Computers in Biology and Medicine

An efficient compression scheme for 4-D medical images using hierarchical vector quantization and motion compensation

Binh P. Nguyen [a,*], Chee-Kong Chui [b], Sim-Heng Ong [a,c], Stephen Chang [d]

[a] Department of Electrical and Computer Engineering, National University of Singapore, 4 Engineering Drive 3, Singapore 117576, Singapore
[b] Department of Mechanical Engineering, National University of Singapore, 9 Engineering Drive 1, Singapore 117576, Singapore
[c] Division of Bioengineering, National University of Singapore, 7 Engineering Drive 1, Singapore 117574, Singapore
[d] Department of Surgery, National University Hospital, 5 Lower Kent Ridge Road, Singapore 119074, Singapore

ARTICLE INFO

Keywords:
4-D compression
Hierarchical vector quantization
Motion estimation
Block distortion measure Variance of residual

ABSTRACT

This paper proposes an efficient compression scheme for compressing time-varying medical volumetric data. The scheme uses 3-D motion estimation to create a homogenous preprocessed data to be compressed by a 3-D image compression algorithm using hierarchical vector quantization. A new block distortion measure, called variance of residual (VOR), and three 3-D fast block matching algorithms are used to improve the motion estimation process in term of speed and data fidelity. The 3-D image compression process involves the application of two different encoding techniques based on the homogeneity of input data. Our method can achieve a higher fidelity and faster decompression time compared to other lossy compression methods producing similar compression ratios. The combination of 3-D motion estimation using VOR and hierarchical vector quantization contributes to the good performance.

1. Introduction

The acquisition of medical images in digital format has been rapidly increasing over the last two decades. One characteristic of medical images is their very large storage requirements. For example, 256 MB is required for a $512 \times 512 \times 512$ 16-bit computed tomography (CT) dataset. The size of a typical medical time-varying volumetric image, which is a collection of volume images captured in a sequence of time steps, can range from hundreds of megabytes to hundreds of gigabytes. These datasets are often stored on servers and transmitted to clients when needed. The manipulation and visualization of such huge datasets is a challenging problem due to overwhelming data size, insufficient memory and *I*/*O* bandwidth, and heavy computational requirements. In addition to developing fast processing algorithms and using high performance hardware, compression would be extremely useful in such situations. The primary objective of medical image compression methods is to reduce the large amount of data to be stored, transmitted, or processed while preserving important diagnostic information.

Compression techniques can be either lossless or lossy. Lossless algorithms allow exact reconstruction of the original data, while lossy algorithms introduce some error or loss after the compression process. Lossless techniques may be used in cases where the datasets are not very large, but are not suitable for applications with limited transmission bandwidth or storage constraints, e.g., in picture archiving and communication systems (PACS), teleradiology, and remote dataset browsing systems. Lossy methods are more appropriate for these applications since they offer higher compression ratios (CRs). Popular lossy compression methods for medical images often use an image transform, such as the discrete cosine transform (DCT) [1–4] or the wavelet transform [5–10], followed by quantization and/or a coefficient partitioning technique such as the embedded zerotree wavelet (EZW) [11] and set partitioning in hierarchical trees (SPIHT) [12], and finally a symbol coding method. The basic idea of applying an image transform to volume data is to obtain a different distribution in the transform domain that can make quantization and symbol coding more efficient.

In the case of manipulating or rendering time-varying medical volumetric images, which can be considered four-dimensional (4-D) images, data compression becomes an even more important requirement. Among the relatively small number of published papers on 4-D medical image compression, methods that treat the data as a 4-D field are dominant. Wilhelms and Gelder [13] proposed a 4-D tree, which is an extension of the octree, for compression and rendering of time-varying data. Each tree node contains a model of the data represented as a fixed number of

* Corresponding author. Tel.: +65 9699 3579.
E-mail address: phubinh@nus.edu.sg (B.P. Nguyen).

basis functions, a measure of the modeling error, and a measure of the importance of the corresponding data. Although their method can provide flexibility in controlling the image quality and rendering speed by using user-defined tolerances of modeling error and data importance, the compression ratios are not as good as other methods. Other methods often use the 4-D discrete wavelet transform (DWT) in a lossy compression scheme. Zeng et al. used the 4-D DWT and extended the EZW to 4-D to encode echocardiographic data [14]. Although their method can handle arbitrarily sized input data and offers a wide range of compression ratio, the fidelity of decompressed data in term of pick signal to noise ratio (PSNR) is rather low. Another limitation is that their experiments were conducted based on only one set of small size 4-D ultrasound data; no other image modality nor large dataset were examined. In another work, Lalgudi et al. [15] studied the extension of JPEG2000 to 4-D and proposed a method to compress fMRI images using the DWT and JPEG2000. A 1-D DWT is applied along the time dimension of a 4-D data first, followed by another 1-D DWT along the z-axis, and finally the resulting 2-D wavelet coefficients are compressed using JPEG2000. The experimental results show that the method is slightly better than other methods using 3-D JPEG2000. Another method was proposed by the same group in [16] with the extension of the EZW and SPIHT to 4-D to use with the 4-D DWT to compress fMRI and 4-D ultrasound images. Nevertheless, only an insignificant improvement was achieved compared to their previous work, i.e. the 4-D JPEG2000. Liu and Pearlman [17] extended subband block hierarchical partitioning (SBHP), another coefficient partitioning technique originally described in [18], to 4-D and used it with 4-D wavelet decomposition for progressive fidelity and resolution decompression of 4-D images. In lossy mode, although the method outperforms two other possible 3-D SBHP coding schemes, no comparison with other lossy methods are reported. Generally, a method that relies on the 4-D wavelet transform can offer relatively high CRs with reasonable fidelity. However, it is not easy to achieve fast decompression due to the complexity of the 4-D wavelet transform. In addition, a number of time steps (i.e., frames) will have to be decoded even if only one of them is to be manipulated or rendered.

The other approaches process the time and spatial domain separately. It is obviously possible to compress 3-D volumes independently but this approach does not exploit the dependency of corresponding voxels in different volumes. A good method should exploit the high correlation between volumes in 4-D medical images. Several methods consider a 4-D image as 3-D video and extend the state-of-the-art methods in video coding to 3-D for exploiting redundancies in all four dimensions. Kassim et al. [19] proposed a combination of the 3-D integer wavelet transform and 3-D motion compensation for lossy-to-lossless compression of 4-D medical images. Their experimental results showed that the lossless mode of this method can reduce the coding bit rates by approximately 25% compared with a method that applies a conventional 3-D compression technique for each time step, and the drop in image quality in the lossy mode is hardly noticeable when compared to other methods producing the same CR. Sanchez et al. [20] introduced a lossless compression method for 4-D medical images based on the most advanced features of the H.264/AVC standard, e.g., multi-frame motion compensation, variable block size and sub-pixel mode in motion estimation. The CRs obtained are superior to 3D-JPEG2000 when encoding fMRI images but similar to other lossless compression methods when applied to other types of medical images. Recently, this method is extended in [21] with a new multiframe motion compensation process that more effectively reduces the redundancy in both spatial and temporal dimensions by employing a 4-D search, variable-size block matching, and bidirectional prediction. The output of the motion compensation process, i.e., the residual and motion vectors, are compressed using a new context-based adaptive binary arithmetic coder designed based on the probability distribution of the data. Evaluation results show that compared to 4-D JPEG2000 and H.264/AVC, this method archives an average improvement on compression ratio of 13% on real fMRI data. However, the outcome of the method when applied to other medical imaging modalities is still unknown.

In this paper, we propose a lossy compression method that uses hierarchical vector quantization (HVQ) and 3-D motion compensation for the compression of 4-D medical images. While this approach is motivated by the encoding technique presented in [22], we have introduced a number of modifications and extensions and included a novel 3-D motion estimation algorithm for improving the fidelity of the decompressed data. Simulations confirm that we are able to obtain better fidelity and faster decompression compared to other compression methods while achieving similar CRs.

The rest of the paper is organized as follows. An overview of the new 4-D medical image compression scheme is presented in Section 2. A detailed description of the 3-D image compression algorithm using HVQ is provided in Section 3. Section 4 presents the 3-D motion estimation and compensation algorithms used in the compression scheme. The experiments are described in Section 5 and discussed in Section 6. The paper concludes in Section 7.

2. Overview of proposed scheme

In our work, video compression concepts are applied to 4-D medical images, which consist of sequences of 3-D image frames. Fig. 1 provides an overview of the encoding process. In order to encode a group of 3-D image frames, the first frame (the key frame) F_0 is separately encoded and reconstructed using a 3-D compression algorithm. The 3-D compression algorithm used in our work is based on HVQ, and is presented in Section 3. To encode the remaining frames (i.e., the intermediate frames),

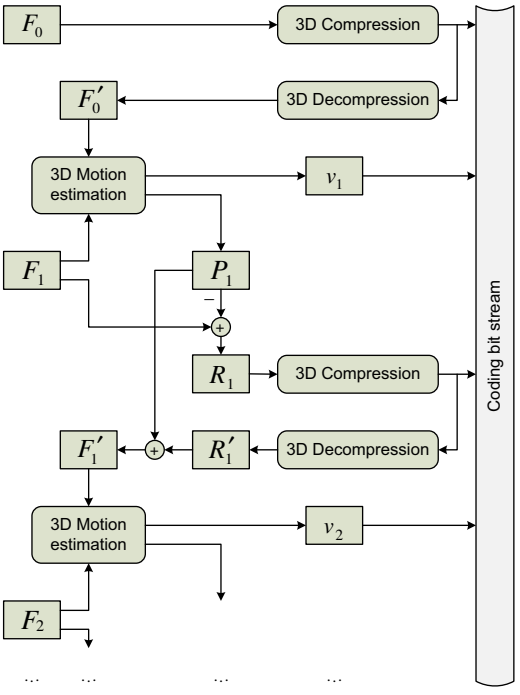

Fig. 1. Overview of the encoding process.

a new 3-D motion estimation algorithm, described in Section 4, is applied to produce the predicted frame P_i at time step i from the reconstructed frame F'_{i-1} so that P_i is matched with the frame at the same time step F_i as closely as possible (according to a matching criterion). F_i is then motion-compensated by subtracting P_i to produce a motion-compensated residual frame R_i. R_i is encoded and transmitted with a set of motion vectors v_i, which is the information required to recreate P_i from F'_{i-1}. Subsequently, the encoded version of R_i is decoded to produce R'_i, which is then added to P_i to generate the reconstructed frame F'_i to be used to encode the next frame. Optionally, the resulting bit stream can be arithmetic coded for further compression.

In the decoding phase, the key frame and residual frames are decoded using the decompression process corresponding to the 3-D compression algorithm. These frames will be used to successively reconstruct the intermediate frames. The first predicted frame R_1 is reconstructed based on the reconstructed key frame F'_0 and the associated motion vectors v_1. Then P_1 is added to the decoded version of the first residual frame, R'_1, to produce the reconstructed first intermediate frame F'_1. Subsequently, F'_1 and the decoded version of the second residual frame, R'_2, with its associated motion vectors v_2 are used to reconstruct the second intermediate frame F'_2. The process continues until all intermediate frames are reconstructed.

3. Three-dimensional image compression using HVQ

Part of our 3-D compression algorithm is based on the hierarchical vector quantization scheme proposed by Schneider and Westermann [22], henceforth referred to as SW. In this method, the volume data is initially partitioned into disjoint cubes of $4 \times 4 \times 4$ voxels. Each cube is down-sampled by a factor of two by averaging disjoint sets of $2 \times 2 \times 2$ voxels. A 64-component vector (i.e., the first level data) is formed to store the difference between the original data samples and the respective down-sampled value. By applying the same process to the down-sampled version, a single value that represents the mean value of the entire $4 \times 4 \times 4$ cube and an 8-component vector (i.e., the second level data) carrying the residual data is obtained. The mean of the cube is stored in a 1-component vector (i.e., the third level data). In the next step, all the 64-component and 8-component vectors are separately sent to two vector quantizers to produce two codebooks containing 64- and 8-component codewords. This method is mainly applied to 8-bit volume datasets with the assumption that the length of each codebook is 256. Therefore, each cube is represented by three 8-bit values: one value represents the mean of the cube while the other two are indices into the respective codebooks representing the difference information. To decode a particular cube, its mean value and the difference information from the two codebooks are summed. The vector quantizer uses a principal component analysis (PCA) splitting scheme to find an initial codebook, which is then refined by using the LBG algorithm [23]. Some optimizations are applied to the refinement step, including restricted searching to a k-neighborhood of the original cell and partial searches for a speedup of the distortion calculation. The major steps of the method are shown in Fig. 2.

The vector quantization scheme of the SW method can achieve compression rates and fidelity similar to that of wavelet based compression [22]. Decompression speed is extremely fast due to the simple decoding. Furthermore, this compression scheme can perform decompression and rendering simultaneously on a graphics processing unit (GPU) to achieve the interactive frame rate for visualization applications. Our investigations have revealed that it is still possible to improve on the performance of SW by making several modifications. Firstly, regardless of the sizes of the two codebooks and other additional information, the CR for an 8-bit volume dataset using two 8-bit index codebooks is limited by a factor of $64 \div 3 \approx 21.3$. The CR in vector quantization is fixed for a given image resolution and does not depend on the content of the volume. For a $4 \times 4 \times 4$ cube having all voxels of the same intensity value, which is commonly found in medical images, this method uses 3 bytes to represent 64 bytes of voxels. However, this cube can be encoded in a more compact form using only one byte representing the mean value of all voxels in the cube, thus leading to an improvement in CR. Secondly, although this method is applied to 8-bit volumes using two codebooks of 256 bytes in length, it can be extended for encoding a volume with other bit-width formats using two codebooks with lengths other than 256. Nowadays, most imaging modalities store image data in a 16-bit format, and for encoding 16-bit volume data, the two codebooks should be wider for better fidelity.

The SW method is used in our 3-D compression algorithm with the following enhancements (Fig. 3). In the first modification, we divide the partitioned cubes into two types based on their homogeneity. Type 1 refers to cubes with variance values smaller than a pre-defined threshold. A Type 1 cube is represented by only its mean value, which is more compact than decomposition and vector quantization. Type 2 cubes comprise the remaining cubes, which are encoded using the original scheme. For a cube, its type is denoted by a type bit (or HVQ-enabled bit) and encoded accordingly in the processing step. Since our new motion estimation method (Section 4) aims to produce cubes with minimum variance values, and due to the characteristics of 3-D medical images, there should be numerous Type 1 cubes in a volume dataset. This leads to an increase in CR. A second benefit is that only cubes having variance values greater than the threshold need to be quantized, thus considerably reducing the amount of computation and the quantization error (since the number of input vectors is decreased). This enhancement also has the potential to improve the fidelity of the reconstructed volume if an appropriate threshold is chosen. Furthermore, the decoding speed should be faster since no decompression is required for all Type 1 cubes; only the duplication of their mean values is performed.

The second modification to the SW method is that the lengths of the two codebooks are widened from 256 (or 8-bit index) up to

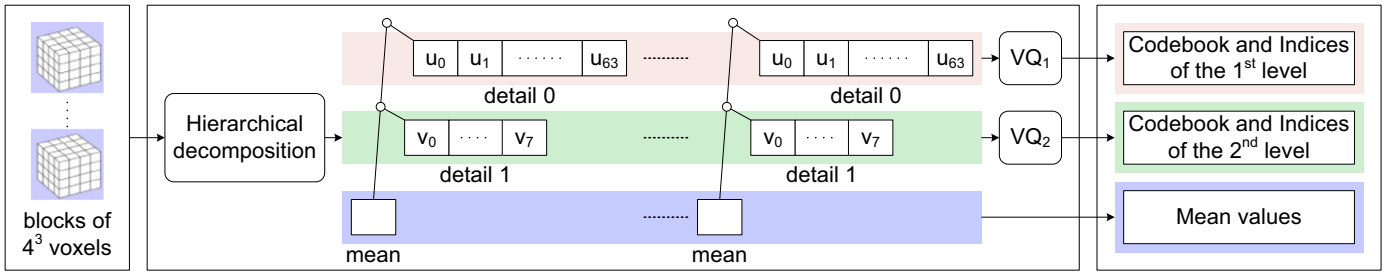

Fig. 2. Hierarchical decomposition and quantization of volumetric data.

4096 (or 12-bit index) for a significant improvement in fidelity with a slight trade-off in CR. The experiments in Section 6 will demonstrate the efficiency of the enhanced compression algorithm.

Please note that the cube size of $4 \times 4 \times 4$ voxels was chosen for the balance between the CR, fidelity and processing time. There are possibilities of choosing other cube size, leading to the variation in these factors. However, this paper does not cover analysis on the size of the partitioned cubes. In addition, although extra bits are needed to identify the two cube types, the overhead does not really affect the overall CR since the added storage is very small compared to the volume size (with the ratio of 1 bit to 64 or 128 bytes in most cases).

4. Three-dimensional motion estimation and compensation

In video coding, motion estimation and motion compensation are very important since they are used to reduce the temporal

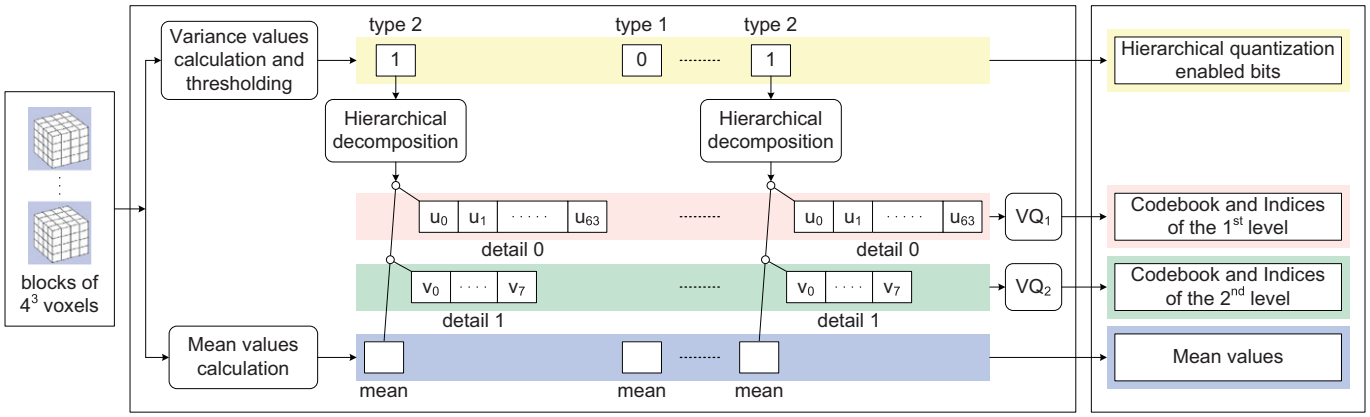

Fig. 3. Enhanced 3-D image compression scheme.

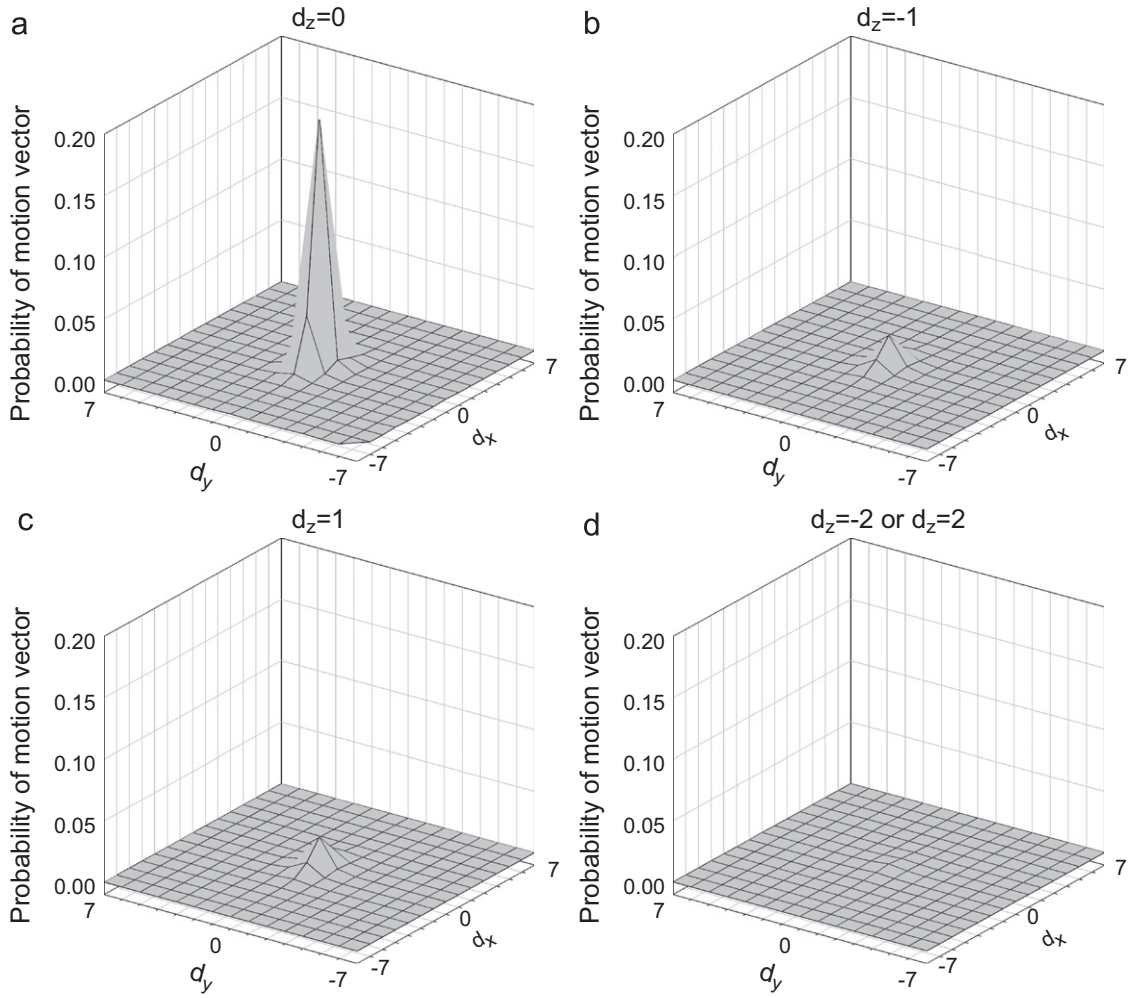

Fig. 4. Motion vector probability distribution with different values of d_z: (a) $d_z=0$; (b) $d_z=-1$; (c) $d_z=1$; and (d) $|d_z|=2$.

redundancy information between successive frames, thus improving CR. Among existing motion estimation methods, block matching algorithms (BMAs) are widely used because of their simplicity and effectiveness. Block matching aims to find, within a search window, the best-matched block from the previous frame based on a block distortion measure or other matching criteria. The displacement of the best-matched block is described as a motion vector relative to the block in the current frame. The algorithm that can find the best-matched block is the full search (FS) method, which evaluates all the candidate blocks within the search window. However, the high computational cost of FS limits its application in practice. In order to reduce the computational complexity, many fast BMAs have been proposed. Most fast BMAs are designed on the assumption that block distortion decreases monotonically when the search position moves toward the minimum distortion point. It is thus not necessary to check all the search points in the search window since the best-matched position can be found by following the changing trend of the distortion. Various search patterns have been used in BMAs to reduce the search points when finding the best-matched block, such as square patterns in the three-step search (TSS) [24], new three-step search (NTSS) [25], four-step search (4SS) [26], and block-based gradient descent search (BBGDS) [27]; cross patterns in the cross search algorithm (CSA) [28] and 2-D logarithmic search (TDLS) [29]; diamond patterns in diamond search (DS) [30]; and hexagon patterns in hexagon-based search (HEXBS) [31]. More details of BMAs can be found in [32].

In 4-D image compression, motion estimation and compensation have been used for improving CR. In [33], the 3-D wavelet transform is used in combination with 3-D motion estimation and compensation to achieve high-rate compression of time-varying volumes. The 3-D extension of NTSS has been used with the 3-D integer wavelet transformation to compress 4-D images [19]. In [20,21], 2-D motion estimation and compensation techniques of the most recent video coding standard H.264/AVC are adopted for the lossless compression of 4-D medical images. In this paper, a new 3-D block matching motion estimation method named the cross cube search (CCS) is proposed. In this approach, we also introduce a new block distortion measure named variance of residual (VOR) that can enhance the efficiency of the 3-D motion compensation step in our compression scheme. The experiments in Section 5 show that (a) our algorithm significantly reduces the computational complexity of the block matching motion estimation, and (b) using the proposed measure (VOR) instead of the widely used mean squared error (MSE) can result in better PSNR performance.

4.1. Novel block distortion measure

As implied from the compression scheme presented in Section 3, the distortion of the final volume after decompressing is mainly dependent on the distortion of the vector quantization step. Therefore, the fidelity of the final data is affected by the residual blocks (or cubes) that are the inputs to the vector quantizer. The residual blocks, which are the outputs of the motion estimation and compensation steps, are, in general, obtained from the block matching algorithm and the block distortion measure. In motion estimation for video coding, the most widely used block

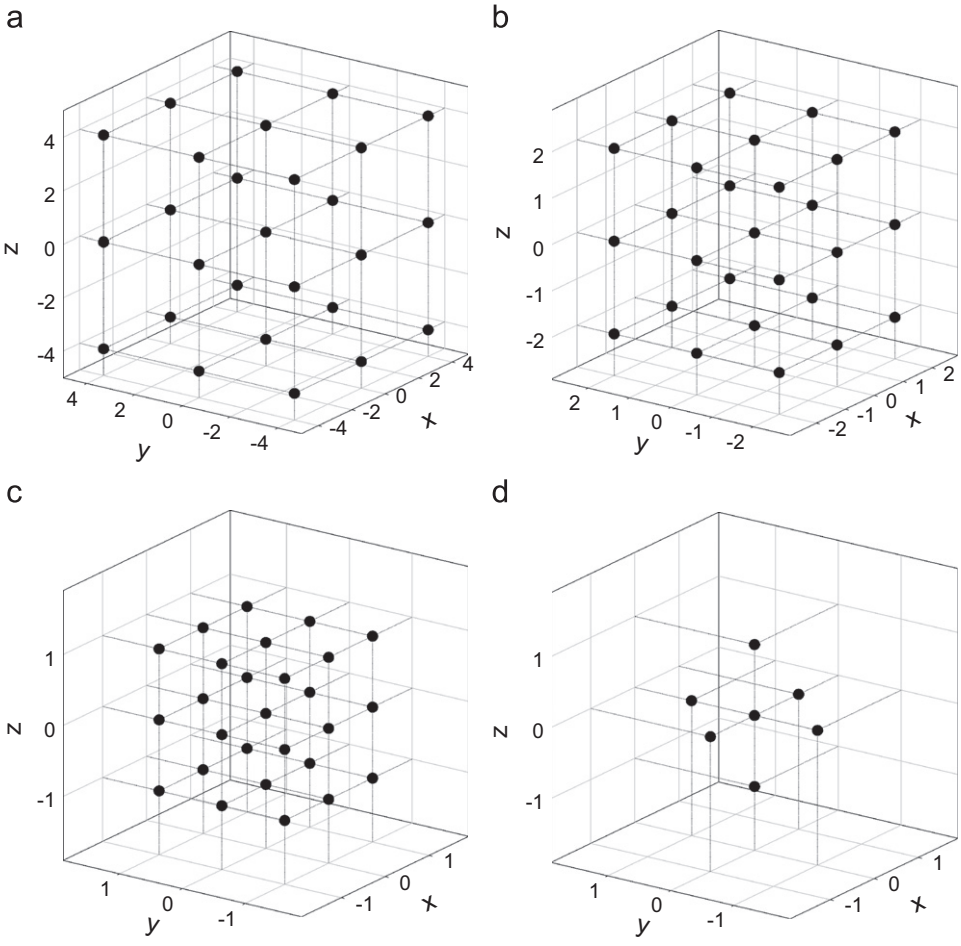

Fig. 5. CCS search patterns: (a) LCSP; (b) MCSP; (c) SCSP; and (d) CSP.

distortion measure is the MSE. However, this measure may not be the most suitable one for the proposed system because a small value of MSE calculated from the block matching step does not guarantee a small distortion of the vector quantization in the next step. We introduced a new distortion measure to obtain a smaller distortion of the vector quantization. The proposed measure is the variance value of all voxel intensities in the residual cube. The VOR for an $L \times M \times N$-sample block is defined by

$$\mathrm{VOR} = \frac{1}{LMN}\sum_{i=0}^{L-1}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1}(P_{ijk}-Q_{ijk})^2-\mu^2 \quad (1)$$

where P_{ijk} is a sample of the current block, Q_{ijk} is a sample of the reference area, and μ, the mean of all voxel intensities in the residual block, is calculated by

$$\mu = \frac{1}{LML}\sum_{i=0}^{L-1}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1}(P_{ijk}-Q_{ijk}). \quad (2)$$

VOR is clearly a measure of the homogeneity of the residual block. A small value of VOR means that the corresponding residual block has small changes in voxel intensity. This results in the nearly identical elements of the input vector of the quantizer. Quantizing multi-dimensional vectors in which the vector elements are nearly identical often produces a small distortion. Other widely used measures, e.g. MSE, are not suitable to this compression scheme since they do not guarantee the small changes in voxel intensity in residual blocks. This is demonstrated in Table 2 and Fig. 11 in Section 5.

4.2. Novel 3-D motion estimation algorithms

To study the characteristics of motion vector probability (MVP) distribution, we applied motion estimation using FS on four 16-bit 4-D medical image datasets, described in Section 5, to obtain the optimal motion vectors. VOR is employed as the block distortion measure with block size of $4 \times 4 \times 4$ and the search window size of $15 \times 15 \times 15$ voxels. This means that each component of the motion vector ranges from -7 to $+7$. Fig. 4 depicts the 3-D graphs of the MVP distribution corresponding to difference values of the z-component, d_z. The graphs associated with $|d_z| > 2$ are not shown since the MVPs are relatively small. As seen in Fig. 4, the motion vectors are center-biased. Furthermore, the probability of the motion vector in the axis-aligned directions is larger than the other directions. Based on this cross center-biased characteristic of the MVP distribution, the CCS algorithm uses four search patterns: large cube search pattern (LCSP), medium cube search pattern (MCSP), small cube search pattern (SCSP) and cross search pattern (CSP) (Fig. 5).

The CCS algorithm comprises the following steps:

- *Step 1:* Apply LCSP and CSP at the center of the search window. Evaluate the matching distortion of all search points in these

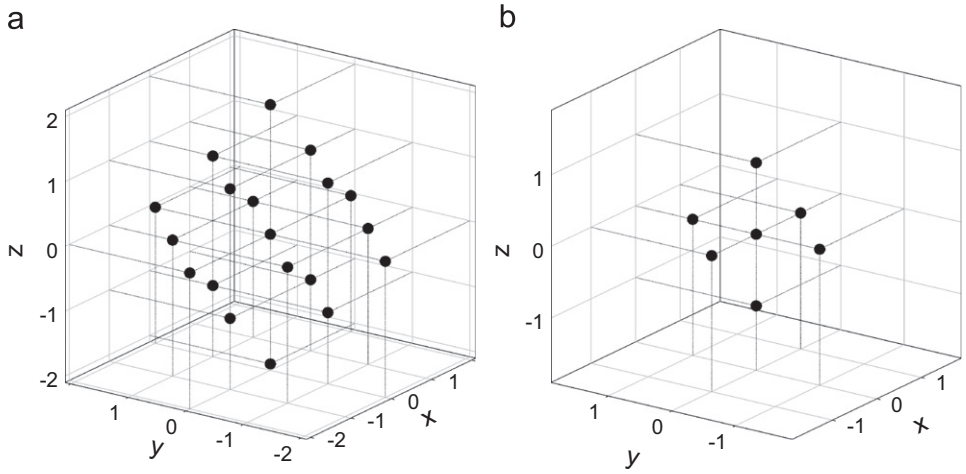

Fig. 6. OS search patterns: (a) LOSP and (b) SOSP.

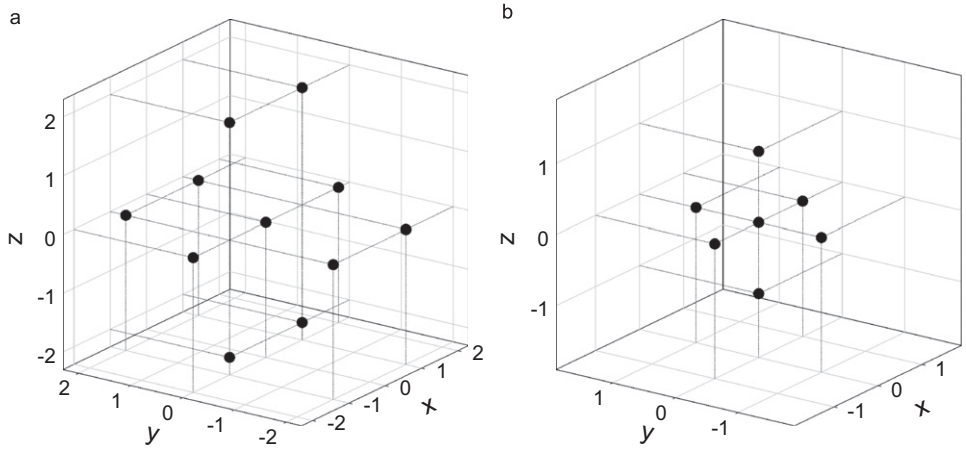

Fig. 7. 3-D HEXSB search patterns: (a) large pattern and (b) small pattern.

two search patterns. If the minimum matching distortion point (MMDP) occurs at the center, the search process stops and the motion vector is found at the center. Otherwise, put the center of the search pattern on the current MMDP and go to Step 2a if MMDP is found in LCSP, or Step 2b if MMDP is found in CSP.

- *Step 2a:* Use MCSP to find the new MMDP. Move the center of the search window to this MMDP, and then go to Step 3.
- *Step 2b:* Repeat using CSP to find the new MMDP until the MMDP occurs at the center point of the search pattern. The search process stops when the motion vector is found at the last MMDP.
- *Step 3:* Apply SCSP to find the new MMDP which is the position of the motion vector.

In addition, two well-known block-matching algorithms in video coding, DS and HEXBS, are extended from 2-D to 3-D for comparing their performance with the proposed method. These two algorithms are based on the MVP distribution and are noted for their computational efficiency. The 3-D extension of the DS algorithm, which is called octahedron search (OS), uses two search patterns (Fig. 6). The large octahedron search pattern (LOSP) contains 19 check points while the small octahedron search pattern (SOSP) comprises 7 check points. During the search process, LOSP is repeatedly used until MMDP occurs at the center point. The search pattern is then switched from LOSP to SOSP. The motion vector is formed based on the position of the point yielding the minimum matching distortion among the seven check points in SOSP.

3-D HEXBS, the 3-D version of HEXBS, uses the two search patterns shown in Fig. 7. The large search pattern (LSP) consists of 11 check points, while the small search pattern (SSP) is actually SOSP in OS. In the first step, LSP is used to find the MMDP. If the MMDP is not located at the center point of LSP, the search process is repeated by centering LSP at the position of the MMDP. Otherwise, it goes to the last step, where SSP is applied to find the last MMDP, which is the position of the motion vector.

5. Experiments

In this section, we describe the experiments used to evaluate the performance of the proposed method, which we term the enhanced Schneider and Westermann method with motion compensation (ESW-MC). The test data comprise one magnetic resonance (MR) 8-bit and three 16-bit time-varying volume datasets which were acquired at the National University Hospital, Singapore, and one 12-bit computed tomography (CT) time-varying volume dataset from the University Hospital of Geneva, Switzerland (Table 1).

The raw image data and their descriptions are extracted from the DICOM datasets to form the input data of the system. In the encoding process, the number of bytes allocated for representing a cube depends on the bit-width of the dataset, the type of the cube and the existence of motion estimation in the corresponding frame. The size of the search window in our experiments is $15 \times 15 \times 15$ voxels. Each component of a motion vector ranges from -7 to $+7$ and is represented as a signed 4-bit integer, forming the 12-bit motion vector. For Type 1 cubes, their mean values are represented by 8-, 12-, or 16-bit voxels corresponding to dataset bit-widths of 8, 12, or 16, respectively. For Type 2 cubes, the sizes of the two codebooks used to quantize level 1 and level 2 data depend on the dataset bit-width and whether motion estimation is applied in the corresponding frame. Fig. 8 summarizes the representation format of a cube. The codebook sizes and the cube representation format are chosen so that the bits allocated to represent a cube are byte-aligned. This leads to a fast and relatively straight forward implementation of data decompression. Other possible configurations may lead to the

Table 1
Dataset specifications.

Dataset	BREAST	AORTA	ABDOMEN	THORAX	HEART
Bits allocated	8	16	16	16	12
Rows × columns	256 × 256	512 × 512	512 × 512	512 × 512	512 × 512
Slices	26	160	120	136	188
Time steps	5	14	12	16	20
Pixel size (mm)	1.25 × 1.25	0.9375 × 0.9375	0.4688 × 0.4688	0.6445 × 0.6445	0.4883 × 0.4883
Inter-slice spacing (mm)	4.2	1.1	0.8	0.9	0.75
Size (MB)	8	1120	720	1088	1880
Modality	MRI	MRA	MRA	MRA	CT

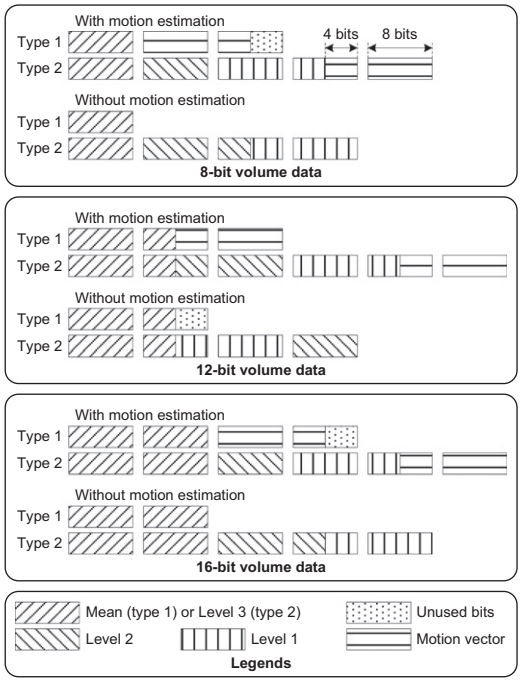

Fig. 8. Bits allocated to represent a cube.

variation in CR, fidelity and decompression time of the compressed data. However, the analysis on alternate configuration is not within the scope of this paper.

Five experiments were performed to evaluate the efficiency of the method. The computing platform was a 2.5 GHz Intel Core 2 Duo laptop equipped with 4 GB RAM. Each test was run at least three times and the execution times averaged. The fidelity of the decompressed data was measured by their PSNRs. The refinement step in the vector quantization process was included only in the last experiment so that we could evaluate its effect on algorithm performance separately from the use of motion compensation and cube classification.

Experiment 1. The objective was to compare our method with the SW method. Since the latter is designed for 8-bit data, we used the BREAST dataset. The dataset was encoded using four compression schemes: (1) SW and (2) SW-CSS, which refer to the SW method without and with motion compensation using CSS, respectively; and (3) ESW and (4) ESW-CSS, which refer to our compression method without and with motion compensation using CSS, respectively. In addition, the vector quantization step in schemes (1) and (2) used two 8-bit codebooks and a 5-iteration refinement step to encode the data. The VOR measurement is used in the motion estimation process in schemes (2) and (4). The results of this experiment are presented in Figs. 9 and 10.

Experiment 2. The aim was to demonstrate of the effectiveness of using motion compensation in the proposed compression scheme and the effectiveness of the new block distortion measure, VOR. The datasets used were AORTA, ABDOMEN, THORAX, and HEART. This experiment comprised four compression schemes: (1) ESW and (2) ESW-CSS, which are the corresponding compressionschemes in Experiment 1; and (3) ESW-FS(MSE) and (4) ESW-FS(VOR), which refer to the use of motion estimation and compensation using FS with MSE and VOR, respectively. Only the first frame in each dataset was chosen as the key frame and no threshold was applied in the 3-D compression phase, meaning that all the partitioned

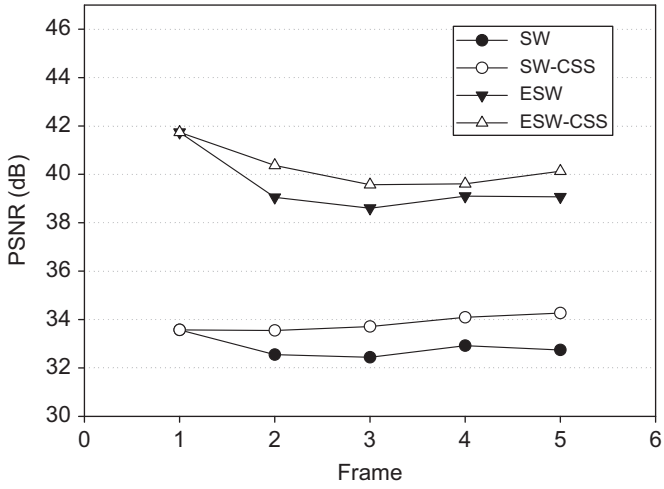

Fig. 9. Test result on BREAST dataset. The compression ratios of the respective methods are 21.33, 14.22, 16.00, and 12.80. The corresponding average processing times per frame are 8, 9, 14, and 15 s, respectively.

cubes were Type 2 cubes. The results of this experiment are shown in Table 2 and Fig. 11.

Experiment 3. This experiment aimed to compare the three proposed 3-D motion estimation algorithms, CCS, OS, and 3-D HEXBS, with FS and the 3-D NTSS method [19] in terms of speed and fidelity of the decompressed data. The new block distortion measure, VOR, was used in all these motion estimation methods. The results of this experiment on our 12- and 16-bit datasets are presented in Table 3.

Experiment 4. Our 3-D compression algorithm uses a threshold to classify a partitioned cube into two types associated with two different coding processes. This experiment analyzes the effect of this threshold value on the processing speed, the CR, and the fidelity of the compressed data using our 12- and 16-bit datasets. Since the homogeneity of a key frame is different from that of an intermediate frame, there should be two different threshold values, θ_1 for key frames and θ_2 for intermediate frames; thus, two tests were executed. In the first test, for each value of θ_1, the first four frames of each dataset were encoded using our 3-D compression algorithm without motion estimation (ESW) and the average PSNR was calculated. The results of this test are shown in Fig. 12. The value of θ_1 yielding the highest average PSNR was chosen for the second test when compressing the first frame as a key frame and the next three frames as intermediate frames. These intermediate frames were encoded using CCS motion estimation with the varying threshold value θ_2. The processing time, CR and PSNR of these three frames were accumulated and averaged. Fig. 13 presents the results of the second test.

Experiment 5. The last experiment was performed to evaluate the use of the refinement step in vector quantization in our method. Several tests without refinement and with the number of refinement iterations ranging from 1 to 3 were executed. In these tests, θ_1 and θ_2 were set at 100 and 300, respectively. The results of this experiment are shown in Fig. 14.

6. Results and discussion

6.1. Experiment 1

In the first experiment, the first frame was selected as a key frame for compression schemes using motion estimation. Fig. 9

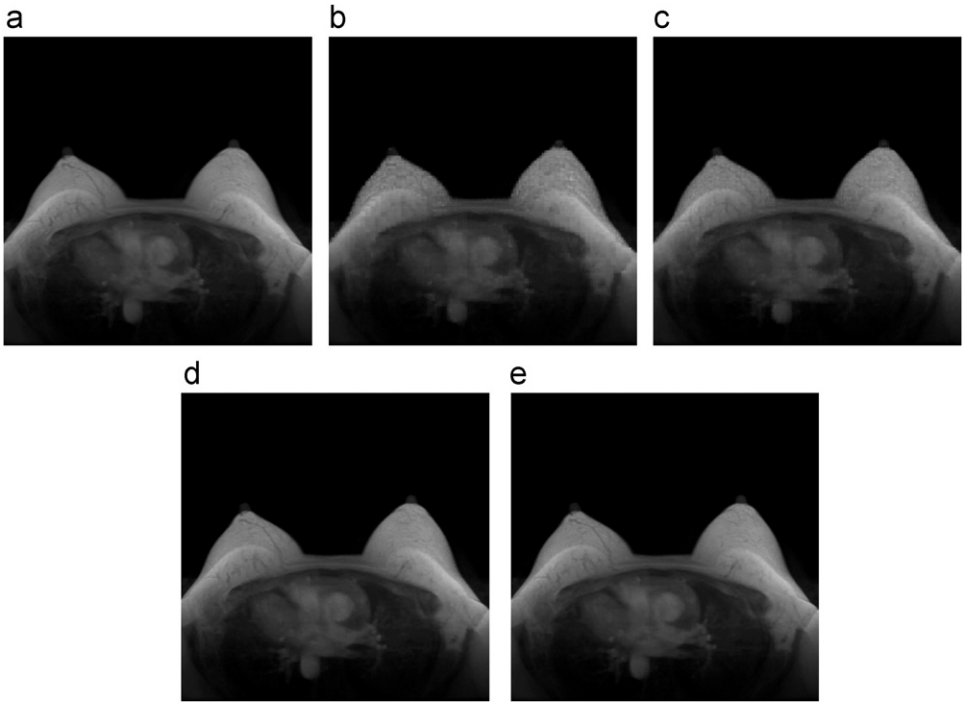

Fig. 10. 2-D texture mapping rendering of the second frame in the BREAST dataset encoded using different methods: (a) Rendered image of the original volume; (b) SW; (c) SW-CSS; (d) ESW and (e) ESW-CSS.

Table 2
Experiment 2 results: compression ratio, processing time, and average PSNR.

Dataset	Method	Ratio	Time (s)	Avg. PSNR (dB)
AORTA	ESW	23.58	48	62.42
	ESW-CSS	18.66	63	63.97
	ESW-FS(MSE)	18.66	793	62.72
	ESW-FS(VOR)	18.66	889	64.31
ABDOMEN	ESW	20.24	47	60.34
	ESW-CSS	17.95	51	61.80
	ESW-FS(MSE)	17.95	594	60.35
	ESW-FS(VOR)	17.95	656	61.94
THORAX	ESW	22.22	49	61.17
	ESW-CSS	18.25	64	62.17
	ESW-FS(MSE)	18.25	643	61.03
	ESW-FS(VOR)	18.25	767	62.14
HEART	ESW	26.60	62	40.68
	ESW-CSS	18.97	84	44.35
	ESW-FS(MSE)	18.97	907	44.14
	ESW-FS(VOR)	18.97	1157	44.68

shows the line graph of the PSNR values between the encoded frame and the corresponding original raw frame.

Due to the relatively small size of the BREAST dataset, the compressed file size used to calculate CR in this experiment did not include the sizes of the two codebooks and other information in the file header. The results show that widening the bit-width of the codebook index from 8 bits in SW to 12 bits in ESW significantly improved the fidelity of the encoded frame (by 6.1–8.2 dB) with a slight trade-off in processing speed and CR. Furthermore, the motion compensation process also appreciably contributed to the quality improvement of each compressed frame (an increase of up to 1.5 dB for SW and 1.3 dB for ESW). This is clearly seen in Fig. 10, which presents the rendered images of the second frame in the BREAST dataset with different compression schemes.

6.2. Experiment 2

From Fig. 11, it is clear that using motion estimation and compensation with the new block distortion measure, VOR, can significantly improve the fidelity of the encoded frame. Using VOR with FS, the improvement in PSNR ranged from 0.97 to 4.00 dB, depending on the dataset (Table 2). However, using FS dramatically slowed down the processing speed (from about 14 to 18 times). It took about 8–19 min to encode a frame using motion estimation with FS. In this case, replacing FS with CSS was a better choice since it still produced a large improvement in fidelity (1.00–3.67 dB) with a slight trade-off in speed (only 1.08–1.35 times slower). Using MSE as the block distortion measure was not suitable for the proposed compression scheme since fidelity was not improved even though the FS algorithm was used.

6.3. Experiment 3

It can be seen from Table 3 that the proposed 3-D motion estimation algorithm CCS produced a fidelity almost equivalent to FS. However, CSS ran faster than FS by a factor of almost 80. CSS was approximately 1.5 times faster while still maintaining a higher PSNR than 3-D NTSS. In addition, OS and 3-D HEXBS, our extensions of the DS and HEXBS algorithms, are good choices for fast motion estimation process since they could run from 2 to 3 times faster with hardly noticeable fidelity loss compared to 3-D NTSS.

6.4. Experiment 4

As shown in Fig. 12, when the threshold value θ_1 increased, the average CR noticeably increased due to the increase of Type 1 cubes in each dataset. The average PSNR was almost unchanged for $\theta_1 \leq 500$ before decreasing. This trend is also observed in Fig. 13 with the variation of θ_2. When the threshold increased, in addition to the improvement of CR, the encoding and decoding process should be faster since the number of Type 1 cubes

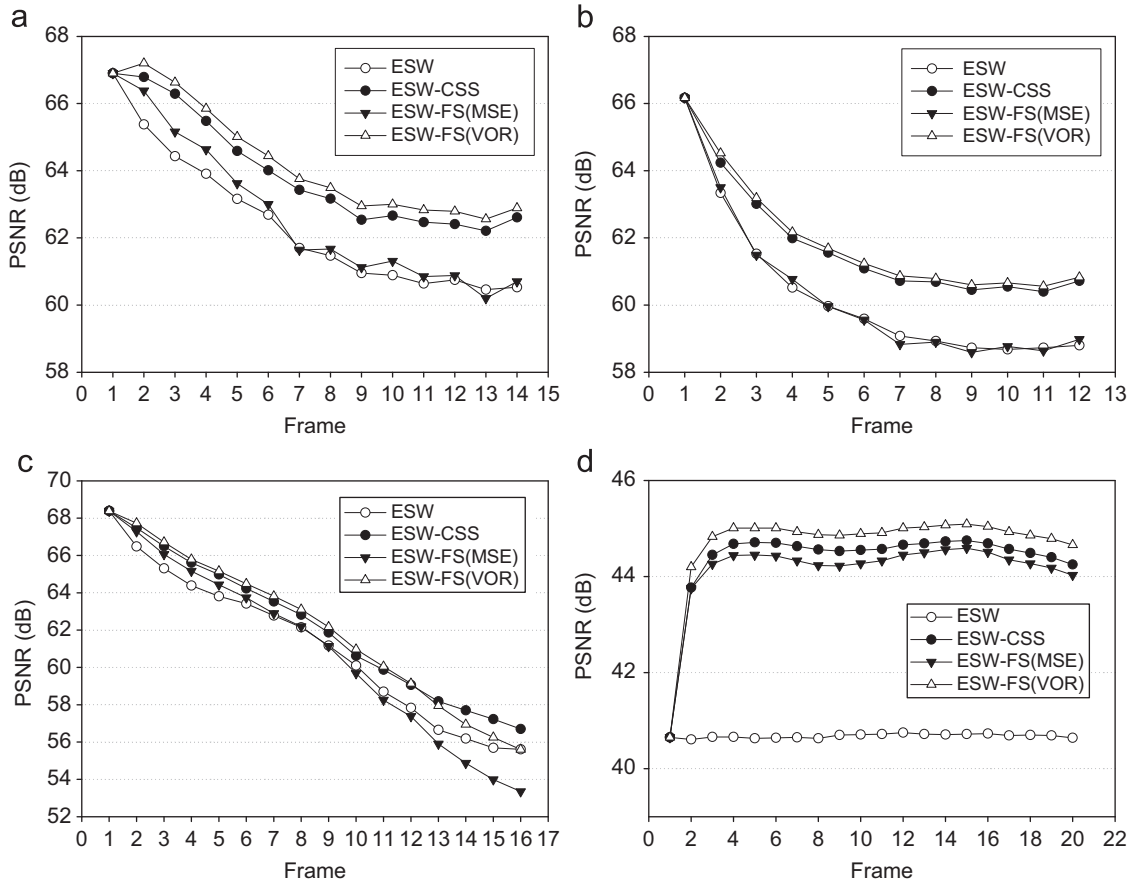

Fig. 11. Variation in PSNR *versus* time step when applying the four compression schemes on various datasets. (a) AORTA; (b) ABDOMEN; (c) THORAX; and (d) HEART.

Table 3
Comparison of motion estimation methods.

Dataset	Method	Avg. number of search points	Speed-up	Avg. PSNR (dB)
AORTA	FS	3194.89	1	64.31
	3-D NTSS	60.67	52.66	63.94
	3-D HEXBS	19.57	163.29	63.59
	OS	31.57	101.19	63.74
	CCS	43.42	73.58	63.97
ABDOMEN	FS	3158.16	1	61.94
	3-D NTSS	59.18	53.36	61.75
	3-D HEXBS	18.59	169.88	61.56
	OS	29.83	105.86	61.61
	CCS	40.76	77.49	61.80
THORAX	FS	3175.44	1	62.14
	3-D NTSS	60.06	52.87	62.12
	3-D HEXBS	19.10	166.29	62.05
	OS	30.69	103.46	62.14
	CCS	42.47	74.77	62.17
HEART	FS	3211.29	1	44.68
	3-D NTSS	59.08	54.35	44.32
	3-D HEXBS	18.58	172.84	44.07
	OS	29.37	109.34	44.10
	CCS	41.41	77.55	44.35

increased (and processing Type 1 cubes is faster than processing Type 2). Particularly, when $\theta_1 = 100$, the encoding process for key frames was 12.06–24.64% faster with the largest increment from 19.65:1 to 23.36:1 in CR and the same PSNR compared to the case when the threshold was not set. When θ_2 was set at 300, the fidelity was still optimal, but the encoding speed increased by up to 22.82% and CR also increased with the largest change from 17.39:1 to 19.34:1. Ignoring the time taken for loading data to the memory and storing data, it took about 770 ms for decompressing one key frame of AORTA, our largest 16-bit dataset, when θ_1 was 0. When θ_1 was set at 200 and 500, the average decoding time was 670 and 630 ms, respectively. Decompressing intermediate frames was approximately 1.6 times slower since motion compensation needed to be executed. This is significantly faster than that of other compression methods.

6.5. Experiment 5

From Fig. 14, we see that the fidelity of the encoded data increased considerably when the refinement step was included. The most significant increase in fidelity occurred after the first refinement iteration, with average PSNR increasing from about 0.24–0.80 dB. From the second iteration onwards, the increment sharply decreased and stabilised after several iterations. The average refinement processing times for one frame in the AORTA, ABDOMEN, THORAX, and HEART datasets were 128, 135, 136, and 260 s respectively. It took from 2 to 3 times longer than the time spent to encode one frame. The number of iterations to be used with the refinement step is based on the dataset size and the expected processing time. In most cases, encoding a dataset using the 1-iteration refinement step is a good choice.

6.6. Discussion

Since level 3 data can be considered shrank representation of the original volume, redundancy would be found in the resulting bit stream. Therefore, an extra gain to CR can be achieved by

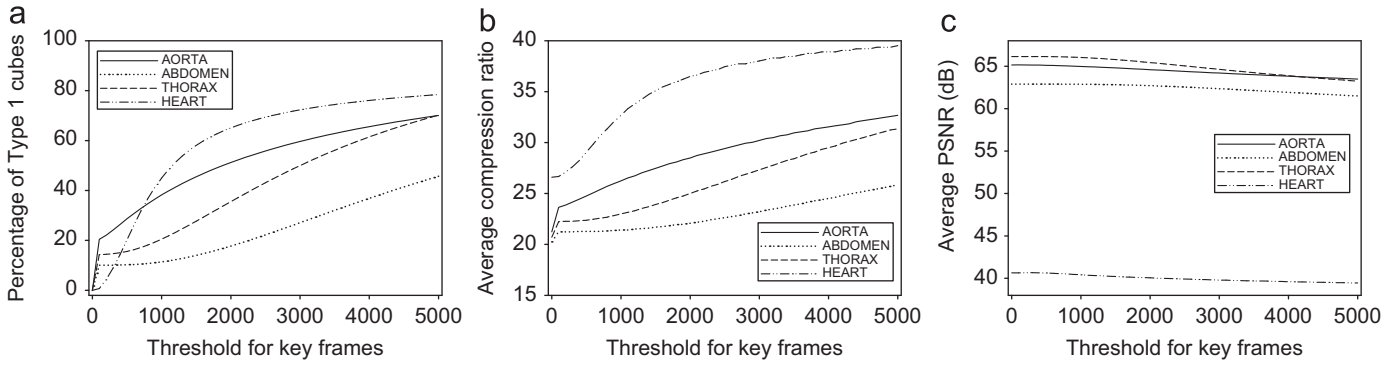

Fig. 12. Effect of threshold value θ_1 on the average of: (a) percentage of type 1-cubes; (b) compression ratio; and (c) average PSNR of the first four key frames in our 12- and 16-bit datasets.

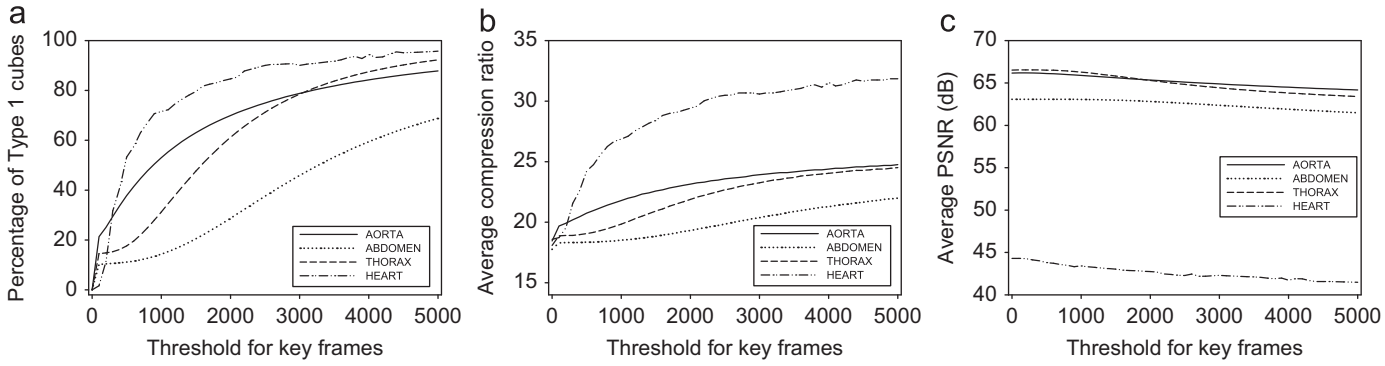

Fig. 13. Effect of threshold value θ_2 on the average of: (a) percentage of type 1-cubes; (b) compression ratio; and (c) average PSNR of the first 3 intermediate frames in our 12- and 16-bit datasets. The threshold for the first (key) frame was set at 100 for all the datasets.

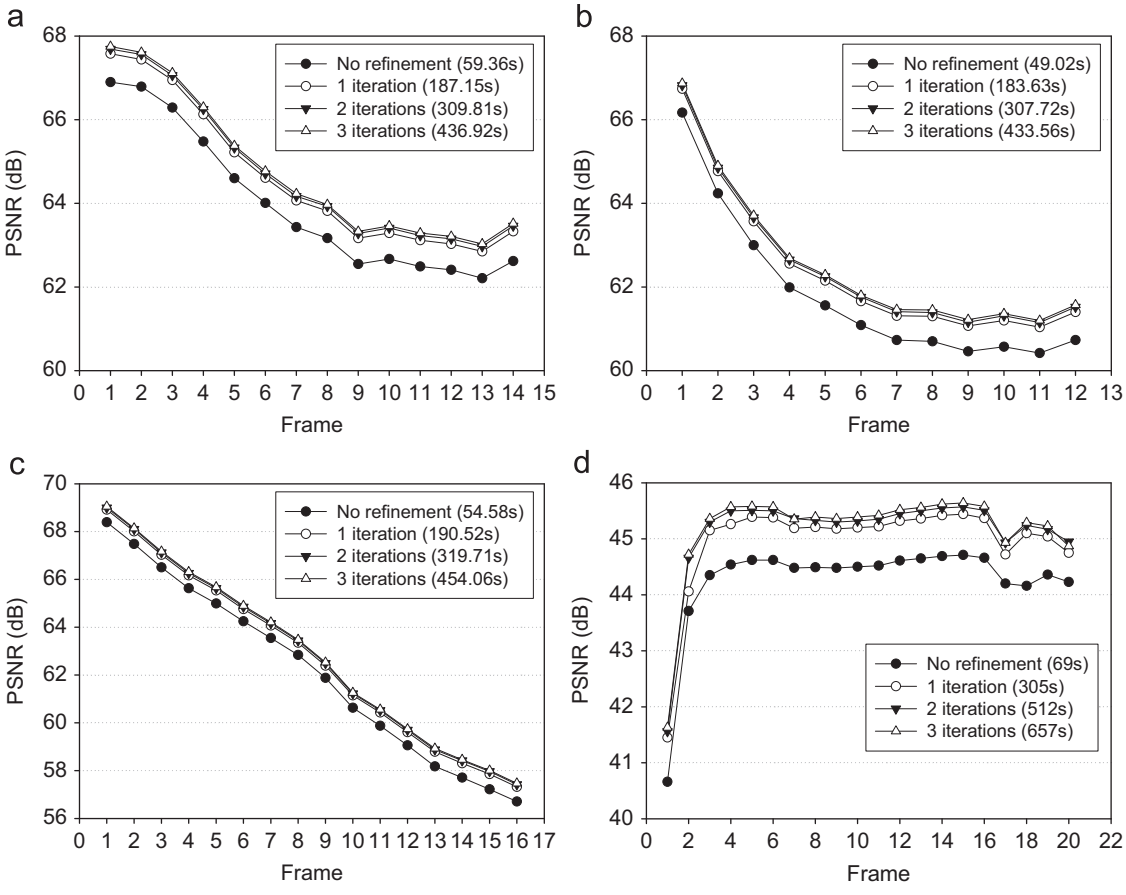

Fig. 14. Variation in PSNR *versus* time step when applying the different number of refinement iterations on: (a) AORTA; (b) THORAX; (c) ABDOMEN; and (d) HEART dataset. The average encoding times per frame are indicated in the legend.

Table 4
Comparison of lossy 4-D medical image compression systems.

Method	Computing platform	Typical dataset	Performance	Processing time
ESW-MC	2.5 GHz Intel Core 2 Duo, 4GB RAM	$512 \times 512 \times 160 \times 14$, 16-bit MRA, 1.1 GB (AORTA)	PSNR = 64–65 dB (CR = 18–32)	Encoding: 60 s/frame, decoding (no I/O access): 670–900 ms/frame
4-D DWT and 4-D EZW [14]	Pentium III 550 MHz	$240 \times 224 \times 32 \times 32$, float, 210 MB	PSNR = 37.87 dB (CR = 32) PSNR = 37.49 dB (CR = 64)	Encoding: 266.1 s, decoding: 230.5 s
4-D JPEG2000 [15]	Not reported	$64 \times 64 \times 21 \times 1000$, 13-bit fMRI, 16 MB (mb01)	PSNR ≈ 57.5 dB at 0.5 bpp (CR = 26) PSNR ≈ 60.0 dB at 1.0 bpp (CR = 13)	Not reported
3-D JPEG2000 (xyt) [15]	Not reported	$64 \times 64 \times 21 \times 1000$, 13-bit fMRI, 16 MB (mb01)	PSNR ≈ 57.0 dB at 0.5 bpp (CR = 26) PSNR ≈ 59.5 dB at 1.0 bpp (CR = 13)	Not reported
3-D JPEG2000 (xyz) [15]	Not reported	$64 \times 64 \times 21 \times 1000$, 13-bit fMRI, 16 MB (mb01)	PSNR ≈ 52.0 dB at 1.0 bpp (CR = 13)	Not reported
4-D SPIHT [16]	Not reported	$64 \times 64 \times 21 \times 1000$, 13-bit fMRI, 16 MB (mb01)	PSNR ≈ 57 dB at 0.5 bpp (CR = 26) PSNR ≈ 59 dB at 1.0 bpp (CR = 13)	Not reported
4-D SBHP [17]	Not reported	$64 \times 64 \times 21 \times 1000$, 13-bit fMRI, 16 MB (mb01)	SNR ≈ 22 dB at 0.5 bpp (CR = 26) SNR ≈ 27 dB at 1.0 bpp (CR = 13)	Not reported
3-D IWT and 3-D NTSS[19]	Pentium 4 2.4GHz	$128 \times 128 \times 107 \times 145$, 16-bit CT, 50 MB (Sequence A)	PSNR = 40–42 dB at 1.0 bpp (CR = 16)	8 s for encoding-decoding each frame

bpp: bits per pixel.

applying any lossless data compression method to the resulting bit stream. In order to analyze this, all the compressed frames using ESW-CSS in Experiment 2 were encoded using the arithmetic coding method introduced in [34]. The new average CR obtained for the AORTA, ABDOMEN, THORAX, and HEART datasets were 23.33, 22.26, 22.37, and 23.65, respectively. The average of extra compression and decompression time due to the arithmetic coding step ranged from 100 to 200 ms per frame, depending on the dataset. These overheads are relatively small.

Table 4 lists a cursory comparison of the proposed scheme with existing 4-D lossy compression methods. It should be noted that CR, image fidelity and processing time are not measured under the same conditions. However, from the experimental results and the information from Table 4, it is seen that good image fidelity and fast decompression are two advantages of the proposed method. While the CR of our method is not the best compared to that of methods based on wavelet transform, e.g. 3-D JPEG2000, it can be considered to be at a high level. At the same level of CR, our method may achieve a comparable or even better PSNR. Especially, the method offers a very fast decompression speed thanks to the simple decoding. Therefore, the proposed method is suitable for numerous applications, particularly time-critical applications dealing with massive data.

The proposed method has two other interesting features. First, the classification of the partitioned cubes into two types can increase CR and the encoding and decoding speed with a slight loss in fidelity. Currently, this process is automatically done using the two thresholds θ_1 and θ_2. However, we can manually classify the cubes based on their homogeneity and input regions of interest (ROIs). Cubes with low homogeneity belonging to ROIs should be encoded using HVQ. Each of the remaining cubes containing less important information or having high homogeneity is represented by the mean value of the its voxels. This strategy can significantly improve CR and processing speed while preserving information of interest. Second, this method is a block-based coding method. In order to decode one block, we only need the two codebooks of the corresponding frame and the content of one block in the previous frame together with a motion vector in the event that the corresponding frame is an intermediate frame, and that the previous frame has already been decoded. Therefore, we can decompress any slice or any block in one frame without decoding the frame. This will be useful when we wish to obtain an overview of an encoded dataset.

The proposed method has two disadvantages. First, due to the complexity of vector quantization based methods, the proposed scheme may require a relatively long time for compressing a large dataset. Nevertheless, this drawback should not be an issue since the encoding phase is to be done only once, and using a powerful computer will help to hasten the process. Second, although using motion compensation in the proposed method can significantly improve the fidelity of the encoded data, it slows down the decoding process in the random frame access mode. This is due to the fact that to decompress one frame, a number of previous frames from the nearest key frame have to be decoded if their contents are not ready. However, this drawback does not appear in the data browsing mode. This limitation can also be reduced by setting a small number of intermediate frames to be encoded in one group and using cache memory to store neighboring frames which have already been decoded. Another possible solution is to use parallel processing to concurrentize the data preparation and data decompression processes [35].

7. Conclusion

We have proposed an efficient compression scheme, which can be applied to numerous applications, particularly the compression

of 4-D medical images. The scheme uses 3-D motion estimation to create a homogenous preprocessed data to be effectively compressed by a 3-D image compression algorithm using hierarchical vector quantization. A new block distortion measure, the variance of residual, and three 3-D fast block matching algorithms are developed for improving the motion estimation process in term of speed and homogeneity of the preprocessed data. The 3-D image compression algorithm is designed to be suitable with the input data using two different encoding techniques: representing high homogeneity partitioned cubes by the mean values of their voxels, and using hierarchical vector quantization to compress homogenous cubes. These two techniques can increase the processing speed and CR while maintaining the fidelity of the compressed data. The experimental results on typical 4-D medical images showed that our method can achieve a higher fidelity and faster decompression time compared to other lossy compression methods producing similar CRs. The combination of motion compensation and hierarchical vector quantization is the key factor for the good performance.

Conflict of interest statement

None declared.

Acknowledgments

This work is supported in parts by a research grant from National University of Singapore (R-265-000-270-112 and R-265-000-270-133).

References

[1] H. Lee, Y. Kim, A.H. Rowberg, E.A. Riskin, Statistical distributions of DCT coefficients and their application to an interframe compression algorithm for 3-D medical images, IEEE Transactions on Medical Imaging 12 (3) (1993) 478–485.
[2] N. Mohsenian, A. Nosratinia, B. Liu, M.T. Orchard, Adaptive entropy constrained transform coding of magnetic resonance image sequences, IEEE Transactions on Nuclear Science 42 (6) (1995) 2309–2316.
[3] H. Lee, Y. Kim, E.A. Riskin, A.H. Rowberg, M.S. Frank, A predictive classified vector quantizer and its subjective quality evaluation for X-ray CT images, IEEE Transactions on Medical Imaging 14 (2) (1995) 397–406.
[4] Y.-G. Wu, S.-C. Tai, Medical image compression by discrete cosine transform spectral similarity strategy, IEEE Transactions on Information Technology in Biomedicine 5 (3) (2001) 236–243.
[5] M.A. Pratt, C.H. Chu, S. Wong, Volume compression of MRI data using zerotrees of wavelet coefficients, in: Proceedings of SPIE, Wavelet Applications in Signal and Image Processing IV, vol. 2825, Denver, CO, USA, 1996, pp. 752–763.
[6] G. Menegaz, J.-P. Thiran, Three-dimensional encoding/two-dimensional decoding of medical data, IEEE Transactions on Medical Imaging 22 (3) (2003) 424–440.
[7] P. Schelkens, A. Munteanu, J. Barbarien, M. Galca, X. Giro-Nieto, J. Cornelis, Wavelet coding of volumetric medical datasets, IEEE Transactions on Medical Imaging 22 (3) (2003) 441–458.
[8] Z. Xiong, X. Wu, S. Cheng, J. Hua, Lossy-to-lossless compression of medical volumetric data using three-dimensional integer wavelet transforms, IEEE Transactions on Medical Imaging 22 (3) (2003) 459–470.
[9] S.-G. Miaou, S.-T. Chen, Automatic quality control for wavelet-based compression of volumetric medical images using distortion-constrained adaptive vector quantization, IEEE Transactions on Medical Imaging 23 (11) (2004) 1417–1429.
[10] X. Wu, T. Qiu, Wavelet, coding of volumetric medical images for high throughput and operability, IEEE Transactions on Medical Imaging 24 (6) (2005) 719–727.
[11] J.M. Shapiro, Embedded image coding using zerotrees of wavelet coefficients, IEEE Transactions on Signal Processing 41 (12) (1993) 3445–3462.
[12] A. Said, W.A. Pearlman, A new, fast, and efficient image codec based on set partitioning in hierarchical trees, IEEE Transactions on Circuits and Systems for Video Technology 6 (3) (1996) 243–250.
[13] J. Wilhelms, A.V. Gelder, Multi-dimensional trees for controlled volume rendering and compression, in: Proceedings of the 1994 Symposium on Volume Visualization, VVS '94, 1994.
[14] L. Zeng, C.P.J Jansen, S. Marsch, M. Unser, P.R. Hunziker, Four-dimensional wavelet compression of arbitrarily sized echocardiographic data, IEEE Transactions on Medical Imaging 21 (9) (2002) 1179–1187.
[15] H.G. Lalgudi, A. Bilgin, M.W. Marcellin, A. Tabesh, M.S. Nadar, T.P. Trouard, Four-dimensional compression of fMRI using JPEG2000, in: Proceedings of the SPIE International Symposium on Medical Imaging, 2005, pp. 1028–1037.
[16] H.G. Lalgudi, A. Bilgin, M.W. Marcellin, M.S. Nadar, Compression of fMRI and ultrasound images using 4D SPIHT, in: Proceedings of the 2005 IEEE International Conference on Image Processing, vol. 2 of ICIP 2005, 2005, pp. 746–749.
[17] Y. Liu, W.A. Pearlman, Four-dimensional wavelet compression of 4-D medical images using scalable 4-D SBHP, in: Proceedings of the 2007 Data Compression Conference, DCC '07, 2007, pp. 233–242.
[18] C. Chrysafis, A. Said, A. Drukarev, A. Islam, W.A. Pearlman, SBHP-a low complexity wavelet coder, in: Proceedings of the 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 4 of ICASSP '00, 2000, pp. 2035–2038.
[19] A.A. Kassim, P. Yan, W.S. Lee, K. Sengupta, Motion compensated lossy-to-lossless compression of 4-D medical images using integer wavelet transforms, IEEE Transactions on Information Technology in Biomedicine 9 (1) (2005) 132–138.
[20] V. Sanchez, P. Nasiopoulos, R. Abugharbieh, Efficient lossless compression of 4-D medical images based on the advanced video coding scheme, IEEE Transactions on Information Technology in Biomedicine 12 (4) (2008) 442–446.
[21] V. Sanchez, P. Nasiopoulos, R. Abugharbieh, Novel lossless fMRI image compression based on motion compensation and customized entropy coding, IEEE Transactions on Information Technology in Biomedicine 13 (4) (2009) 645–655.
[22] J. Schneider, R. Westermann, Compression domain volume rendering, in: Proceedings of the 2003 IEEE International Conference on Visualization, VIS '03, 2003, pp. 293–300.
[23] Y. Linde, A. Buzo, R.M. Gray, An algorithm for vector quantizer design, IEEE Transactions on Communications 28 (1) (1980) 84–95.
[24] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, T. Ishiguro, Motion compensated interframe coding for video conferencing, in: Proceedings of the National Telecommunications Conference, vol. 4 of NTC '81, 1981, pp. G5.3.1–G5.3.5.
[25] R. Li, B. Zeng, M.L. Liou, A new three-step search algorithm for block motion estimation, IEEE Transactions on Circuits and Systems for Video Technology 4 (4) (1994) 438–442.
[26] L.-M. Po, W.-C. Ma, A novel four-step search algorithm for fast block motion estimation, IEEE Transactions on Circuits and Systems for Video Technology 6 (3) (1996) 313–317.
[27] L.-K. Liu, E. Feig, A block-based gradient descent search algorithm for block motion estimation in video coding, IEEE Transactions on Circuits and Systems for Video Technology 6 (4) (1996) 419–422.
[28] M. Ghanbari, The cross-search algorithm for motion estimation, IEEE Transactions on Communications 38 (7) (1990) 950–953.
[29] J.R. Jain, A.K. Jain, Displacement measurement and its application in interframe image coding, IEEE Transactions on Communications 29 (12) (1981) 1799–1808.
[30] S. Zhu, K.-K. Ma, A new diamond search algorithm for fast block matching motion estimation, in: Proceedings of the International Conference on Information, Communications and Signal Processing, vol. 1 of ICICS '97, 1997, pp. 292–296.
[31] C. Zhu, X. Lin, L.-P. Chau, Hexagon-based search pattern for fast block motion estimation, IEEE Transactions on Circuits and Systems for Video Technology 12 (5) (2002) 349–355.
[32] Y.-W. Huang, C.-Y. Chen, C.-H. Tsai, C.-F. Shen, L.-G. Chen, Survey on block matching motion estimation algorithms and architectures with new results, The Journal of VLSI Signal Processing 42 (3) (2006) 297–320.
[33] S. Guthe, W. Straβer, Real-time decompression and visualization of animated volume data, in: Proceedings of the 2001 IEEE International Conference on Visualization, VIS '01, 2001, pp. 349–356.
[34] A. Said, Arithmetic Coding, Lossless Compression Handbook, Academic Press, San Diego, CA, USA, 2003, pp. 101–152 (Chapter 5).
[35] D. Nagayasu, F. Ino, K. Hagihara, A decompression pipeline for accelerating out-of-core volume rendering of time-varying data, Computers & Graphics 32 (3) (2008) 350–362.

Binh P. Nguyen received the B.Eng. (Hons.) in Information Technology and M.Sc. in Information Processing and Communications from Hanoi University of Technology, Vietnam in 2002 and 2004, respectively. He is currently a lecturer at the School of Information and Communication Technology, Hanoi University of Technology, Vietnam and working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. His research interests include visualization of medical images, GPU-based algorithms, multi-core and multi-GPU computing.

Chee-Kong Chui is currently an Assistant Professor in the Control and Mechatronics Group, Department of Mechanical Engineering, National University of

Singapore, Singapore. He was the principal investigator of the Biomedical Simulation & Device Design Project at the then Institute of Bioengineering prior to pursing a Ph.D. in Biomedical Precision Engineering Lab, The University of Tokyo, Japan. His research interests include computer integrated and robot-assisted surgery, human–machine interface, medical device design, biomechanical modeling and simulation.

Sim-Heng Ong is an associate professor in the Department of Electrical and Computer Engineering and the Division of Bioengineering, National University of Singapore. He received his B.E. (Hons.) from the University of Western Australia and his Ph.D. from the University of Sydney. His major fields of interest are computer vision and biomedical image processing. He has over 250 papers published in international journals and conference proceedings.

Stephen Chang is a keen developer of the laparoscopic approach to Hepatobiliary and Pancreatic surgery and eagerly shares his skills with the other surgeons in this region through course and telecommunications. He has since published his experience in several major surgical journals. In addition to his surgical practice, Dr. Chang has several research interests including pancreatic diseases, colorectal metastases, minimally invasive approach to biliary and stone diseases as well as robotic surgery. He is currently holding numerous research grants in developing surgical techniques in this area and is the research director in the division of Hepatobiliary and Pancreatic Surgery.