tree from its original suffix tree status into a prefix tree status. That is, given a context sequence: $x^4 = x_1x_2x_3x_4$, we traverse the tree along the order from x_1 to x_4 rather than from x_4 to x_1 . In this way, the context can be naturally selected along the traversing route, in which the possible selection can be represented as: $x_1, x_1x_2, x_1x_2x_3$, or $x_1x_2x_3x_4$. With this context selection structure, it can be inferred that, starting from x_1 , if traversing to x_2 in the context tree is successful, the context x_1x_2 will be more likely to have a larger probability for the next pixel to be encoded than the context x_1 . Given the context template as shown in Fig. 1, the probability estimation for the next pixel/error x under all the contexts can be summarized as follows:

$$P(x|x_1) == \frac{C_{x|x_1}}{\sum_{x_i \in \alpha} C_{x_i|x_1}} = \frac{C_{x|x_1}}{C_{x_1}} \quad (2)$$

$$P(x|x_1x_2) == \frac{C_{x|x_1x_2}}{\sum_{x_i \in \alpha} C_{x_i|x_1x_2}} = \frac{C_{x|x_1x_2}}{C_{x_1x_2}} \quad (3)$$

$$P(x|x_1x_2x_3) == \frac{C_{x|x_1x_2x_3}}{\sum_{x_i \in \alpha} C_{x_i|x_1x_2x_3}} = \frac{C_{x|x_1x_2x_3}}{C_{x_1x_2x_3}} \quad (4)$$

$$P(x|x_1x_2x_3x_4) == \frac{C_{x|x_1x_2x_3x_4}}{\sum_{x_i \in \alpha} C_{x_i|x_1x_2x_3x_4}} = \frac{C_{x|x_1x_2x_3x_4}}{C_{x_1x_2x_3x_4}} \quad (5)$$

where $C_{x|x_1x_2}$ stands for the occurrence counts of x under the context of x_1x_2 , $C_{x_1x_2}$ stands for the occurrence counts of the sequence x_1x_2 , $==$ is used to indicate that the probability is estimated by not equal to, and α for the alphabet of all the possible predictive error values. As our algorithm design is limited to sequential video compression, where the context match is conducted sequentially along a fixed order, the smaller contexts are always part of those larger contexts. Therefore, we have:

$$C_{x_1x_2x_3x_4} \leq C_{x_1x_2x_3} \leq C_{x_1x_2} \leq C_{x_1} \quad (6)$$

which is likely to lead

$$P(x|x_1x_2x_3x_4) \geq P(x|x_1x_2x_3) \geq P(x|x_1x_2) \geq P(x|x_1). \quad (7)$$

Due to the fact that $C_{x|x_1x_2x_3x_4} \leq C_{x|x_1x_2x_3} \leq C_{x|x_1x_2} \leq C_{x|x_1}$, the above is not guaranteed. However, observations of practical video statistics reveal that (7) is maintained in most cases. Detailed analysis follows.

Without losing generality, we take the case of comparing $P(x_i|x_1x_2)$ with $P(x_i|x_1)$, where x_i is the i th input symbol to be encoded, to examine the cases that sufficient statistics has already been established and frequency counts for high order contexts are available. Since the shorter context x_1 is always part of the longer context x_1x_2 , we have

$$C_{x_i|x_1} = \sum_{x \in \alpha} C_{x_i|x_1x} \quad (8)$$

Therefore, (2) can be rearranged as (9), shown at the bottom of the page. In comparison with $P(x_i|x_1x_2) = C_{x_i|x_1x_2}/\sum_{x \in \alpha} C_{x|x_1x_2}$, their difference is $\sum_{x \in \alpha \& x \neq x_2} C_{x_i|x_1x_2}$ in the numerator of (9), and $\sum_{x \in \alpha \& x \neq x_2} C_{x_i|x_1x_2} + \sum_{y \in \alpha \& y \neq x_2} \sum_{x \in \alpha \& x \neq x_i} C_{x|x_1y}$ in the denominator of (9). As a matter of fact, the difference between the numerator and the denominator of (9) is only the term: $\sum_{y \in \alpha \& y \neq x_2} \sum_{x \in \alpha \& x \neq x_2} C_{x|x_1y}$. As the context tree is already established sufficiently, the term $\sum_{y \in \alpha \& y \neq x_2} \sum_{x \in \alpha \& x \neq x_2} C_{x|x_1y}$ will become very large, and thus makes (9) to be smaller than $P(x_i|x_1x_2)$.

To examine the cases where context tree construction is at its early stage and the statistics available can only provide estimation of lower order conditional probabilities, we compare $P(x_i)$ with $P(x_i|x_1)$ as follows:

$$P(x_i) = \frac{C_{x_i}}{\sum_{x \in \alpha} C_x} = \frac{C_{x_i|x_1} + \sum_{x \in \alpha \& x \neq x_1} C_{x_i|x}}{C_{x_1} + \sum_{x \in \alpha \& x \neq x_1} C_x}. \quad (10)$$

In practice, elements in $\sum_{x \in \alpha \& x \neq x_1} C_{x_i|x}$ are more likely to be zeros than those of unconditional frequency counts $\sum_{x \in \alpha \& x \neq x_1} C_x$. This drives $P(x_i)$ to be smaller than $P(x_i|x_1) = C_{x_i|x_1}/C_{x_1}$.

In summary, for MPEG motion compensated error frames, the proposed context tree can be established by the following steps:

- Step 1) Starting with a root node with all counts set to zero, which is an initial context tree T_0 with an empty context, recursively input the next error and construct its context as $x_1x_2x_3x_4$ according to the template given in Fig. 1. With those predictive errors on the boundaries of the error-frame, the context sequence will be reduced according to its position. These include 1) the first error value on the first row will have an empty context; 2) other errors on the first row will have context x_1 ; 3) the errors on the first column will have context x_2x_4 ; and 4) the errors on the last column will have context $x_1x_2x_3$.
- Step 2) Given the i th error x_i to be encoded, search the current context tree, T_{i-1} , to match its context sequence $x_1x_2x_3x_4$ starting from the root node. The search stops whenever a different error is encountered or the full sequence of $x_1x_2x_3x_4$ has been matched. Following the stopped search, there are two different outcomes, which include 1) the search stopped at the root node with no context matched; and 2) the search stopped with one of the four contexts, x_1 , x_1x_2 , $x_1x_2x_3$, or $x_1x_2x_3x_4$, being matched; For 1), the algorithm goes to Step 3), and for 2), the algorithm goes to Step 4);
- Step 3) Encode x_i with the statistics model at the root node. In the case that there is no frequency count for x_i , encode "EoS" to indicate such status and move to the basic model [14], [15], where every possible error value is guaranteed to have a frequency count. Go to Step 5);

- Step 4) Encode x_i with the statistics model at the stopping internal node. If the search stops over a branch, use the statistics model at the internal node, which is immediately above the stopping point. If there are no frequency counts for x_i at the internal node, move up by one level and use the statistics model at its parent node. This upwards moving should carry on until the frequency counts for x_i are found or the basic model is reached. During this process, each upward moving should have one "EoS" being encoded;
- Step 5) This step is mainly designed to update the context tree according to the outcome of present traversing. Such tree updating can be described in three operations. These include 1) Inserting unsuccessfully matched part of the sequence $x_1x_2x_3x_4x_i$ into the tree starting from the stopping point. If the search is stopped at an internal node, create a new branch to represent the unsuccessfully matched sequence. Otherwise, split the branch by creating a new internal node and a new branch at the stopping point. The new branch starts with the first symbol of the nonmatched sequence within $x_1x_2x_3x_4x_i$. If this first symbol is not x_i , another new branch should be added to this new internal node representing x_i . As a result, a new statistics model is also created at the new internal node; 2) for every "EoS" encoded, add a new branch representing x_i to the internal node; and 3) upgrade the frequency count for x_i in all the statistics models along the searching route, where frequency count for x_i is available.

III. EXPERIMENTS AND CONCLUSIONS

To evaluate the proposed lossless compression functionality to be added into MPEGs, we adopted six publicly available video clips to enable an open-assessment. These include: *mobile*, *flower-garden*, *Susie*, *mom*, *table-tennis* and *OSU-2*. Since little work was reported for video compression and no benchmarking software is available in the public domain, we use the existing state-of-the-art in lossless image compression to provide a benchmark for the proposed algorithm. We justify such benchmark selection by the following arguments: 1) the existing work on lossless video compression reported so far all used the lossless image compression as their benchmark [1]–[5]. These include the old lossless JPEG compression standard used by Memon *et al.* [1], JPEG-LS by Brunello *et al.* [4] and Martins *et al.* [2], and LOCO, CALIC by GlicBawls [3]. 2) As the six video clips included in the test data set are available in the public domain, any further comparisons can be made by anybody with his own techniques without repeating the algorithm proposed here by simply using the same test data for their algorithm evaluation.

As described in the Introduction, the best algorithm in compressing still images to date can be summarized as JPEG-LS, CALIC, and integer wavelets-based methods [16]. As it was reported in medical image compression [15] that the integer wavelets based compression performs

$$\begin{aligned} P(x_i|x_1) &= \frac{\sum_{x \in \alpha} C_{x_i|x_1x}}{\sum_{y \in \alpha} \sum_{x \in \alpha} C_{x|x_1y}} = \frac{C_{x_i|x_1x_2} + \sum_{x \in \alpha \& x \neq x_2} C_{x_i|x_1x}}{\sum_{x \in \alpha} C_{x|x_1x_2} + \sum_{y \in \alpha \& y \neq x_2} \sum_{x \in \alpha} C_{x|x_1y}} \\ &= \frac{C_{x_i|x_1x_2} + \sum_{x \in \alpha \& x \neq x_2} C_{x_i|x_1x}}{\sum_{x \in \alpha} C_{x|x_1x_2} + \sum_{y \in \alpha \& y \neq x_2} C_{x_i|x_1y} + \sum_{y \in \alpha \& y \neq x_2} \sum_{x \in \alpha \& x \neq x_i} C_{x|x_1y}} \end{aligned} \quad (9)$$

TABLE I
EXPERIMENTAL RESULTS MEASURED BY BITS/PIXELS

Video clips	Context-tree	CALIC	JPEG-LS	Improvement over JPEG-LS	Improvement over CALIC
Mobile	3.29	3.51	3.67	11.5%	6.6%
Flower-garden	3.96	4.19	4.42	11.6%	5.8%
Susie	2.27	2.68	2.83	24.4%	18.1%
Mom	2.22	2.70	2.74	23.6%	22%
Table-tennis	2.82	3.04	3.10	11.1%	8%
OSU-2	3.46	3.65	3.79	9.5%	5.5%

TABLE II
EXPERIMENTAL RESULTS MEASURED BY PROCESSING SPEED

Video clips	Frame size	Context-tree compressed speed
Mobile	720x576	6.660 sec/frame
Flower-garden	352x240	2.503 sec/frame
Susie	720x480	4.145 sec/frame
Mom	360x240	1.222 sec/frame
Table-tennis	352x240	1.756 sec/frame
OSU-2	352x240	1.560 sec/frame

slightly inferior to that of JPEG-LS and CALIC in terms of compression ratios, we only adopted JPEG-LS and CALIC as our benchmark.

Table I illustrates the first group of experimental results to measure the compression performances in bit/pixel. As seen in the table, the proposed context-tree algorithm outperforms JPEG-LS by up to 24% and CALIC by up to 22%. In comparison with JPEG-LS and CALIC, the strength of the proposed context tree can be highlighted as 1) the proposed context tree is able to maximize the conditional probability estimation by rearranging the context sequence search along a prefixed order; 2) the context tree fine tuned the statistics modeling which captured the nature of those motion compensated errors by focusing frequency counts upon those small but frequently occurred errors; and 3) for those large and rarely occurred error values, the proposed context tree lumps them together to estimate their statistics via quantized statistics regions, reserving the strength of universal statistics modeling as suggested by Weinberg in [7]. In the case of JPEG-LS and CALIC, the second option produced better compression efficiency, which reveals that, after MPEG motion estimation and compensation, there still exists spatial redundancy to be removed. The experimental results suggest that both JPEG-LS and CALIC worked well in terms of removing the remaining spatial redundancy.

From the algorithm design as described in Section II, it can be seen that the context-tree algorithm does incur certain level of complexity due to the tree organization of all contexts. To test how this context-tree algorithm performs in terms of processing speed, we implemented the algorithm on a PC of Pentium-4 2.0G with 512M memory, and compressed all the video clips inside the test data set. The results are summarized in Table II, from which it is seen that the processing speed ranges from 1.756 s/frame to 6.66 s/frame. These illustrate that the proposed algorithm is practically acceptable in terms of processing speed.

In this correspondence, we proposed a context tree to enable MPEGs to have a new functionality for lossless video compression. Such work will extend MPEGs to other video communication applications, where any loss of information is not tolerated. Benchmarked by existing state-of-the-art in still image lossless compression, the proposed design has illustrated high level of efficiency and effectiveness for lossless video compression.

REFERENCES

- [1] N. D. Memon and K. Sayood, "Lossless compression of video sequences," *IEEE Trans. Commun.*, vol. 44, pp. 1340–1345, Oct. 1996.
- [2] B. Martins and S. Forchhammer, "Lossless compression of motion compensation," in *Proc. IEEE DCC'98*, Los Alamitos, CA, 1998, p. 560.
- [3] B. Meyer and P. Tischer, "Glicbawls—grey level image compression by adaptive weighted least squares," in *Proc. DCC01*, Snowbird, UT, Mar. 2001.
- [4] D. Brunello, G. Calvagno, G. A. Mian, and R. Rinaldo, "Lossless compression of video using temporal information," *IEEE Trans. Image Process.*, vol. 12, no. 2, pp. 132–139, Feb. 2003.
- [5] R. Nohre, "Topics in descriptive complexity," PhD dissertation, Dept. Comput. Sci., Technical Univ. Linkoping, Sweden, Oct. 1993.
- [6] J. Rissanen *et al.*, "A universal data compression system," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 656–664, 1983.
- [7] M. J. Weinberger, J. J. Rissanen, and R. B. Arps, "Applications of universal context to lossless compression of grey-level images," *IEEE Trans. Image Process.*, vol. 5, no. 4, pp. 575–586, 1996.
- [8] E. A. Riskin, "Optimal bit allocation via the generalized BFOS algorithm," *IEEE Trans. Inform. Theory*, vol. 37, no. 2, pp. 400–402, 1991.
- [9] M. J. Weinberger *et al.*, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *IEEE Trans. Image Process.*, vol. 9, pp. 1309–1324, Aug. 2000.
- [10] J. Jiang, "A low cost content adaptive and rate controllable near lossless image codec in DPCM domain," *IEEE Trans. Image Processing*, vol. 9, no. 4, pp. 543–554, Apr. 2000.
- [11] X. L. Wu and N. Memon, "Context-based, adaptive, lossless image coding," *IEEE Trans. Commun.*, vol. 45, no. 4, pp. 437–444, Apr. 1997.
- [12] C. Taskiran *et al.*, "ViBE: a compressed video database structured for active browsing and search," *IEEE Trans. Multimedia*, vol. 6, no. 1, pp. 103–118, 2004.
- [13] J. Jiang, "A novel design of arithmetic coding for data compression," in *Proc. Inst. Elect. Eng., Comput. Digital Techn.*, Nov. 1995, vol. 142, no. 6, pp. 419–424, 1350-2387.
- [14] J. Xia, J. Jiang, S. Y. Yang, and C. H. Hou, "An empirical study to turn MPEG-2 into lossless video compression," in *Proc. VIE03, IEE International Conference on Visual Information Engineering*, Surrey, U.K., pp. 234–241, ISBN: 085 296 7578.
- [15] D. A. Clunie, "Lossless compression of greyscale medical images: effectiveness of traditional and state-of-the-art approaches," in *Proc. SPIE*, vol. 3980, pp. 74–84, Medical Imaging 2000: PACS Design and Evaluation.
- [16] A. Said and W. Pearlman, "An image multiresolution representation for lossless and lossy compression," *IEEE Trans. Image Process.*, vol. 5, pp. 1303–1310, Sep. 1996.