Pergamon

Computerized Medical Imaging and Graphics

Computerized Medical Imaging and Graphics 22 (1998) 323–339

A comparison of lossless compression methods for medical images

Juha Kivijärvi[a], Tiina Ojala[a], Timo Kaukoranta[a], Attila Kuba[b], László Nyúl[b], Olli Nevalainen[a,]*

[a]Turku Centre for Computer Science (TUCS), Department of Computer Science, University of Turku, Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland
[b]Department of Applied Informatics, József Attila University, Szeged, PO Box 652, H-6701 Szeged, Hungary

Abstract

In this work, lossless grayscale image compression methods are compared on a medical image database. The database contains 10 different types of images with bit rates varying from 8 to 16 bits per pixel. The total number of test images was about 3000, originating from 125 different patient studies. Methods used for compressing the images include seven methods designed for grayscale images and 18 ordinary general-purpose compression programs. Furthermore, four compressed image file formats were used. The results show that the compression ratios strongly depend on the type of the image. The best methods turned out to be TMW, CALIC and JPEG-LS. The analysis step in TMW is very time-consuming. CALIC gives high compression ratios in a reasonable time, whereas JPEG-LS is nearly as effective and very fast. © 1998 Elsevier Science Ltd. All rights reserved

Keywords: Lossless image compression; Medical images; Image databases; PACS

1. Introduction

Digital storing of medical image data is problematic, especially because of the requirement to preserve the best possible image quality. In several countries this has led to the interpretation that medical images should be totally reproducible. The reason for this is the uncertainty in the lossy image compression methods' ability to preserve the relevant information. During the image generation phase it is still not known which details will later turn out to be important, considering the safety of the patients. The lossy image compression techniques, which reproduce completely acceptable decoded images on scenes such as houses or landscapes, do not fulfil the strictest quality requirements needed in medical applications. There is always the possibility that a vague detail might give a reason to suspect some critical changes in a patient's condition. For this reason, the lossy techniques, which tend to give high compression ratios, such as 1:10–1:30, are not acceptable in medical image compression. On the other hand, there are several well-known reasons why savings in the size of the image archives are necessary:

- Digital medical image databases are very large; usually many images are taken of each patient.
- Even if the number of patients were not radically increasing, the patient databases tend to grow because of the demand for very long storage periods of patient data.
- Individual patient records often consist of large images because of the high resolution and many bits per pixel used.
- The transfer time of digital images depends on the size of the compressed images. The practical usefulness of a picture archiving and communication system (PACS) presupposes that transfer operations must fulfil reasonable time requirements.

In many cases the only possibility is to trust in the lossless image compression methods. Unfortunately, their compression ratios are far from those of lossy methods, thus the algorithm development is severely behind the hopes set on the algorithms.

This research is part of a larger research project [1,2], where the aim is to create a functional hospital information record and search system. Here we try to evaluate the existing lossless compression methods when applied to a sample of the image archive. We measure the compression result by the compression ratio

$$R = \frac{\text{original image size}}{\text{compressed image size}} \qquad (1)$$

Thus, larger values of R indicate a better compression method. It should be noted that the original image size used in the above formula contains no overhead originating from the header information or the byte border alignments.

* Corresponding author. E-mail: olli.nevalainen@utu.fi

Problems in the byte border alignment occur if the number of bits for a pixel is not a multiple of the byte length. (For example, if the pixels are expressed by 12 bits, image formats which do not include any compression generally do not utilize the four last bits of the second byte.) We thus calculate the image size simply by the formula

$$\text{original image size} = \text{Xdimension} * \text{Ydimension} * \text{bits per pixel}$$

On the other hand, the compressed image size includes all the header information needed in the reconstruction of the original image data. However, this does not have a great effect on the compression ratios, except for some very small images (for which we also apply an improved format of storage). Note that the images are originally given in the DICOM format [3], which includes patient data headers. We have discarded these headers because they are irrelevant from the compression point of view.

Fast performance is a desirable property for an algorithm and in some cases a long running time may even rule out the application of a space-efficient compression technique. This is why we measured the running times needed for compressing and decompressing the test images.

This paper is organized as follows. In Section 2 we describe the image material we have used and give some characterizations of the image types. In Section 3 we recall briefly the compression techniques used. These consist of seven specific grayscale image compression techniques, four compressed image formats and 18 general all-purpose compressing techniques. Because the latter group contains numerous freely available programs, we have included only methods that are either known to compress some data very well or are widely used. Section 4 contains the summaries of the experiments and conclusions are drawn in Section 5.

2. Image types

The medical database of the Albert Szent-Györgyi Medical University contains images of five different modalities: computed radiology (CR), computed tomography (CT), magnetic resonance (MR), nuclear medicine (NM) and ultrasound (US). The images originate from 10 types of patient studies. There are altogether 125 patient studies, which contain in total 3147 monochrome images. Table 1 shows a summary of the image types and their properties. We observe that the pixel depths are 8, 12, 13 or 16 bpp and a particular image type may appear with many different resolutions. The heterogeneity of the image material is added by the fact that some of the image types are of constant size whereas some others may appear in several different sizes. Appendix A contains an example of each image type. Three of the image types are small (64 × 64 or 128 × 128) and their compression presupposes special care for certain methods. It turns out that a simple compression technique (such as FELICS) works well for these small images whereas certain advanced techniques (e.g. SPIHT) do not operate for them. One solution is to form a larger image composed of several small images and handle it as a whole, see Table 1, the *merged* image types. A merged image is composed of the images of one study. Thus, the images to be merged are of the same size and type.

Table 1
Image types[a]

Mod	Type	bpp			Resolution					Study count	Image count	Filesize/b	Real size/b
		8	12	16	64^2	128^2	256^2	512^2	Other				
CR	PED CHEST		×						×	4	4	35958856	26969088
CT	ABDOMEN		×	×				×	×	17	237	126134740	100365346
CT	HEAD		×	×				×	×	14	1138	548073169	411762116
MR	ABDOMEN		×	×			×			18	24	3146112	2359296
MR	SPINE		×	×			×	×		14	152	44304888	43155456
NM	BRAIN	×				×				4	480	7871520	7864320
	merged	×							×	*4*	*4*	*7864388*	*7864320*
NM	HB			×	×					*15*	*480*	*3939360*	*3932160*
	merged			×					×	15	16	3932432	3932160
NM	HEART			×	×					*17*	*510*	*4185570*	*4177920*
	merged			×					×	17	17	4178209	4177920
NM	WB	×					×		×	14	84	8258810	8257536
US		×							×	8	38	11405914	11405344
Total										125	3147	793278939	620248582
	merged									*125*	*1714*	*793257518*	*620248582*

[a]The resolutions of the image types are marked by '×'. Image count gives the total number of images in all the studies. Filesize includes the space for the headers and the extra bits due to the byte border alignment; real size consists of the pixel data only. PED CHEST = pediatric chest, HB = heart and body, WB = whole body.

Table 2
Lossless image compression methods[a]

From external sources

Method	Program	Ver.	Author	Year	Status	Restrictions	Notes
BTPC	BTPC	4	John Robinson	1997	free	max 8 bpp	
CALIC	CALIC		Xiaolin Wu	1995	eval		
GIF	PBMPLUS		Jef Poskanzer	1991	free	max 8 bpp	
JPEG-LS	LOCO	0.85	Hewlett-Packard Lab	1997	eval		not optimized
PNG	PNGLIB	0.96	Andreas Dilger	1997	free		main program for the library was written
SPIHT	SPIHT	8.01	Amir Said	1995	eval	$w,h >= 128$	
TMW	TMW	0.51	Bernd Meyer	1997	eval		decoding of 16 bit images not successful

Own implementations

Method	Options	Restrictions
CLIC	(default)	
CPM		$w = h = 2^k$
FELICS	(default)	
LJPEG	5 (predictor)	

[a]Ver. = version number, free = freeware, eval = for test or evaluation purposes, w = image width, h = image height. Default options for lossless compression were used except for PNG, for which the PNG_FILTER_PAETH option was used.

3. Compression methods

3.1. Grayscale image compression methods

Table 2 contains a summary of the grayscale image compression methods used. Some of these methods are tested with freely available programs and others we have implemented according to the algorithm description from the literature. Our own implementations do not include any special code optimizations for the running time; they are written according to good programming style.

CALIC, a Context Based, Adaptive, Lossless Image Codec [4] is a compression technique based on the pixel context of the present pixel to be coded (i.e. the setting of the pixels of some predetermined pattern of neighbour pixels). The method is capable of learning from the errors made in the previous predictions and in this way it can improve its prediction adaptively when the compression proceeds. Pixel values are predicted by a nonlinear predictor, which selects the prediction pixels and the particular prediction function among several possible prediction functions on the basis of the local context. The context is built up of the local gradient magnitudes in horizontal and vertical directions. The accuracy of the prediction is improved by adding an estimate of the error to the predicted value. This estimate is the average error of the previous prediction values in the present context. The context for error estimation is selected in such a way that it models the magnitudes of the local gradient and the two previous error values both in relation to the local texture of the image and the prediction value in the most effective way.

Four coefficients are used to weight the horizontal and vertical gradient magnitudes and the previous prediction errors, when calculating the context for error estimation. The coefficients should be selected on the basis of the training set drawn from the type of images to be compressed. The final set of prediction errors is coded by arithmetic or Huffman coding.

The coefficients used in this study have been optimized for ISO images and it is therefore possible that a case specific optimization could still improve our result. The final coding was performed by the arithmetic code.

The *Central Prediction Method (CPM)* [5] is an iterative compression technique that applies the hierarchical four-bit coding to the image to code the prediction errors. The pixel values are predicted in two phases. In the first phase the value of a pixel is predicted as an average of the four pixels of the vertical and horizontal neighbours. In the second phase half of the remaining pixels are predicted by the pixel values in the two diagonal directions. A quarter of the pixel values remains unpredicted and will be moved to a separate block, which will then be compressed recursively by the same prediction method. Fig. 1 illustrates this.

The recursion terminates when the size of the block of original pixel values is 2×2. The bit levels of the resulting image are coded by the hierarchical four-bit coding scheme. The coding scheme divides the bitplane of size $N \times N$ into four subplanes of size $N/2 \times N/2$, each of which is coded by

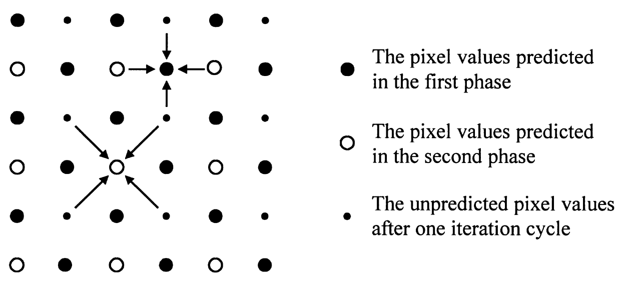

● The pixel values predicted in the first phase
○ The pixel values predicted in the second phase
• The unpredicted pixel values after one iteration cycle

Fig. 1. Prediction scheme of CPM.

a bit in a four-bit codeword. If all the pixel values in a subplane are zero, the subplane is coded by a zero. Otherwise it is coded by a one. The procedure is recursively applied to each of the subplanes. The recursion stops when the subplane is coded by a zero or when the size of the subplane is 1×1. The four-bit method presupposes a square-like form with 2^k pixels in each side.

The *Binary Tree Predictive Coding (BTPC)* [6] is designed for lossy and lossless image compression and is suitable for both photographs and graphics. The method forms a binary pyramid of the image by recursively dividing the image into two subimages whose sizes are half of the size of the previous image. The first subimage (S) is formed from the original image by leaving out alternate pixels; from each odd-numbered row the even-numbered pixels are discarded, and from each even-numbered row the odd-numbered pixels are discarded, see Fig. 2. The second subimage (E) stores the pixels missing from the first subimage, but in a transformed form; a context model (shape-based prediction, which uses different predictors for edges, flat areas, etc.) uses the neighbouring pixels in the subimage S to predict a pixel value. The actual stored value is the prediction error, i.e. the difference between the predicted and the real pixel value. The prediction error transform aims to a skewed distribution of values, which can be effectively compressed. The next layer of the image pyramid is constructed from the previous S subimage by the same process.

Each error value of a particular level of the pyramid has two children at a lower level of the pyramid. Thus, error values form a binary tree. The actual coding of the image pyramid is done by adaptive Huffman coding. If all the descendants of a zero error value are also zeros, the subtree can be coded by a single codeword. The implementation used in this study works for 8-bit images only.

Portable Network Graphics (PNG) [7,8] is essentially a file format for lossless compression and storage of images. It was developed to replace GIF. In addition to the images, one can also store other data in PNG files. The system accepts pixel depths 1, 2, 4, 8 and 16 and consumes no extra bits between two pixels in the storage. The compression method can also handle other pixel depths given to it. It is possible to filter the data before compression to get a better compression ratio.

The actual compression is performed by the so-called *deflate/inflate-compression*. Deflate is an LZ77-based method which is used in several general compression programs (e.g. zip, gzip). Huffman coding is included in the method to encode the data. We used the so-called *Paeth-filtering* [9] for the image preprocessing. The compression level option was set to default compression (6), which turned out to be two to three times faster than the best compression with only 1% decrease of compression ratio. A simple main program was made to call the png library, which is freely available on the Internet.

The *SPIHT*-compression technique (*Set Partitioning in Hierarchical Trees*) [10] transforms the image to a multiresolution representation by the wavelet transform or, in the case of lossless compression, by the S transform. This transformation is similar to the subband decomposition, but uses only integer operations. The S transform scans the columns of the image and calculates for each successive pair of the pixels the average and the difference. The averages are stored at the upper part of the image (*l*), and the differences are stored at the lower part (*h*), see Fig. 3. The same is repeated for the columns of *l* and *h* parts giving *ll*, *lh*, *hl* and *hh* images. At the next level the *ll* block will be processed in the same way. This gives a multiresolution pyramid of the image with reduced variance.

S1	E1	S1	E1	S1
E1	S1	E1	S1	E1
S1	E1	S1	E1	S1
E1	S1	E1	S1	E1
S1	E1	S1	E1	S1

The pixels after the first iteration cycle. S corresponds to the subsampling of the image and E corresponds to the difference between the true value of the pixels and the prediction based on the neighboring S values.

S4	E1	E3	E1	S4
E1	E2	E1	E2	E1
E3	E1	E4	E1	E3
E1	E2	E1	E2	E1
S4	E1	E3	E1	S4

The pixels after four iteration cycles. The S1 values have been processed recursively to yield E2, E3 and finally E4 and S4 values. The values S4, E1, E2, E3 and E4 are transmitted.

Fig. 2. Principle of the pyramid coding in BTPC.

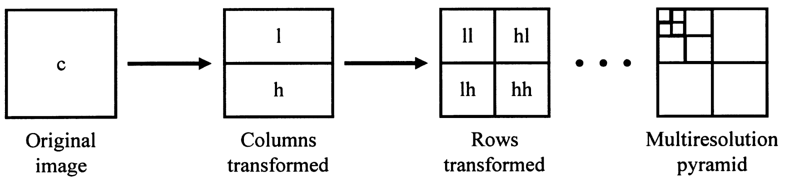


Fig. 3. Principle of the SPIHT compression technique.

After each one-dimensional transform, the pixel differences are replaced by the difference of transformed pixel value and prediction value. This is known as the *P transform*. The prediction values are calculated as a function of pixels of the lower levels of the pyramid and the known pixels of the current level. The prediction function can be selected on the basis of the properties of the image, but the coefficients of the function are not very critical. Arithmetic coding concludes the encoding process.

TMW [11] is based on the use of linear predictors and implicit segmentation. The compression process is split into an *analysis step* and a *coding step*. In the first step, linear predictors and other parameters suitable for the image are calculated. These are included in the compressed file and subsequently used for the coding step.

In TMW, three different kinds of predictors are used. *Pixel-predictors* predict a pixel value. *Sigma-predictors* predict the magnitude of pixel-predictor's prediction error. *Blending-predictors* predict how well suited a particular pixel-predictor is to predict a pixel value. These all are based on the corresponding values of the causal neighbours. In the analysis step, the parameters of these predictors are iteratively optimized. This is very time-consuming. However, if we have several images that resemble one another, it is sufficient to build a model for only one of them. This model can then be used for compressing all the other similar images. The coding step is much faster and basically consists of arithmetic coding of the prediction error according to the distribution.

The implementation used by us failed to reconstruct the 16-bit images. However, it seems that the problem lies in the decompressor. Thus, the results have been presented. We used the parameter file created for the test image ref12b-0 in all our tests, because the model generation software was not available. Better results would probably have been achieved if we had been able to build a model for each image type.

The *Low Complexity Lossless Compression for Images (LOCO-I)* [12] is based on context modelling and predictive coding combined with Huffman coding. The pixel values are predicted adaptively according to the output of a simple edge detector. The prediction value is a simple function of the values of the predictor pixels, which are chosen among three neighbouring pixels.

The prediction residuals are coded by the *Golomb–Rice code* [13,14] in the context that is built out of the differences of four local pixel values. The differences are quantized to reduce the number of contexts. Golomb–Rice codes are characterized by one parameter that forms the optimal probability distribution for the code. For the present context, the parameter value, which is intended to yield as short a code length as possible, is calculated from the estimate of the expected prediction error magnitude in the context. For flat regions, which are detected by checking the context, LOCO-I uses *run-length coding*. Here, a run of symbol is coded by coding the length of the run.

LOCO-I has been chosen as the algorithm of *JPEG-LS*, the emerging lossless/near-lossless compression standard for continuous-tone images. A freely available implementation of the method was used for testing. Default values were chosen for the thresholds for context quantization.

The *JPEG (Joint Photographics Experts Group)* standard for lossless compression LJPEG [15] is based on a prediction method combined with the entropy coding. The prediction method transforms the original image to an error image with reduced variance using a predictor, which is initially chosen among eight possible predictors. The prediction is a function of some of the three previous neighbouring pixels (pixels to the left, above and upper-left of the current pixels). The error image is finally compressed using Huffman or arithmetic coding.

In this study the Huffman code was applied for the compression of the error images. The Huffman codes were derived from static codes for 8-bit images by adding the missing codewords. The Huffman code chooses a category for the error value on the basis of the most significant bit. The codeword for the error value is the codeword for the category combined with rest of the bits in the error value (the sign and the least significant bits). In the tests we have used predictor $W + \lfloor (N - NW)/2 \rfloor$ (W corresponds to the pixel west of the current pixel and so on).

FELICS (Fast, Efficient, Lossless Image Compression System) [16] is based on context modelling and Rice coding. The context for a pixel consists of the two previous pixel values L and H, which make up a value range from the smaller (L) of the values up to the larger (H). Coding of the present pixel value P starts by checking whether the pixel value belongs to the value range or not. The result of this check is indicated by the first bit in the codeword. If the value belongs to the range, the difference $P - L$ is coded using a binary code. If the pixel value does not belong to the range, the second bit in the codeword indicates whether the value is bigger or smaller than any of the values in the range. In the first case the difference $P - H - 1$ is coded using the exponential Rice prefix code. Otherwise, the difference $L - P - 1$ is coded using the same code. The Rice code has one parameter. The parameter value giving the shortest code length is estimated for each context during the coding process.

Two implementations of FELICS were used in this study. One program was implemented for images with pixel depth 8 while the other was implemented for images with pixel depths from 9 to 16.

The *Context-based Lossless Image Compression (CLIC)* [17] uses several contexts to estimate probability density functions of error values, which are finally coded by the arithmetic coding. The error values are calculated as a difference between a pixel value and a prediction. The prediction function can be chosen among several possible functions such as $(N + W)/2$, where N corresponds to the pixel value north (above) of the current pixel, etc. This is the prediction function we used. The probability density function of the error values is estimated for each context by calculating frequencies of error values. In order to save memory, this can be done bitwise by calculating the frequency of each bit (having value one) in each context. In this case each bit is coded separately. The context was built out of the error values of the pixels above and left of the current pixel.

We used a backup model when a context of the main model was updated less than 64 times. The prediction function of the backup model was the same as in the main model. The context of the backup model was the error value of the pixel left of the current pixel.

The *Graphics Interchange Format (GIF)* is a widely used image format by CompuServe Incorporated. It applies an LZW based compression method. The format is restricted to an 8-bit grayscale and palette colour images. The format is the most suitable for storing drawn images with few colours.

Table 3
General-purpose compression programs[a]

Name	Version	Author	Year	Status	DOS	W95	Linux	src	beta	solid
ACB	2.00c	George Buyanovsky	1997	Shareware	×					×
ARJ	2.60	Robert Jung	1995	Shareware	×	×				
BZIP	0.21	Julian Seward	1996	Freeware		×	×	×		×
BZIP2	0.1pl2	Julian Seward	1997	Freeware		×	×	×		×
ESP	1.92	GyikSoft	1997	Freeware	×					×
GZIP	1.2.4	Jean-loup Gailly	1993	Freeware	×		×	×		×
HA	9.999β	Harri Hirvola	1995	Freeware	×		×	×	×	
JAR	1.02	Robert Jung	1997	Shareware	×	×				×
PKZIP	2.04g	PKWARE Inc.	1993	Shareware	×	×				
Quantum	0.97	Cinematronics	1995	Freeware	×					
RAR	2.02	Eugene Roshal	1997	Shareware	×	×	×			×
RKIVE	1.90β1	Malcolm Taylor	1997	Shareware	×	×			×	×
SZIP	1.04	Michael Schindler	1997	Freeware	×	×	×			×
SZIP	1.05Xd	Michael Schindler	1997	Freeware	×	×	×		×	×
UC2	3.0 Pro	AIPNL	1995	Shareware	×	×				×
UFA	0.034β4	Igor Pavlov	1997	Shareware		×			×	×
X1	9.95a	Stig Valentini	1997	Freeware	×	×	×		×	×
YAC	1.02	Aleksandras Surna	1995	Freeware	×					×

Name	Options used	Notes
ACB	u (max)	16 MB RAM required
ARJ	-jm (max)	many options, popular
BZIP	-9 (max block size)	single file compressor
BZIP2	-9 (max block size)	no arithmetic compression
ESP	/mm (8-bit multimedia)	
GZIP	-9 (max)	single file compressor, popular
HA	21 (choose best)	
JAR	-m4 (max)	
PKZIP	-ex (max)	popular
Quantum	-c5 (max) -t21 (max table size)	
RAR	-mm (multimedia) -m5 (max) -s (solid)	popular
RKIVE	-mtx (best mode) -mm2 (multimedia forced)	-mtx requires 32MB RAM
SZIP 1.04		single file compressor
SZIP 1.05Xd	-b41 (max blocksize) -o255 (max order) -r1 (recordsize)	new, experimental version
UC2	-tst (tight multimedia)	
UFA	-m5 (text compression) -mx (max) -s (solid)	
X1	u (solid) m2 (method 2)	
YAC		

[a]DOS = program is available for plain DOS, W95 = program (possibly a separate version) supports Windows 95™ long file names, Linux = Linux version is available, src = program sources are available, beta = program is in beta testing, i.e. not an official release, solid = program supports solid compression.

3.2. General-purpose lossless compression methods

Table 3 contains a summary of the general-purpose lossless compression methods applied in this work. Most of these programs are selected according to our previous knowledge on their ability to compress general data. GZIP, PKZIP and ARJ have been included because they are widely used.

It should be noted that BZIPs, GZIP and SZIPs are single file compressors. When several images were compressed in the same file, TAR program was used in these cases. Thus, compression was *solid*, i.e. all the files to be compressed are handled as a single file. It is well known that if the files are of the same kind, this usually gives better results than compressing the files separately. We have tested solid compression if the program supports it. Note that TAR inserts headers between the files, but this was found to have no practical effect on the compression ratio.

There are two versions of BZIP and SZIP. BZIP2 0.1pl2 is a newer version of BZIP 0.21, but its compression results are worse. This is because instead of the arithmetic coding used in BZIP, Huffman coding is used in BZIP2. (On the other hand, BZIP2 is patent-free, unlike BZIP.) SZIP 1.05Xd is an experimental, non-supported version of SZIP. It has some options so that it can be made to compress better at the expense of increased running time.

4. Empirical tests

Table 4 shows a summary of the test results for the 11 image compression techniques. The numbers shown in the table are the average compression ratios calculated for the different pixel depths and sizes. Here we will express the results as averages over all these. For small images we show (if possible) the results for both single and merged images.

The results of Table 4 indicate that the four most space-efficient compression techniques for this set of data are CALIC, TMW, SPIHT and JPEG-LS. The table also shows the average compression ratio of each compression method. Here the average is calculated as the total size of uncompressed images divided by the size of compressed database. Therefore, the averages are close to the results for CT/HEAD type of images, as two-thirds of the database is of this type. (Note that the weighted average is not the only way to compare the different compression techniques—compare with the unweighted average of all compression ratios of images or studies.) The average compression ratio of CALIC is 2.98 for all image types. For SPIHT and JPEG-LS the results are essentially of the same quality. The average compression ratio of TMW was as much as 3.17. This is because TMW compressed CT/HEAD images over 10% better than CALIC. CALIC compressed most of the other images better. However, if a unique model were generated for each image type, the compression results of TMW would probably be better.

The variation of the compression ratios of the techniques is large among different image types. For CALIC the compression ratio ranges from 2.06 to 5.23. For single images the range is even greater, 1.52–24.16 (there are three images with compression ratios > 10). In addition, there are some cases where the relative order of the three most effective techniques is not clear; for MR images SPIHT gives the highest compression ratios and for NM/BRAIN images JPEG-LS has the highest ratio. It is interesting to note that FELICS is essentially as efficient as CALIC for the very small (64×64) images. Also LJPEG works relatively well for the very small images. JPEG-LS is especially space-efficient for the 8 bpp images.

In the next category of the techniques we have FELICS and CLIC. FELICS succeeds well on the very small images—it gives the best results for the NM/HB images compressed separately. This is probably because FELICS is a rather simple method and the very small images (64×64) are not suitable for more sophisticated methods. For example, the results of TMW are very poor on the small

Table 4
The average compression ratios for the image compression methods by image type (JP-LS = JPEG-LS, FEL = FELICS)

Mod	Type	TMW	CALIC	SPIHT	JP-LS	CLIC	FEL	LJPEG	PNG	CPM	BTPC	GIF
CR	PED CHEST	2.71	2.86	2.81	2.74	2.68	2.60	2.24	1.97			
CT	ABDOMEN	3.77	3.72	3.44	3.59	3.40	2.90	2.68	2.44			
CT	HEAD	3.16	2.83	2.82	2.65	2.50	2.18	2.05	1.74			
MR	ABDOMEN	1.98	2.06	2.03	2.02	1.86	1.89	1.71	1.45	1.57		
MR	SPINE	2.81	2.90	2.94	2.81	2.69	2.67	2.42	2.05	2.19		
NM	BRAIN	3.35	4.87	4.58	4.96	4.52	3.67	2.55	3.79	3.28	4.30	4.04
	merged	*5.04*	*5.23*	*4.78*	*5.11*	*4.81*	*3.77*	*2.55*	*3.99*		*4.83*	*4.40*
NM	HB	2.08	3.28		2.71	2.44	3.28	2.90	2.31	2.58		
	merged	*3.45*	*3.44*	*3.39*	*3.39*	*3.26*	*3.36*	*2.91*	*2.46*			
NM	HEART	2.11	3.36		2.77	2.51	3.36	2.96	2.36	2.64		
	merged	*3.54*	*3.54*	*3.48*	*3.47*	*3.35*	*3.45*	*2.97*	*2.51*			
NM	WB	3.81	4.09	3.84	4.05	3.82	3.28	2.33	3.24		3.77	3.19
US	(missing)	2.80	3.02	2.54	2.90	2.73	2.40	2.02	2.50		2.40	1.94
Total		3.17	2.98		2.81	2.67	2.36	2.18	1.90			
	merged	*3.21*	*2.99*	*2.94*	*2.82*	*2.68*	*2.36*	*2.18*	*1.90*			

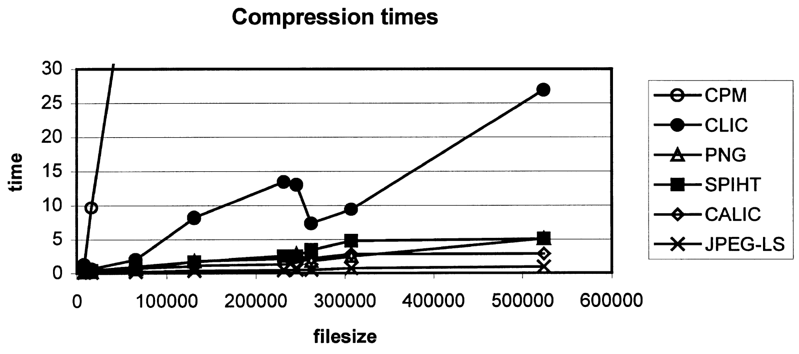


Fig. 4. Average compression times (s) for image compression methods with respect to filesize (bytes) for Axil320. The compression times of TMW, BTPC, GIF, FELICS and LJPEG are omitted. TMW results were run on a different hardware. BTPC and GIF were not able to compress 16 bit images. FELICS compression times are essentially the same as for SPIHT. LJPEG compression times are between JPEG-LS and CALIC.

images because TMW generates a 1596 byte parameter file for each image, which we have included in the compressed filesize.

The CPM algorithm presupposes a square-like image and we therefore restricted our test with it to this kind of images only. Similarly, our GIF results are restricted to 8 bpp images. For both of these techniques the results do not encourage modifications allowing other image types. Also the implementation of BTPC was restricted to 8 bpp images. However, moderately good results were obtained by BTPC.

One aspect is the merging of smaller images to form a larger one. A general observation is that the mutual preference of the different methods is preserved by the merging operation. The compression ratio, however, increases radically (for certain images even by 30%). This is because the information obtained from one image can be utilized when compressing other images. The improvement is greatest for JPEG-LS and CLIC. See Appendix B for more details on the test results.

Tables 6–8 give a summary of the compression and decompression times. The tests were run on an Axil320 computer equipped with two 90 MHz HyperSPARC processors. The results of Tables 7 and 8 are given for different sizes and pixel depths (ignoring the image types). These results are also illustrated in Figs. 4 and 5. It is observed that JPEG-LS, LJPEG, BTPC and GIF are very fast.

The compression of a very large image CR.17220.1 (resolution 2500 × 2048 × 12) runs in about 13 s for JPEG-LS. (Note that CR-17220.1 is not the same CR/PED CHEST image as used in the testing of the general compression programs.) As a second group we have CALIC, SPIHT and FELICS. Compression times for CR.17220.1 are 49, 102 and 85 s, respectively.

For PNG with fixed filter selection the average compression time is two to three or even four times (depending on the modality) larger with the best (9) level than with the default (6) level while the gain in compression ratio is only 1%. The compression time with the default level is close to CALIC and SPIHT but the decompression time is among the fastest of the methods tested here. This is a clear advantage of PNG, because the decompression is in many cases time critical.

Merging of the images has in most cases a favourable effect on the time usage of the techniques. For example, JPEG-LS runs for merged NM/HEART images in one-tenth of the time (per image) needed for the compression of the same single images. The compression of the 30 images in the szl study took 4.75 s and the merged image required only 0.5 s.

Figs. 6 and 7 illustrate the (time/compression ratio) trade-off of the compression methods. The results are averages of all test runs performed. JPEG-LS, CALIC and SPIHT

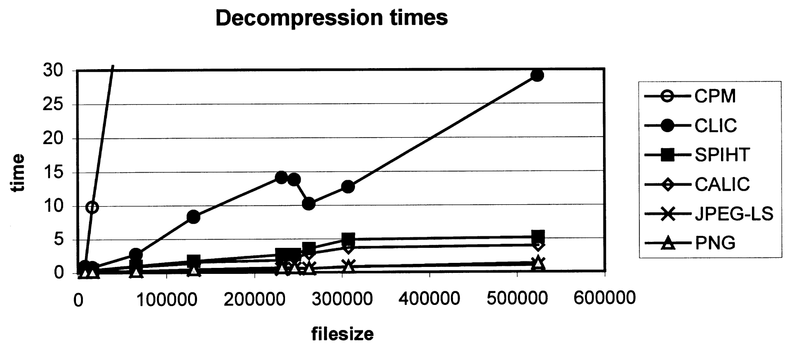


Fig. 5. Average decompression times (s) for image compression methods with respect to filesize (bytes). The compression times of TMW, BTPC, GIF, FELICS, PNG and LJPEG are omitted. FELICS decompression times are essentially the same as for SPIHT. LJPEG decompression times are between JPEG-LS and CALIC.

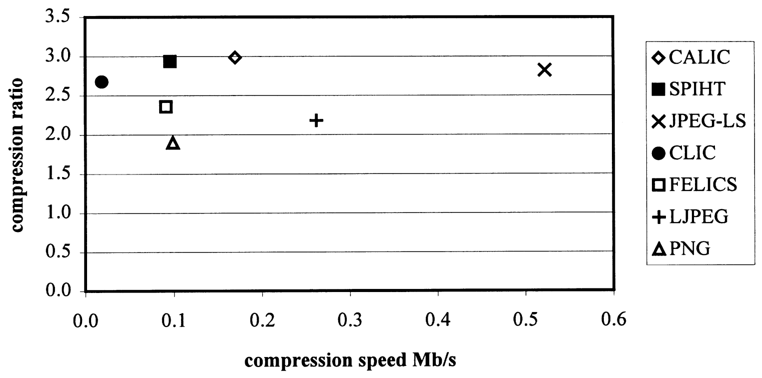


Fig. 6. Compression ratios and compression speed (Mb s^{-1}) for some methods. Total filesizes have been used in the speed calculations. The merged images were used in all the calculations, because SPIHT was unable to compress the small images.

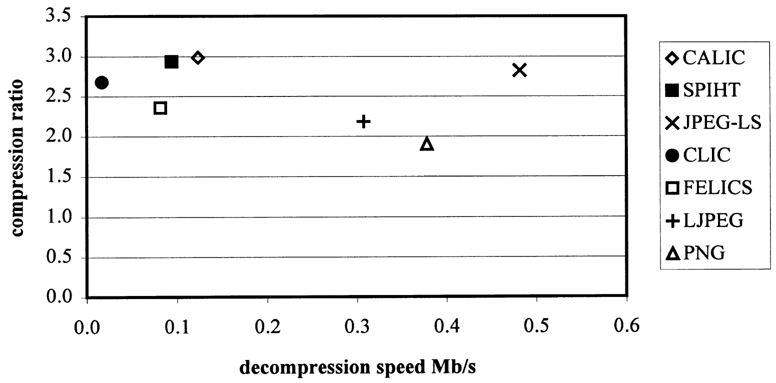


Fig. 7. Compression ratios and decompression speed (Mb s^{-1}) for some methods.

compare favourably with the other methods. The compression ratio of CLIC is relatively good but the method suffers from the long running time.

Table 9 shows the compression results for 18 general compression programs. These were not tested for all the data. Instead, a sample study of each image type was chosen. The CALIC results for these studies are included for comparison. The averages are unweighted and should therefore be used with caution. In particular, they should not be compared to the compression ratios in Table 4.

Table 10 shows the best general compression programs (with CALIC included for comparison) for each image type. A summary of these results is given in Table 5. It turns out that, as expected, most of the time CALIC outperforms the general compression programs when the images are compressed separately. However, in the case of NM/BRAIN images, SZIP and ACB perform even better. This is probably because these images are of a very special type, see Appendix A. Furthermore, the images contain practically only four bits of information, which also explains the high compression ratios. See Appendix B for more data on the general compression methods.

By allowing the compression programs to archive a whole study in a single file, much higher compression ratios

Table 5
Comparison between the best general compression programs and CALIC by study type. The compression ratios are averages on all the images of the sample

Mod	Type	CALIC	Best general compressor		Best general archived		CALIC merged	Best general merged	
CR	PED CHEST	2.59	2.22	BZIP	2.22	BZIP			
CT	ABDOMEN	3.46	2.79	BZIP	2.89	UFA			
CT	HEAD	2.91	2.72	BZIP	2.84	SZIP 1.05			
MR	ABDOMEN	2.42	2.12	UFA	2.16	UFA			
MR	SPINE	3.09	2.55	UFA	3.70	RKIVE			
NM	BRAIN	4.86	4.98	SZIP 1.04	5.28	UFA	5.22	5.37	UFA
NM	HB	3.60	3.50	SZIP 1.05	3.72	SZIP 1.05	3.80	3.78	UFA
NM	HEART	3.09	2.86	BZIP	3.14	UFA	3.23	3.20	UFA
NM	WB	3.91	3.85	SZIP 1.05	6.00	ACB			
US		2.95	2.89	UFA	2.99	UFA			

are often achieved—see the rows marked ''all'' in Table 9. Note that CR/PED CHEST contains only one image, so archiving does not change anything. The MR/ABDOMEN study consists of two images, which also does not give a remarkable increase in performance. The most convincing results in this respect are for the NM/WB study. For example, ACB achieves over 50% better compression ratio when compressing the NM/WB study in one file. There is a good explanation for this behaviour. The NM/WB study consists of six images. Five of these are taken from different parts of the human body and one image is made by attaching the relevant parts of the other images side by side. Thus, most of the information is included twice in this study.

5. Concluding remarks

In this study we performed practical tests with a large set of medical patient images from a real archiving system. A difficulty with the testing was that the images were of varying size and pixel depths. This made the application of the algorithms rather difficult. Some of the compression programs do not accept all image types. Because the programs are mostly quite involved we abandoned their revision. Our results are therefore not complete for all methods.

The compression ratios with the four best methods (TMW, CALIC, SPIHT, JPEG-LS) vary from about 2 to 5. This result is in our opinion quite modest. Therefore, a topic for discussion is whether a limited and selective application of lossy compression techniques could be allowed in medical imaging.

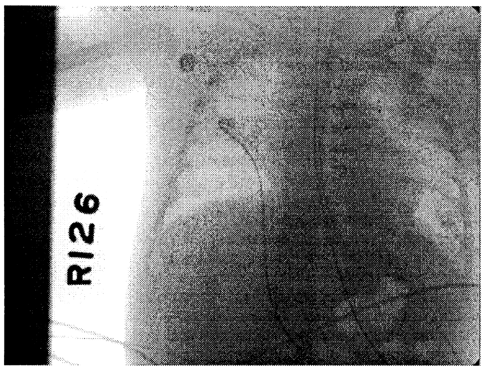
We applied the compression techniques to the whole image file. A natural question is whether we could segment the image to leave out uninformative parts. The use of variable quality compression could increase the storage efficiency essentially. This would, however, presuppose the utilization of application dependent knowledge. Another question is what happens if we homogenize the unimportant areas to contain (say) zeros only. In our images these areas contain some noise which may be hard to compress. The operation of general compression algorithms was a surprise to us. Our original hypothesis was that they are of no use in image compression but there still exists some image types for which general compressors are acceptable.

After the first writing of this report we found a recent paper by Denecker et al. [18] on the state of the art of lossless medical image compression. The paper contains an informative overview on the algorithms. Their set of image compressors is the same as ours except for TMW, CLIC, CPM, PNG and GIF. They also used two general data compressors, GZIP and STAT. The tests have been carried out for a set of medical images (computed tomography, magnetic resonance imaging, positron emission tomography, ultrasound, X-ray, angiography) from the public domain. The pixel depths have been transformed to 8 bpp (even for mammograms). CALIC and LOCO-I were the most space efficient algorithms in these studies and the average compression ratios ranged from 2.04 to 3.65 for their data. The dependency of the compression result on the image type was not discussed for different algorithms.

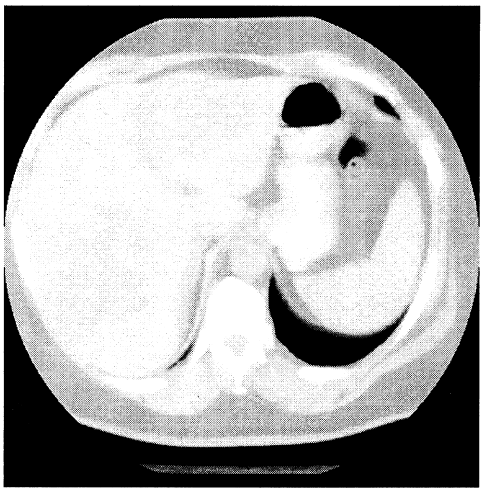
Acknowledgements

The authors wish to thank Antal Nagy from the József Attila University for useful discussions and comments on our manuscript. The authors are grateful to the Departments of Radiology and Nuclear Medicine, Szent-Györgyi Albert Medical University, for supplying a part of the images for the compression studies. This work was partially supported by the Hungarian Ministry of Education Grant FKFP 0908.

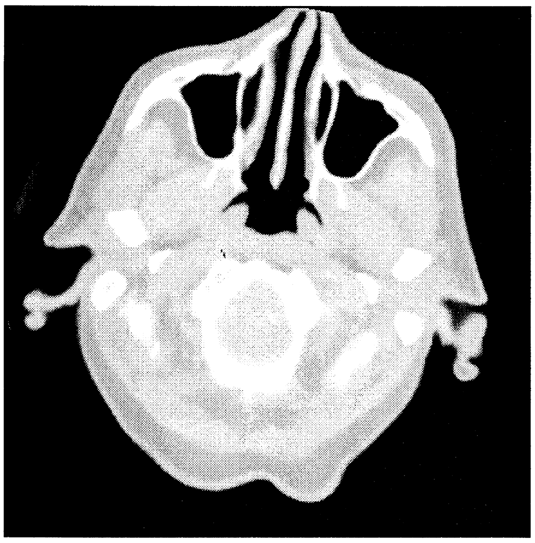
Appendix A Image types

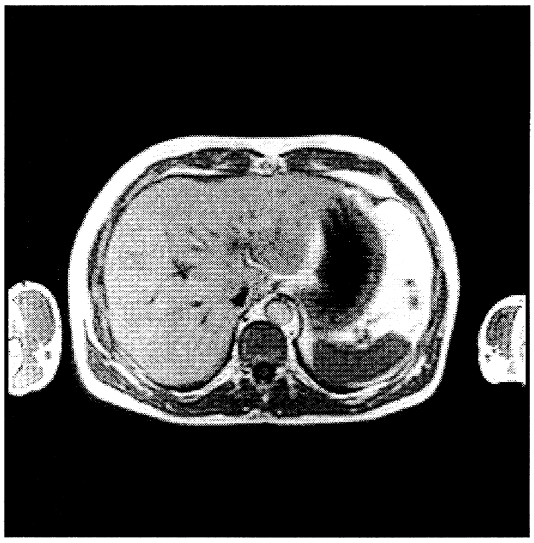


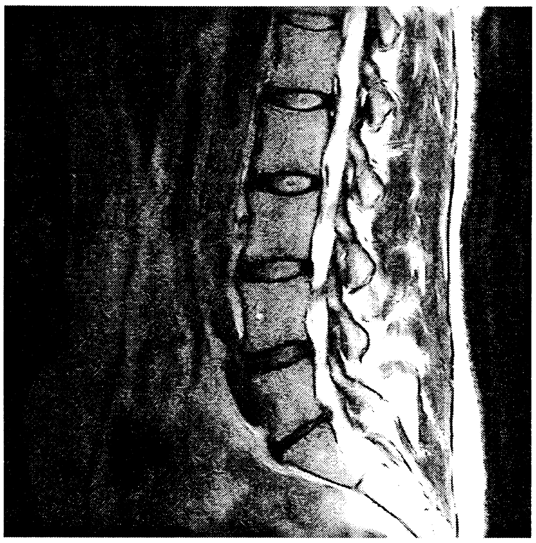
CR / PED CHEST (2392 × 1792 × 12)

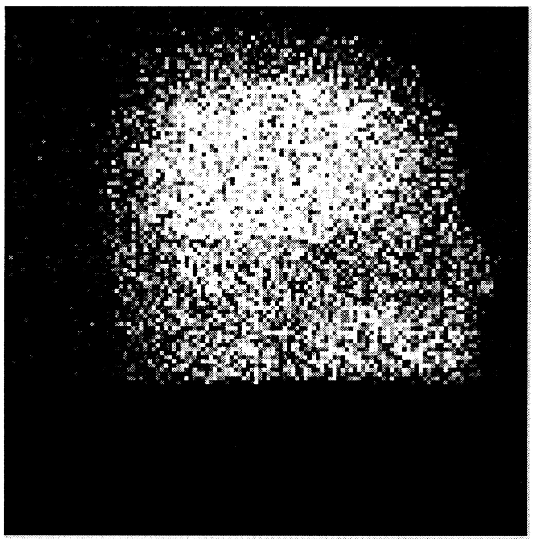


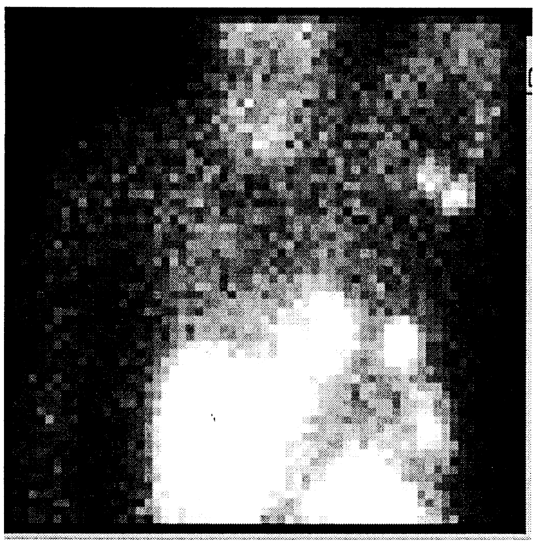
CT / ABDOMEN (512 × 512 × 12)

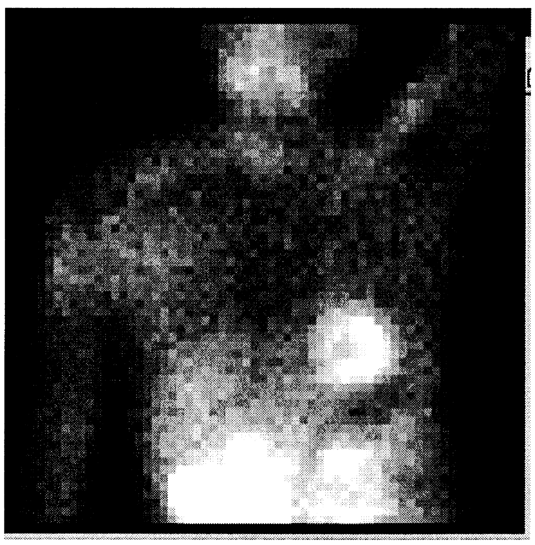

CT / HEAD (512 × 512 × 12)

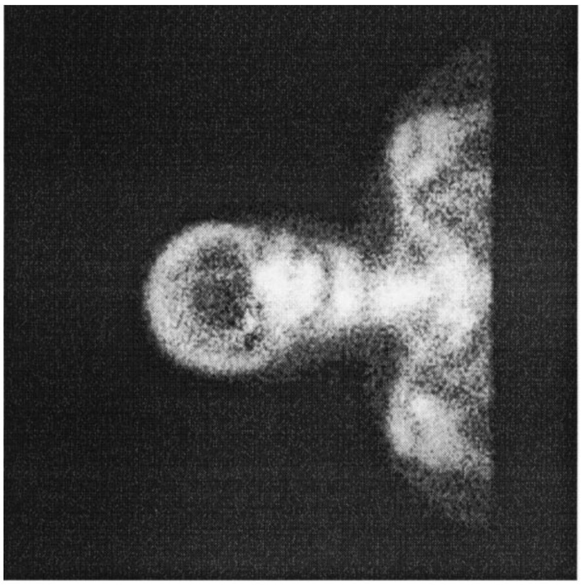

MR / ABDOMEN (256 × 256 × 12)

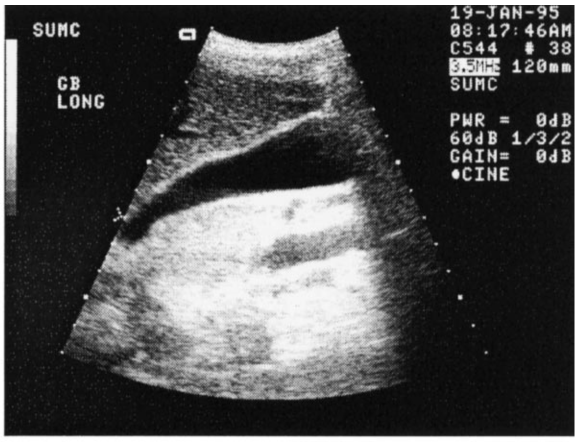

MR / SPINE (512 × 512 × 16)


NM / BRAIN (128 × 128 × 8)


NM / HB (64 × 64 × 16)


NM / HEART (64 × 64 × 16)


NM / WB (256 × 256 × 8)



US (640 × 480 × 8)

Appendix B More test results

Table 6
The average compression and decompression time (s) for the image compression methods by image type (JP-LS = JPEG-LS)

Mod	Type	CALIC	SPIHT	JP-LS	CLIC	FELICS	LJPEG	PNG	CPM	BTPC	GIF
CR	PED CHEST	45.70	92.36	12.74	420.75	81.61	29.94	85.71			
		63.60	92.73	13.91	465.41	92.21	25.34	17.82			
CT	ABDOMEN	2.72	5.06	0.75	26.73	4.94	1.76	4.65			
		3.78	5.16	0.80	29.01	5.56	1.51	1.28			
CT	HEAD	2.65	4.65	0.92	25.04	5.00	1.77	4.69			
		3.64	4.75	0.99	26.97	5.63	1.50	1.22			
MR	ABDOMEN	1.03	1.65	0.39	8.18	2.02	0.59	1.72	56.90		
		1.35	1.70	0.41	8.33	2.21	0.50	0.47	59.58		
MR	SPINE	2.02	3.38	0.59	16.05	3.55	1.16	3.54	160.74		
		2.74	3.46	0.65	17.03	3.95	0.98	0.82	158.22		
NM	BRAIN	0.37	0.44	0.16	0.52	0.31	0.14	0.23	9.68	0.14	0.22
		0.40	0.44	0.16	0.71	0.29	0.13	0.16	9.83	0.09	0.15
	merged	*13.65*	*23.69*	*2.49*	*51.76*	*27.02*	*7.58*	*12.40*		*5.56*	*12.76*
		17.66	*24.16*	*2.82*	*75.06*	*27.86*	*7.02*	*3.76*		*2.44*	*5.64*
NM	HB	0.29		0.17	1.35	0.83	0.10	0.21	1.05		
		0.31		0.17	0.91	0.84	0.10	0.13	0.97		
	merged	*1.62*	*2.59*	*0.50*	*13.12*	*2.91*	*0.92*	*2.85*			
		2.20	*2.66*	*0.54*	*13.87*	*3.21*	*0.78*	*0.72*			
NM	HEART	0.29		0.17	1.34	0.82	0.10	0.20	1.07		
		0.31		0.17	0.90	0.83	0.09	0.13	0.99		
	merged	*1.59*	*2.50*	*0.49*	*12.98*	*2.89*	*0.90*	*2.83*			
		2.12	*2.57*	*0.53*	*13.75*	*3.19*	*0.78*	*0.72*			
NM	WB	0.97	1.39	0.28	2.71	1.44	0.47	0.78		0.37	0.75
		1.21	1.41	0.29	3.87	1.42	0.44	0.34		0.20	0.40
US		2.75	4.63	0.72	8.41	5.03	1.39	2.40		1.17	2.11
		3.56	4.75	0.80	11.97	4.87	1.25	0.78		0.51	1.13

Table 7
The average compression times (s) for the image compression methods by image size[a]

Resolution		bpp	Count	Size	CALIC	SPIHT	JP-LS	CLIC	FEL	LJPEG	PNG	CPM	BTPC	GIF
64	64	16	990	8.1	0.29		0.17	1.34	0.82	0.10	0.20	1.06		
128	128	8	480	7.9	0.37	0.44	0.16	0.52	0.31	0.14	0.23	9.68	0.14	0.22
256	256	8	70	4.6	0.73	0.97	0.23	1.83	0.98	0.34	0.58	51.92	0.28	0.54
256	256	12	55	5.4	1.05	1.67	0.39	8.13	2.02	0.59	1.74	55.03		
256	256	16	59	7.7	1.14	1.68	0.35	8.26	2.04	0.58	1.81	52.78		
340	340	12	162	28.1	1.34	2.55	0.42	13.45	2.61	0.87	2.16			
384	320	16	33	8.1	1.60	2.54	0.50	13.04	2.90	0.91	2.83			
1024	256	8	14	3.7	2.19	3.45	0.52	7.10	3.78	1.14	1.78		0.86	1.80
640	480	8	30	9.2	2.81	4.75	0.75	8.61	5.17	1.43	2.50		1.20	2.16
512	512	12	1142	449.1	2.83	5.00	0.96	26.89	5.34	1.89	5.04	287.5		
512	512	13	18	7.7	2.64	4.34	0.81	26.79	4.72	1.74	4.24	288.4		
512	512	16	106	55.6	3.25	5.68	0.85	27.30	5.43	1.91	5.70	312.8		
Other			25	41.1	11.24	21.74	2.86	88.90	20.44	7.05	17.68			

[a]Size = total size in megabytes; JP-LS = JPEG-LS, FEL = FELICS.

Table 8
The average decompression times (s) for the image compression methods by image size[a]

Resolution		bpp	Count	Size	CALIC	SPIHT	JP-LS	CLIC	FEL	LJPEG	PNG	CPM	BTPC	GIF
64	64	16	990	8.1	0.31		0.17	0.90	0.83	0.09	0.13	0.98		
128	128	8	480	7.9	0.40	0.44	0.16	0.71	0.29	0.13	0.16	9.83	0.09	0.15
256	256	8	70	4.6	0.88	0.99	0.24	2.60	0.95	0.32	0.29	53.16	0.16	0.30
256	256	12	55	5.4	1.37	1.72	0.42	8.31	2.20	0.50	0.47	57.74		
256	256	16	59	7.7	1.51	1.73	0.38	8.41	2.22	0.50	0.47	52.73		
340	340	12	162	28.1	1.81	2.61	0.45	14.10	2.89	0.75	0.69			
384	320	16	33	8.1	2.16	2.61	0.53	13.81	3.20	0.78	0.72			
1024	256	8	14	3.7	2.82	3.50	0.56	10.19	3.78	1.04	0.59		0.41	0.90
640	480	8	30	9.2	3.65	4.87	0.84	12.27	4.99	1.28	0.80		0.52	1.16
512	512	12	1142	449.1	3.90	5.10	1.04	29.04	6.01	1.60	1.31	293.1		
512	512	13	18	7.7	3.69	4.46	0.85	29.19	5.33	1.50	1.28	273.5		
512	512	16	106	55.6	4.51	5.82	0.92	29.56	6.13	1.62	1.33	298.9		
Other			25	41.1	15.37	21.97	3.12	101.8	22.44	6.11	4.16			

[a]Size = total size in megabytes; JP-LS = JPEG-LS, FEL = FELICS

Table 9
The compression ratios for the general compression programs[a]

Mod	Type	CAL	SZx	SZ	BZ	UFA	BZ2	ACB	X1	HA	RKV	Q	UC2	RAR	JAR	GZ	ZIP	ARJ	YAC	ESP
CR	PED CHEST	2.59	2.22	2.20	2.22	2.21	2.22	2.01	2.11	2.04	1.83	1.57	1.79	1.93	1.85	1.28	1.30	1.28	1.41	1.32
	all		2.22	2.20	2.22	2.21	2.22	2.01	2.11	2.04	1.83	1.57	1.79	1.93	1.85	1.28	1.30	1.28	1.41	1.32
CT	ABDOMEN	3.46	2.73	2.74	2.79	2.73	2.77	2.52	2.67	2.66	2.46	2.01	2.18	1.85	2.07	1.79	1.79	1.79	1.77	1.64
	all		2.88	2.84	2.85	2.89	2.83	2.65	2.78	2.66	2.47	2.09	2.19	1.85	2.09	1.79	1.79	1.79	1.78	1.65
CT	HEAD	3.91	2.59	2.59	2.72	2.62	2.71	2.34	2.49	2.39	2.24	1.88	2.08	1.98	1.93	1.76	1.77	1.76	1.80	1.57
	all		2.84	2.78	2.81	2.74	2.80	2.56	2.58	2.39	2.25	1.98	2.08	1.94	1.96	1.76	1.77	1.76	1.80	1.57
MR	ABDOMEN	2.42	2.07	2.06	2.10	2.12	2.08	2.00	2.03	2.02	1.82	1.67	1.54	1.63	1.64	1.55	1.54	1.55	1.46	1.38
	all		2.10	2.09	2.11	2.16	2.11	2.02	2.08	2.02	1.83	1.69	1.55	1.63	1.65	1.55	1.54	1.56	1.46	1.39
MR	SPINE	3.09	2.51	2.51	2.55	2.55	2.54	2.33	2.48	2.47	2.27	1.98	1.86	1.95	2.06	1.82	1.82	1.81	1.80	1.66
	all		2.58	2.55	2.56	2.61	2.55	2.43	2.55	2.47	3.70	2.05	1.87	1.96	2.08	1.82	1.82	1.81	1.80	1.70
NM	BRAIN	4.86	4.97	4.98	4.77	4.63	4.60	4.97	4.73	4.81	4.50	4.79	3.96	4.02	3.67	4.08	3.98	3.98	3.79	3.78
	all		5.15	5.12	5.02	5.28	4.96	5.23	5.07	4.81	–	4.46	4.20	4.15	4.20	4.17	4.00	4.03	3.84	3.93
	merged	5.22	5.17	5.15	5.03	5.37	4.98	5.25	5.12	5.20	4.98	4.67	4.37	4.31	4.29	4.37	4.36	4.31	3.91	4.05
NM	HB	3.60	3.50	3.48	3.47	3.23	3.14	3.23	2.96	2.97	2.95	2.50	2.21	2.23	2.33	2.31	2.25	2.27	2.17	1.83
	all		3.72	3.67	3.67	3.70	3.63	3.52	3.60	2.97	3.10	2.78	2.54	2.49	2.78	2.52	2.27	2.30	2.39	2.40
	merged	3.80	3.74	3.71	3.69	3.78	3.65	3.58	3.65	3.64	3.72	2.86	2.63	2.65	2.82	2.57	2.56	2.57	2.44	2.40
NM	HEART	3.09	2.86	2.85	2.86	2.74	2.53	2.66	2.45	2.45	2.19	2.10	1.89	1.89	1.99	1.96	1.92	1.92	1.86	1.47
	all		3.13	3.10	3.12	3.14	3.09	2.90	3.01	2.45	2.29	2.32	2.14	2.10	2.32	2.13	1.93	1.94	2.06	1.91
	merged	3.23	3.15	3.13	3.14	3.20	3.10	2.93	3.04	3.04	2.86	2.37	2.21	2.22	2.35	2.17	2.17	2.17	2.09	1.90
NM	WB	3.91	3.85	3.84	3.77	3.80	3.70	3.82	3.77	3.81	3.71	3.42	3.09	3.13	3.05	3.16	3.11	3.10	2.93	3.11
	all		5.08	4.24	4.98	4.38	4.88	6.00	3.86	3.81	4.83	5.48	3.10	3.14	4.87	3.15	3.12	3.11	4.23	3.12
US		2.95	2.88	2.87	2.76	2.89	2.74	2.77	2.79	2.77	2.84	2.22	2.28	2.16	2.07	2.11	2.13	2.13	2.01	2.34
	all		2.99	2.96	2.80	2.99	2.78	2.89	2.82	2.77	2.85	2.30	2.29	2.17	2.14	2.11	2.13	2.13	2.01	2.34
Average		3.39	3.02	3.01	3.00	2.95	2.90	2.87	2.85	2.84	2.68	2.41	2.29	2.28	2.27	2.18	2.16	2.16	2.10	2.01
	all	3.59	3.27	3.17	3.22	3.19	3.23	3.23	3.06	3.01	3.13	2.71	2.41	2.38	2.61	2.26	2.26	2.25	2.29	2.15

Mod	Type	Image count	Xsize	Ysize	bpp	Filesize	Realsize
CR	PED CHEST	1	1792	2392	12	8572946	6429696
CT	ABDOMEN	11	512	512	12	5767344	4325376
CT	HEAD	10	512*	512*	12	4656864	3492528
MR	ABDOMEN	2	256	256	12	262176	196608
MR	SPINE	40	512	512	16	20972200	20971520
NM	BRAIN	120	128	128	8	1967880	1966080
NM	HB	30	64	64	16	246210	245760
NM	HEART	30	64	64	16	246210	245760
NM	WB	6	256*	256*	8	589915	589824
US		23	480	640	8	7065945	7065600

[a]Only a single study of each image type was compressed. The study details are shown in the lower part of the table. CALIC has been included for comparison. RKIVE failed in archiving all the NM/BRAIN images; *all* = all files compressed in one archive; CAL = CALIC, Q = Quantum, SZx = SZIP 1.05x, SZ = SZIP 1.04, GZ = GZIP, BZ2 = BZIP2, BZ = BZIP, RKV = RKIVE, ZIP = PKZIP; *average* is the unweighted mean of the ratios in the column; * = the most common, but not only, resolution in the study.

Table 10
Compression ratios of the top 10 generic compression programs for each image type (CALIC has been included for comparison)

Compressed separately			Archived together (not CALIC)			Merged		
CR PED CHEST								
1	CALIC	2.589	1	CALIC	2.589			
2	BZIP	2.221	2	BZIP	2.221			
3	BZIP2	2.219	3	BZIP2	2.219			
4	SZIP 1.05X	2.216	4	SZIP 1.05X	2.216			
5	UFA	2.206	5	UFA	2.206			
6	SZIP 1.04	2.198	6	SZIP 1.04	2.198			
7	X1	2.111	7	X1	2.111			
8	HA	2.045	8	HA	2.045			
9	ACB	2.013	9	ACB	2.013			
10	RAR	1.931	10	RAR	1.931			
CT ABDOMEN								
1	CALIC	3.465	1	CALIC	3.465			
2	BZIP	2.789	2	UFA	2.886			
3	BZIP2	2.770	3	SZIP 1.05X	2.877			
4	SZIP 1.04	2.736	4	BZIP	2.846			
5	UFA	2.734	5	SZIP 1.04	2.836			
6	SZIP 1.05X	2.734	6	BZIP2	2.834			
7	X1	2.675	7	X1	2.784			
8	HA	2.662	8	HA	2.662			
9	ACB	2.520	9	ACB	2.653			
10	RKIVE	2.464	10	RKIVE	2.472			
CT HEAD								
1	CALIC	3.915	1	CALIC	3.915			
2	BZIP	2.720	2	SZIP 1.05X	2.839			
3	BZIP2	2.706	3	BZIP	2.814			
4	UFA	2.621	4	BZIP2	2.798			
5	SZIP 1.05X	2.595	5	SZIP 1.04	2.782			
6	SZIP 1.04	2.593	6	UFA	2.739			
7	X1	2.493	7	X1	2.583			
8	HA	2.394	8	ACB	2.559			
9	ACB	2.345	9	HA	2.394			
10	RKIVE	2.244	10	RKIVE	2.249			
MR ABDOMEN								
1	CALIC	2.417	1	CALIC	2.417			
2	UFA	2.117	2	UFA	2.155			
3	BZIP	2.098	3	BZIP	2.115			
4	BZIP2	2.082	4	BZIP2	2.109			
5	SZIP 1.05X	2.068	5	SZIP 1.05X	2.100			
6	SZIP 1.04	2.064	6	SZIP 1.04	2.094			
7	X1	2.032	7	X1	2.080			
8	HA	2.018	8	ACB	2.020			
9	ACB	1.998	9	HA	2.018			
10	RKIVE	1.824	10	RKIVE	1.832			
MR SPINE								
1	CALIC	3.088	1	RKIVE	3.704			
2	UFA	2.546	2	CALIC	3.088			
3	BZIP	2.546	3	UFA	2.606			
4	BZIP2	2.545	4	SZIP 1.05X	2.583			
5	SZIP 1.05X	2.514	5	BZIP	2.556			
6	SZIP 1.04	2.512	6	BZIP2	2.555			
7	X1	2.477	7	X1	2.554			
8	HA	2.469	8	SZIP 1.04	2.552			
9	ACB	2.333	9	HA	2.469			
10	RKIVE	2.271	10	ACB	2.429			
NM BRAIN								
1	SZIP 1.04	4.975	1	UFA	5.285	1	UFA	5.371
2	ACB	4.973	2	ACB	5.234	2	ACB	5.252
3	SZIP 1.05X	4.970	3	SZIP 1.05X	5.149	3	CALIC	5.217
4	CALIC	4.861	4	SZIP 1.04	5.116	4	HA	5.198
5	HA	4.805	5	X1	5.072	5	SZIP 1.05X	5.166
6	Quantum	4.794	6	BZIP	5.017	6	SZIP 1.04	5.147

Table 10 (*continued*)

Compressed separately			Archived together (not CALIC)			Merged		
7	BZIP	4.774	7	BZIP2	4.959	7	X1	5.124
8	X1	4.732	8	CALIC	4.861	8	BZIP	5.034
9	UFA	4.630	9	HA	4.811	9	RKIVE	4.983
10	BZIP2	4.603	10	Quantum	4.462	10	BZIP2	4.978
NM HB								
1	CALIC	3.600	1	SZIP 1.05X	3.719	1	CALIC	3.802
2	SZIP 1.05X	3.501	2	UFA	3.701	2	UFA	3.779
3	SZIP 1.04	3.483	3	BZIP	3.671	3	SZIP 1.05X	3.743
4	BZIP	3.469	4	SZIP 1.04	3.668	4	RKIVE	3.724
5	ACB	3.233	5	BZIP2	3.627	5	SZIP 1.04	3.710
6	UFA	3.229	6	X1	3.605	6	BZIP	3.693
7	BZIP2	3.145	7	CALIC	3.600	7	X1	3.649
8	HA	2.966	8	ACB	3.522	8	BZIP2	3.647
9	X1	2.961	9	RKIVE	3.101	9	HA	3.643
10	RKIVE	2.948	10	HA	2.970	10	ACB	3.577
NM HEART								
1	CALIC	3.088	1	UFA	3.136	1	CALIC	3.232
2	BZIP	2.862	2	SZIP 1.05X	3.133	2	UFA	3.200
3	SZIP 1.05X	2.857	3	BZIP	3.122	3	SZIP 1.05X	3.150
4	SZIP 1.04	2.850	4	SZIP 1.04	3.104	4	BZIP	3.138
5	UFA	2.740	5	CALIC	3.088	5	SZIP 1.04	3.132
6	ACB	2.658	6	BZIP2	3.087	6	BZIP2	3.103
7	BZIP2	2.534	7	X1	3.008	7	X1	3.041
8	X1	2.455	8	ACB	2.900	8	HA	3.041
9	HA	2.448	9	HA	2.451	9	ACB	2.929
10	RKIVE	2.195	10	Quantum	2.320	10	RKIVE	2.863
NM WB								
1	CALIC	3.910	1	ACB	5.998			
2	SZIP 1.05X	3.852	2	Quantum	5.484			
3	SZIP 1.04	3.838	3	SZIP 1.05X	5.079			
4	ACB	3.819	4	BZIP	4.979			
5	HA	3.813	5	BZIP2	4.879			
6	UFA	3.804	6	JAR	4.869			
7	X1	3.767	7	RKIVE	4.832			
8	BZIP	3.765	8	UFA	4.379			
9	RKIVE	3.708	9	SZIP 1.04	4.244			
10	BZIP2	3.702	10	YAC	4.228			
US								
1	CALIC	2.952	1	UFA	2.995			
2	UFA	2.893	2	SZIP 1.05X	2.994			
3	SZIP 1.05X	2.877	3	SZIP 1.04	2.956			
4	SZIP 1.04	2.873	4	CALIC	2.952			
5	RKIVE	2.841	5	ACB	2.888			
6	X1	2.788	6	RKIVE	2.848			
7	HA	2.774	7	X1	2.822			
8	ACB	2.770	8	BZIP	2.805			
9	BZIP	2.764	9	BZIP2	2.780			
10	BZIP2	2.737	10	HA	2.775			

References

[1] Kuba A, Alexin Z, Nagy A, Nyúl L, Palágyi K, Nagy M, Csernay L, Almási L. DICOM based PACS and its application in education. In: Proceedings of the 14th EuroPACS Meeting, Heraklion, Crete. 1996:46–49.

[2] Nagy A, Nyúl L, Kuba A, Alexin Z, Almási L. Problems and solutions: one year's experience with the SZOTE-PACS. In: Proceedings of the 15th EuroPACS Meeting, Pisa. 1997:39–42.

[3] DICOM, Digital imaging and communication in medicine. ACR-NEMA Standards Publication No. 300–1985. Washington, DC: National Electric Manufacturer's Association, 1995.

[4] Wu X. Lossless compression of continuous-tone images via context selection, quantization, and modeling. IEEE Transactions on Image Processing, 1997;6(5):656–664.

[5] Huang L, Bijaoui A. An efficient image compression algorithm without distortion. Pattern Recognition Letters, 1991;12:69–72.

[6] Robinson JA. Efficient general-purpose image compression with binary tree predictive coding. IEEE Transactions on Image Processing, 1997;6(4):601–608.

[7] Boutell T, Dilger A, Adler M et al. PNG (portable network graphics) specification. http://www.w3.org/TR/REC-png-multi.html (a stable document) 1996.

[8] Crocker LD. The portable network graphic format. Dr Dobb's Journal, 232, 1995;20(7):36–44.

[9] Paeth AW. Image file compression made easy. In: Arvo J, editor. Graphics gems II. San Diego, CA: Academic Press, 1991.

[10] Said A, Pearlman WA. Reversible image compression via multiresolution representation and predictive coding. Visual Communications and Image Processing-93, Proc SPIE, 1993;2094:664–674.

[11] Meyer B, Tischer P. TMW—a new method for lossless image compression. In: Proceedings of the Picture Coding Symposium, Berlin, Germany. 1997:217–220.

[12] Weinberger MJ, Seroussi G, Sapiro G. LOCO-I: a low complexity, context-based, lossless image compression algorithm. In: Storer J, Cohn M, editors. Proceedings of the IEEE Data Compression Conference. IEEE Computer Society Press, 1996:140–149.

[13] Golomb SW. Run-length encodings. IEEE Transactions on Information Theory, 1966;IT12:399–401.

[14] Rice RF. Some practical universal noiseless coding techniques. Technical Report JPL-79-22. Pasadena, CA: Jet Propulsion Laboratory, 1979.

[15] Pennebaker WB, Mitchell JL. JPEG: still image compression standard. New York: Van Nostrand Reinhold, 1993.

[16] Howard PG, Vitter JS. Fast and efficient lossless image compression. In: Storer J, Cohn M, editors. Proceedings of the IEEE Data Compression Conference. IEEE Computer Society Press, 1993:351–360.

[17] Tischer PE, Worley RT, Maeder AJ, Goodwin M. Context-based lossless image compression. The Computer Journal, 1993;36(1):68–77.

[18] Denecker K, Van Assche S, Philips W, Lemahieu I. State of the art concerning lossless medical image coding. In: IEEE ProRISC Workshop CSSP, 1997.

Timo Kaukoranta was born in Turku, Finland, in August 1968. He received the M.Sc. degree in computer science from the University of Turku, Finland, in 1994. Currently, he is a doctoral student at Turku Centre for Computer Science (TUCS), University of Turku, Finland. His primary research interests are in image compression and vector quantization.

Juha Kivijärvi was born in Salo, Finland, in August 1973. He received his M.Sc. degree in computer science from the University of Turku, Finland, in 1998. Currently, he is a doctoral student at Turku Centre for Computer Science (TUCS), University of Turku, Finland. His research interests are in vector quantization and clustering techniques.

Attila Kuba was born in Kecskemet, Hungary, in May 1953. He received the M.Sc. and Ph.D. degrees in 1976 and 1983, respectively. From 1976 to 1993, he was a scientific co-worker in the Kalmar Laboratory of Informatics, Jozsef Attila University, Szeged, Hungary. Since 1993 he has been the head of the Department of Applied Informatics. He lectures in the areas of computer graphics, image processing and medical image processing applications. His research interests are in the field of image processing, medical applications, reconstruction algorithms and discrete tomography.

Olli Nevalainen was born in Kankaanpää, Finland, in April 1945. He received the M.Sc. and Ph.D. degrees in 1969 and 1976, respectively. From 1972 to 1979, he was a lecturer in the Department of Computer Science, University of Turku, Finland. Since 1976 he has been an associate professor and from 1998 a professor in the same department. He lectures in the areas of data structures, algorithm design and analysis, compiler construction and operating systems. His research interests are in the broad field of algorithm design, including image compression, scheduling algorithms and production planning.

Laszlo Nyul was born in Papkeszi, Hungary, in September 1970. He received his B.Sc. (1992) and M.Sc. (1994) in Mathematics and Computer Science from Jozsef Attila University, Szeged, Hungary. He has been with the Department of Applied Informatics, Jozsef Attila University, since 1993. He participated and organized Summer Schools on Image Processing and taught university courses in several CS areas including Digital Image Processing and Computer Graphics. His research interests include image compression, image segmentation and quantification.

Tiina Ojala was born in Turku, Finland, in August 1972. She received her M.Sc. degree in computer science from the University of Turku, Finland, in 1997. Currently, she is an assistant in the Department of Computer Science, University of Turku, Finland. Her research interests are in medical image processing.