TMW — a New Method for Lossless Image Compression

Bernd Meyer, Peter Tischer
Department of Computer Science, Monash University
Clayton, Victoria, Australia, 3168

Email: [bmeyer,pet]@cs.monash.edu.au

Abstract

We present a general purpose lossless greyscale image compression method, TMW, that is based on the use of linear predictors and implicit segmentation. In order to achieve competitive compression, the compression process is split into an analysis step and a coding step. In the first step, a set of linear predictors and other parameters suitable for the image is calculated, which is included in the compressed file and subsequently used for the coding step. This adaption allows TMW to perform well over a very wide range of image types. Other significant features of TMW are the use of a one-parameter probability distribution, probability calculations based on unquantized prediction values, blending of multiple probability distributions instead of prediction values, and implicit image segmentation.

The method has applications beyond image compression. The work is also relevant to image segmentation and image comparison.

For image compression, the method has been compared to CALIC on a selection of test images, and typically outperforms it by between 2 and 10 percent, at the cost of considerably slower compression. In particular, a bitrate of less than 3.92 bpp has been achieved for the luminance band of the well known lenna image, compared to 4.05 bpp reported for CALIC in [Wu97].

1 Introduction

Recent years have seen many advances in the field of lossless coding of greyscale images. However, even the most successful method, CALIC is basically a variation on a few well known methods — namely predictive coding, context based selection of predictor coefficients and a fading-memory model for prediction error distributions.

In this paper, we present an approach to lossless coding of greyscale images that uses several fundamentally new concepts, such as the extraction of global image information, the use of multiple predictors with blending in the probability domain, and the use of unquantized predicted values

2 Two stage encoding

The proposed method uses a two stage encoding process. In the first stage, called Image Analysis, a set of model parameters is chosen in a way that minimizes the length of the encoded image. This set of model parameters is then used in the second stage, the Coding Stage, to do the actual encoding. Obviously, the chosen parameter set has to be considered part of the encoded image and has to be stored or transmitted alongside the result of the Coding Stage. Thus, the format of the encoded image is a two part message, as shown in figure 1.

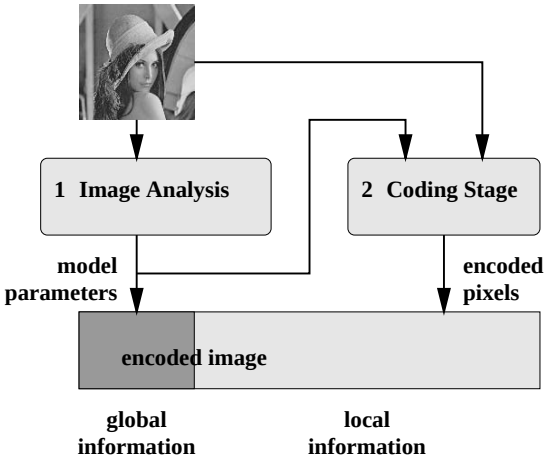

Figure 1: Principle of two part message used in proposed method

Wallace introduced the idea of such two part messages in [Wa68], asserting that the best choice of model parameters, i.e. the one that best describes the true characteristics of the data, is the one that led to the shortest overall message length. A choice of model parameters that overfits the data will result in an increase in the size of the first part of the message that is larger than the resulting savings in the second message part. Conversely, a choice of model parameters that underfits the available data will reduce the length of

the first message part by less than the resulting increase in the second part of the message.

When applied to image compression, the first message part can be seen as describing *characteristics* of the image, i.e. as containing global information. The second part of the message contains purely local information about the values of individual pixels. Ideally, the first part would capture the "essence" or "meaning" of the image, while the second part would only contain information about the particular values of noise for each pixel. While the proposed method does not yet achieve this ideal goal, it *does* constitute a significant step in this direction.

The two stage encoding process allows the proposed method to adapt to and perform well for an extremely wide range of image types. The results given in section 5 range from applying the proposed method to raytraced (i.e. practically noise free) images all the way to applying it to noisy 12 bit medical images.

Another interesting aspect of the two stage process is that it provides a measure for how similar the *characteristics* of two images are. In order to determine this measure for two images A and B, individual parameter sets for each image have to calculated as well as one parameter set covering *both* images. The difference between the total file sizes for

1. Encoding the images independently, with the individual parameter sets, and

2. Encoding both images with the one parameter set covering both of them

is the desired measure.

3 Model used

The model used in the proposed method is based heavily on linear predictors. Three different kinds of predictors are used

- pixel-predictors that predict a pixel value based on the pixel values of its causal neighbours;

- sigma-predictors that predict the *magnitude* of a pixel-predictor's prediction error based on the *magnitude* of that pixel-predictor's prediction errors for the causal neighbours;

- blending-predictors that predict how well suited a particular pixel-predictor is to predict a pixel value, based on how well the pixel-predictor performed on the causal neighbours.

The parameters of the resulting model are the weights of the predictors. The model seems to be powerful and flexible enough to adapt well to all images it has been tested on so far.

3.1 Multiple Linear Predictors

The proposed method uses linear pixel-predictors of the form

$$pred = \sum_{i=1}^{M} w_i pv_i$$

with M being the number of causal neighbours used for the prediction[1]and pv_i being the pixel value of the i-th causal neighbour.[2]The w_i are model parameters determined during the image analysis stage and their values are included in the first part of the encoded message.

One of the key ideas of the proposed method is to use not just *one* but *multiple* such pixel-predictors for each pixel.

The motivation behind this is that the correlation characteristics of a pixel with its causal neighbours typically are not constant over the whole image. Trying to model all pixels with the same predictor would necessarily result in that predictor being suboptimal for at least some areas of the image. [Wu97] addresses the problem by choosing from a set of fixed predictors according to heuristics, but still uses the prediction of only one predictor for the encoding. Also, due to the heuristics being hardcoded, the choices made may potentially be completely unsuitable for the image to be encoded (a good example for this is the CALIC result for the SHAPES image shown in figure 2).

Another approach to addressing the problem is the one used in [Se97]; Multiple predictors are used and their predictions are then blended together to give a single predicted pixel value. This method, however, applies a *single* probability distribution centered around the predicted value. Unless that probability distribution itself is bimodal, the resulting prediction cannot be bimodal, either, and thus potentially fails on edges. See figure 3 for a case in which no single predicted value would be appropriate, and a bimodal probability distribution is desirable.

The proposed method goes one step further than [Se97] and calculates a probability distribution $p(CP = x)$ for the current pixel having the

[1]This model parameter is not determined during the image anlysis stage, but instead is supplied by the user. However, it *is* included in the first part of the encoded message. A value of 12 has been found to be a good choice for images of size 512x512.

[2]We tried starting the sum at $i = 0$ with $pv_0 = 1$, i.e. allowing a constant term in the linear predictor. Invariably, it turned out that the best value for w_0 was very close to 0 and that the savings realized in the second part of the encoded message were smaller than the cost of including w_0 in the first part.

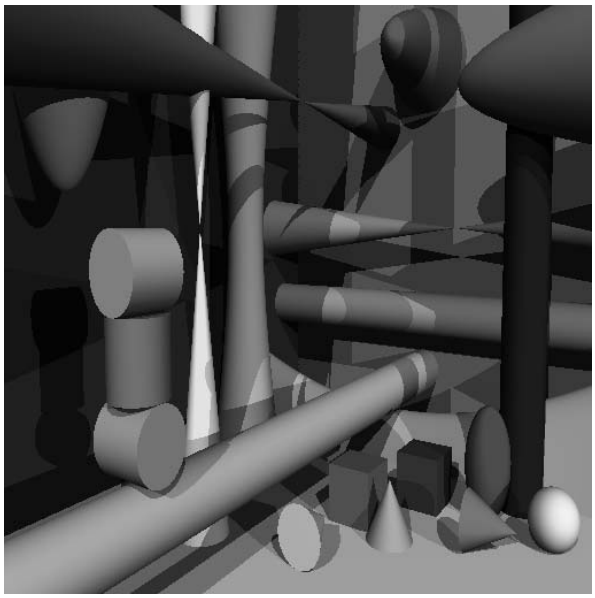

Figure 2: Synthetic image SHAPES

value x individually for each pixel-predictor. Then the *probability distributions* are blended together, resulting in a final probability distribution which is then used to encode the pixel. If the blending weights are varied for different areas of the image, this allows for the use of appropriate predictor blends. It also allows for the generation of complex probability distributions, such as the bottom right distribution in figure 3, from simple ones.

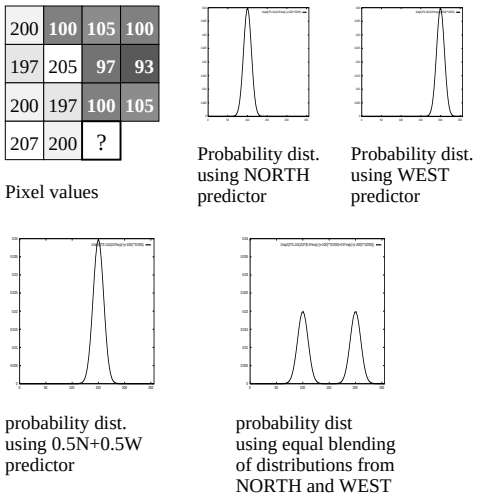

Figure 3: Effects of blending predicted values vs blending probability distributions

3.2 Probability Distribution for Prediction Errors

Probability models in predictive coding give an estimate on the likelihood of a particular prediction error occurring. As the prediction error distribution typically is not stationary over the whole image, most models contain local parameters based on information from a pixel's causal neighbourhood.

However, experience shows that only the magnitudes of prediction errors of close neighbours of a pixel are in any significant way correlated to the magnitude of the prediction error for said pixel. This means that only a limited amount of data is available for determining the local parameters, thus limiting the number of local parameters that can be determined in a statistically meaningful way — the fewer local parameters are used, the less information from non-local neighbours has to be used to determine the parameters. On the other hand, using fewer local parameters also severely limits the possible shapes for the distributions.

There is a range of waysfor dealing with this conflict. On one end of the range is the full histogram (which would require 255 local parameters for an 8 bit image). This is rarely used, as the high number of parameters would result in the use of data from non-local neighbours. Further down the range, other methods such like the ones used in SUNSET [La94] and SMB [Wo94] can be found, which are trading accuracy in describing the distribution shape for a reduction in local parameters (for 8 bit images, SUNSET has a few dozen local parameters, SMB has 9).

The probability model used in the proposed method is at the other end of the range — it uses a model with just a *single* local parameter. This allows us to use only data from close local neighbours. Experiments conducted during development seem to indicate that the positive effects from this high locality more than compensate for the negative effects of not being able to describe non-trivial distribution shapes. This might be due to the use of multiple pixel-predictors described in the previous section.

The distribution used is a variation of the t-distribution given by the formula

$$p(x < X) = K \int_{-\infty}^{X} (\frac{1}{1 + \frac{v^2}{2\sigma^2 N}})^N dv$$

with N currently hardcoded[3] at $\frac{20}{3}$ and K chosen in such a way that $p(x < +\infty) = 1$. The parameter σ is the local parameter estimated from the magnitude of the prediction errors for a pixel's causal neighbours — currently the 30

[3] This should really be a variable parameter included in the first part of the message. However, the current implementation of the function $p(x < X)$ uses a lookup table which would have to be recalculated each time N is changed. At the current time this is not practical.

causal neighbours within a Manhattan distance of 5 pixels are used.[4]

One interesting result of using a continuous distribution for the probability model is that it allows us to use predicted values without quantization. If the predicted value is $\widehat{x}$, then the probability $p(x = X)$of the pixel value x being the integer X can be expressed as

$$p(x = X) = K \int_{X-.5-\widehat{x}}^{X+.5-\widehat{x}} (\frac{1}{1 + \frac{v^2}{2\sigma^{2N}}})^N dv$$

This means that the probability $p(x = X)$ is a continuous function of the predicted value $\widehat{x}$, and that a prediction "between" two integers (e.g. 43.5) will give equal probability to both its neighbours (i.e. 43 and 44). This results in an improvement when compared to methods which are based on discrete distributions and thus require a quantization step which introduces a small amount of quantization noise.[5]

3.3 Calculation of Distribution Parameter

The parameter σ of the t-distribution is calculated using a sigma-predictor. The formula used is

$$\sigma^2 = \sum_{i=0}^{30} v_i * pe_i^2$$

with $pe_0 = 1$ and $pe_{[1..30]}$ the prediction error of the corresponding pixel-predictor for the 30 causal neighbours used. The v_i are model parameters and are included in the first part of the encoded message.[6]

3.4 Predictor Trust

Experiments with the distribution described in the previous two sections has shown that while generally performing well, it fails badly on some low noise images with sharp high contrast edges. The best example is the SHAPES image shown in section 5. In such images, a small value for σ is desirable for almost all pixels, however there are pixels for which the prediction error will be quite large. It turned out that the actual distribution of prediction errors could not be accurately modelled by the modified t-distribution. For *most* pixels, the distribution used was quite good, but for *some* pixels (those on edges encountered for the first time, and thus unexpectedly), all values for the prediction error were essentially equally likely. In order to account for this, a "trust" or "certainty" parameter was introduced for each predictor, giving a modified function for the probability

$$\widehat{p}(x = X) = c * p(x = X) + \frac{(1 - c)}{L}$$

where c is the trust parameter and L is the size of the possible range of values for x. In effect, the probability distribution described in the previous two sections is blended with a distribution which represents total ignorance.

It should be noted that one *global* parameter c is calculated for each pixel-predictor and included in the first message part. In contrast, the parameter σ is calculated for each individual pixel from local information, and only the weights used in that calculation are included in the first part of the message.

3.5 Calculation of Blending Weights

Once the probability distributions for all pixel-predictors have been calculated, they are blended together to give a single, combined distribution.

$$p_{all}(x = X) = \sum_{j=1}^{P} b_j \widehat{p_j}(x = X)$$

where P is the number of predictors used[7]and b_j is the blending weight for predictor j's distribution. The b_j are calculated based on the number of bits the predictor j would have needed to encode the pixels in the causal neighbourhood in a manner similar to Bayesian blending. The actual formulas are

$$\begin{aligned} ln\, c_j &= \sum_{k=0}^{Q} t_k ln\, \widehat{p_j}(x_k = pv_k) \\ b_j &= \frac{c_j}{\sum_{k=1}^{M} c_k} \end{aligned}$$

with $\widehat{p_j}(x_k = pv_k)$ being the probability the predictor j assigned to the actual value of the k-th causal neighbour, Q being the number of causal neighbours used for calculating the blending weights[8]and the t_k being model parameters

[4] The number of pixels used for this should be a variable parameter included in the first message part, too. However, experiments seem to indicate that 30 pixels is a good choice for almost all images tested. Therefore we chose to delay adding in the extra complexity.

[5] The main benefit of using a continuous distribution is described in section 4. The direct savings due to the elimination of the quantization noise are generally negligible — for the lenna test image, the savings are about 0.001 bits per pixel.

[6] Leaving out the constant term pe_0 and thus the parameter v_0 will result in serious degradation of the compression performance.

[7] Just like M, P is a parameter supplied by the user and included in the first part of the encoded message.

[8] The third and final user supplied parameter, together with M and P.

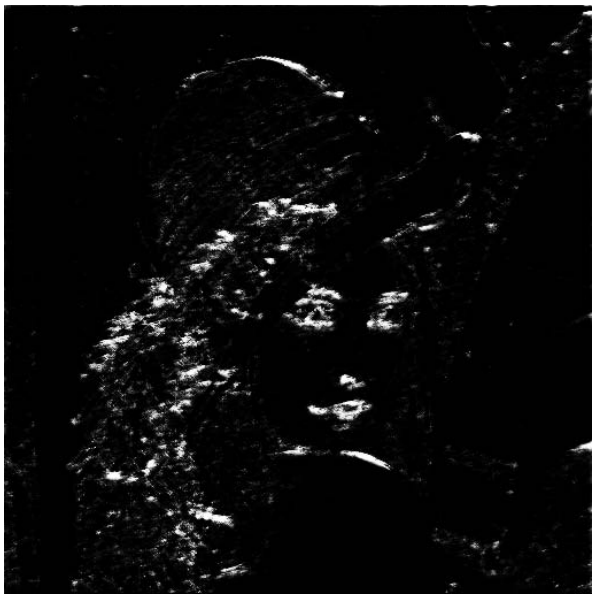

Figure 4: Partial assignment to predictor 10 for a sample encoding of lennagrey using a model with M=11, P=14 and Q=24. Bright areas stand for a high weight for the predictor, dark areas for a low weight.

Figure 5: Partial assignment to predictor 5

determined during the image analysis phase and included in the first part of the encoded message.

Although this is not a completely correct interpretation of the algorithm,this step might be understood by assuming that $ln\, c_j$ is an estimation for the number of bits predictor j would require to encode the current pixel, based on the number of bits required to encode the causal neighbours.

If the b_j are viewed as partial assignments of the current pixel to a class or segment j, the encoding process includes an implicit segmentation of the image. This segmentation, however, is forming segments based on correlation properties rather than visual impression, which means that although it is a good segmentation to achieve high compression, it does not correspond with human perception of what a segmentation should be. In particular, it seems that generally one predictor will cover all smooth areas, while all the other predictors are highly specialized (see figures 4 and 5).

3.6 Coding

Once p_{all} has been calculated, an arithmetic coder is used to do entropy coding according to this distribution. However, it is not necessary to calculate $p_{all}(x = X)$ for all L possible values of X (which would be extremely slow for 12 bit images). Instead, $p_{all}(X \leq x < Y)$ can be calculated, which allows for interval halving. This way, each pixel of a B-bit image can be encoded with B binary coding events.

4 Image Analysis

With the exception of the c parameters, all others are weights in some sort of linear predictor. As great care has been taken to ensure that all functions dependent on such parameters are of a continuous nature, the compressed filesize is a continuous function of the parameters as well.[9]Due to this continuous nature, the partial derivatives of the compressed message length with respect to the individual parameters can be calculated. This allows for the use of reweighted least squares [Bu94] to calculate parameter sets.

The calculation of parameters for a given image is an iterative process. Starting with an arbitrary set of parameters, in each step one of the three sets of predictor weights is optimized, while the other two remain unchanged. The three sets are the coefficients of the linear predictors, w, the parameters of the functions determining σ in the probability model, v, and the parameters of the functions determining the blending weights of probability distributions, t. The c parameters are adjusted in each step, using Newton approximation. All optimizations of parameter sets are based directly on minimizing the *encoded message length.*

After each step, an estimate for the resulting filesize is calculated, based on which the iteration can be stopped when a sufficiently good set of pa-

[9]This is also the reason for not including M, P and Q in the parameters calculated during the analysis phase — as they are integers, the filesize is a discontinuous function of these parameters.

image name	dimensions	bit depth	CALIC	TMW	ratio	description
balloon	720x576	8	2.83	**2.66**	1.06	outdoor scene with balloons
clin01	1301x1001	12	5.95	**5.80**	1.03	X-ray
lennagrey	512x512	8	4.11	**3.91**	1.05	well known test image
ref12b-0	512x512	12	2.77	**2.50**	1.11	CAT-scan
shapes	512x512	8	1.14	**0.76**	1.50	ray traced POVRAY sample scene
http://www.cs.waikato.ac.nz/~singlis/ratios.html						location of the following images
airplane	512x512	8	3.74	**3.60**	1.04	mountain scene with jet plane
baboon	512x512	8	5.88	**5.73**	1.03	extreme closeup on baboon face
boats	720x576	8	3.83	**3.61**	1.06	fishing boats at low tide
bridge	256x256	8	5.68	**5.59**	1.02	outdoor scene
camera	256x256	8	4.19	**4.10**	1.02	camera and grass
couple	256x256	8	3.61	**3.45**	1.05	sixties couple in dark room
goldhill	720x576	8	4.39	**4.27**	1.03	mountain village street scene
lena	512x512	8	4.48	**4.30**	1.04	different version of "lennagrey"
peppers	512x512	8	4.42	**4.25**	1.04	closeup on peppers

Table 1: Compression results for proposed method compared with results from CALIC

rameters has been found. [10]

5 Results

Table 1 lists file sizes (in bits per pixel) obtained by running both the proposed method and CALIC using arithmetic coding[11] on a variety of test images.

The proposed method achieves higher compression than CALIC for all images tested. The file sizes are taken from actual compressed files which were subsequently decompressed successfully.

6 Conclusion

We have presented a new algorithm for compression of lossless greyscale images which consistently achieves higher compression ratios than CALIC, and is more adaptable and flexible in handling different types of greyscale images than any other algorithm currently known. The presented methods also have relevance for the fields of image segmentation, image comparison and image abstraction, and we hope to be able to report on results from these fields in the future.

References

[Wu97] X. Wu and N. Memon, "Context-based, Adaptive, Lossless Image Codec", *IEEE Trans. on Communications*, vol. 45, no. 4, April 1997.

[Pi76] P. Pirsch and L. Stenger, "Statistical Analysis and Coding of Colour Video Signals", *Acta Electronica*, vol. 19, pp. 277-287, 1976.

[La94] G. Langdon and C. Haidinyak, "Context-dependent distribution shaping and parameterization for lossless image compression", *Application of Digital Image Processing XVII*, SPIE, pp. 62-70, 1994

[Bu94] C.S. Burrus, J.A. Barreto and I.W. Selesnick, "Iterative Reweighted Least Squares Design of FIR Filters", *IEEE Transactions on Signal Processing*, vol 42, #11, pp 2926-2936, November 94

[Wo94] R.T. Worley and P.E. Tischer, "An Alternative to Gray Coding for Bitplane Compression", *Australian Computer Science Communications Seventeenth annual computer science conf (ACSC-17) Christchurch, New Zealand*, vol 16, #1, pp 189-97, 1994

[Wa68] C.S. Wallace and D.M. Boulton, "An Information Measure for Classification", *Computer Journal* , vol 11, 1968

[Se97] T. Seemann, P.E. Tischer and B. Meyer, "History-Based Blending of Image Sub-Predictors", *International Picture Coding Symposium PCS97 conference proceedings.*

[10] What "sufficiently good" means depends on the application.

[11] As available from ftp://ftp.csd.uwo.ca/pub/from_wu/