

History-Based Blending of Image Sub-Predictors

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

Abstract

In this paper we describe a new lossless image compression algorithm called HBB. HBB has been designed to see how much further we can push the current one-pass DPCM approach to lossless image coding. Its features include a novel scan ordering and a complex adaptive history-based mechanism which blends sub-predictors for prediction. Its performance is not sensitive to the values of its parameters, and it extends easily to different pixel depths. We show HBB to out-perform current state-of-the-art lossless codecs, but at a computational cost.

1 Introduction

Most modern lossless DPCM image compression algorithms [1, 2, 3, 5, 6, 9] consist of some or all of the following steps: (1) one-pass raster scan traversal of pixels, (2) determination of a context for the current pixel based on neighbouring pixels, (3) locally adaptive prediction of the current pixel, (4) history-based error feedback to correct systematic errors in prediction, and (5) entropy coding of the final prediction errors.

In this paper we describe an image compression system called HBB (History Based Blending). HBB uses a similar architecture to that described above, but focuses on optimizing those components which are often neglected in the push to minimize storage and computational costs. In particular, a new scan ordering and a more complex adaptive non-linear predictor are used.

We show HBB to be very competitive with the current best lossless image compression codecs, beating systems such as CALIC [4, 5] and LOCO [6], but at a greater computational cost. The gains in compression are small compared to the effort expended. This suggests that large improvements should not be expected without moving toward a new approach to lossless image compression, such as TMW [7].

2 Scan Order

Traditionally, the image pixels are encoded in a raster scan order, as shown in Figure 1, traversing the image one row at a time, top to bottom, left to right. Although simple to program and only requiring few row buffers, it does not order pixels in a very spatially coherent manner.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

Figure 1: Raster Scan Ordering

High coherence is important for any history-based mechanisms present in an image compression scheme, such as error-feedback schemes and adaptive probability estimation for entropy coding. Ideally, the proper use of contexts for these mechanisms should remove the dependence on ordering. However, most context modeling techniques use only a small number of simple contexts, resulting in sub-optimal assignments. Thus we suggest that an improved scan ordering may still be beneficial.

1	2	4	7	10
3	5	8	11	13
6	9	12	14	15
16	17	19	22	25

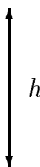


Figure 2: Rain Scan Ordering with $h = 3$

In HBB we use a *rain* scan ordering, shown in Figure 2. The rain ordering groups h rows together into blocks. Within each block, the pixels are traversed in a south-westerly direction, progressing from left to right, starting at the top-left corner of the block. This ordering varies its x and

y position at each step, unlike the mainly one-dimensional raster order. Using the compass point notation of Figure 3 to label the current pixel’s (CP) neighbours, we see that, at all times, the conventional causal neighbourhood is available to both encoder and decoder. We found using this ordering with values of h around 32 to improve results by up to 4% over the raster ordering.

NWNW	NNW	NN	NNE	NENE
WNW	NW	N	NE	
WW	W	CP		

Figure 3: Causal Neighbourhood Notation

3 Context Determination

Contexts result from a many-to-one function of the current pixel’s neighbourhood. They are used in image compression to switch between different models, in the hope that the conditioning will improve some estimation or prediction process. For example, CALIC uses 576 contexts for the error-feedback stage, and around 10 contexts for the entropy coding stage.

In HBB, we also use a small number of contexts based on the position of the leading 1 bit in the standard deviation σ of the six closest neighbouring pixels. For a z -bit per pixel image, this results in z contexts. Two extra contexts are also used, one for the case when $\sigma = 0$ and one for the border pixels, giving a total of $z + 2$ contexts.

Although this is a simple scheme, it has the advantage of being easily extendible to different values of z . We have found it to be very similar in performance to other techniques in the literature, even those which have been trained on image sets. In HBB, these contexts are used by both the prediction and entropy coding models.

4 Prediction Technique

4.1 Predictor Properties

The prediction stage is the most important one in terms of entropy reduction. It is also the stage at which a bad choice of predictor could risk amplifying and introducing noise into the predicted value, which can never be removed.

Let us assume our predictor uses a linear combination of neighbouring pixels N_i to produce a predicted value pv ; that is $pv = \sum_i a_i N_i$, where

a_i are the weights. We propose that on a local scale, a large proportion of an image consists of areas where the intensity varies smoothly, but with varying amounts of noise present.

To minimize the amount of noise introduced into pv we would like to keep $\sum |a_i|$ as small as possible [8]. For correct prediction in flat areas, we require $\sum a_i = 1$, and for correct prediction in planar regions, we need at least one $a_i < 0$ to properly estimate a gradient. These constraints can not all be satisfied simultaneously.

We suggest that the most important constraint on predictor performance is its behaviour in the presence of noise. It has been shown [9] that the performance of most simple static and adaptive predictors with negative a_i s will degrade as the noise level increases.

HBB’s predictor is designed to adapt to images with differing (1) noise levels, (2) smoothness, and (3) directional edges. It does this by blending the predictions of three “experts”, weighted by how well they have predicted in the past.

4.2 A HBB Unit

In its simplest form, a HBB unit blends the clipped¹ predicted values pv_i of n sub-predictors to produce a final predicted value $pv = \sum a_i \cdot pv_i$. The blending weights a_i are chosen to try to minimize the size of the prediction error $pe = pv - CP$ where CP is the current pixel. In fact, exact prediction in flat areas requires $\sum a_i = 1$, so only $n - 1$ weights need actually be determined.

Let us assume we have the set of all past pvs and CPs . Then we could choose $\underline{a} = (a_1, a_2, \dots, a_{n-1})^T$ to minimize the mean-squared error (MSE) between our past predictions and the actual pixels. This would involve solving an $(n-1) \times (n-1)$ linear system of the matrix/vector form $\underline{P} \cdot \underline{a} = \underline{q}$.

In HBB, \underline{P} and \underline{q} are not recomputed for each pixel, but rather a running total is kept and updated as $\underline{P}_{i+1} = \alpha \underline{P}_i + (1 - \alpha) \underline{P}_{new}$ (same for \underline{q}), where α is a fading factor ($0 < \alpha < 1$) which controls how quickly old observations are depreciated, and \underline{P}_{new} is the new observation. This is the *history* mechanism of the HBB predictor. By dividing the new \underline{P} and \underline{q} by $|pe|$, it is possible to bias the update process to minimize the mean absolute error (MAE). The MAE criterion is more robust than MSE, especially for 12 bit pixel data [10].

There are several reasons for blending *predictors* rather than neighbouring pixels. Predictors can span many pixels, hence reducing the order of equations to be solved, and it is possible to include non-linear predictors.

¹The pixels are clipped to the legal permissible pixel range $[0, 2^z - 1]$.

4.3 The HBB Predictor

It is possible to increase n and incorporate more predictors into the blend, but the number of operations required to solve for \underline{a} has complexity $\mathcal{O}(n^3)$. This can be reduced by using a cascaded approach. In HBB we used three different $n = 3$ “child” units, the outputs of which were fed into a separate $n = 3$ “parent” unit. In this case we are a factor of 8 better off. For the three child units, we chose groups of sub-predictors suited to (1) noisy regions, (2) smooth gradients, and (3) strong edges, in the hope of isolating these three types of image behaviour. The actual sub-predictors used are shown in Table 1.

Unit 1 - “Noise”	
1	$(\mathbb{W} + \mathbb{N})/2$
2	$(2\mathbb{W} + \mathbb{N} + \mathbb{NE})/4$
3	$(\mathbb{W} + \mathbb{N} + \mathbb{NW} + \mathbb{NE})/4$

Unit 2 - “Smooth”	
1	$\mathbb{W} + \mathbb{N} - \mathbb{NW}$
2	$2\mathbb{W} - \mathbb{WW}$
3	$2\mathbb{N} - \mathbb{NN}$

Unit 3 - “Edges”	
1	\mathbb{W}
2	\mathbb{N}
3	\mathbb{NE}

Table 1: Sub-Predictors used in HBB

5 Error Feedback

Both CALIC and LOCO use error feedback to help correct systematic errors in their relatively simple predictors. That is, a context sensitive estimate of the prediction error \hat{pe} is added to pv to try and centre it at zero.

CALIC’s “Mean Adjusted Prediction” (MAP) uses the observed sample mean of past prediction errors for the estimate, conditioned on 576 contexts. LOCO uses 1094 contexts and an approximation to the sample median as the estimate. These error feedback schemes can significantly improve the performance of a poor predictor [9].

In HBB we utilize a MAP scheme similar to CALIC’s to correct predicted values pv . We use 2^6 contexts, a bit vector value D computed as

$$D = d_0 d_1 d_2 d_3 d_4 d_5 \quad (1)$$

$$\begin{aligned} d_0 &= b(pv, \mathbb{N}) & d_1 &= b(pv, \mathbb{W}) \\ d_2 &= b(pv, \mathbb{NW}) & d_3 &= b(pv, \mathbb{NE}) \\ d_4 &= b(pv, \mathbb{WW}) & d_5 &= b(pv, \mathbb{NN}) \end{aligned} \quad (2)$$

where $b(x, y) = 0$ if $x < y$ and 1 if $x \geq y$. MAP is used at two different points in HBB. Firstly, the 9 initial sub-predictors (3 predictors into each of the 3 child units) are each separately corrected, using their own pv to compute D . Secondly, the final output of the parent unit is also corrected in the same way.

6 Coding Model

The predicted value from the previous stages is used to form a prediction error, which can take on values $-2^z - 1$ to $+2^z - 1$. This unnecessarily doubles our alphabet size. We used the following re-mapping strategy to keep the prediction errors in the range $[-2^{z-1}, +2^{z-1} - 1]$, while still maintaining decodability:

$$pe' = \begin{cases} pe + 2^z & \text{if } pe < -2^{z-1} \\ pe - 2^z & \text{if } pe \geq +2^{z-1} \\ pe & \text{otherwise} \end{cases} \quad (3)$$

The prediction errors are entropy coded using the SMB (Sign Magnitude Bitplane) [11] histogram approximation approach. SMB uses only $z + 1$ binary arithmetic coding events to encode symbols from an alphabet of 2^z , and is similar to the Sunset approach [12]. First, one binary event states whether $pe = 0$ or not. If $pe \neq 0$, further events encode the sign bit and the bit planes of the binary representation of the error magnitude.

The advantage of SMB is that it is well suited to zero-mean, Laplacian distributed prediction errors, has a very short initialization period, and can adapt quickly to non-stationary statistics. Adaptation is achieved by halving the binary frequency counters whenever the total exceeds 255. Additionally, SMB is easily extended to larger pixel depths as the number of model parameters to be estimated only increases linearly with z . The loss in coding efficiency due to the bit plane approach is made up by its ability to adapt quickly.

7 Algorithm Summary

1. Compute the pv for each of the 9 sub-predictors.
2. Form an order-6 MAP context for each sub-predictor, and accordingly correct each pv to pv' .
3. Form a prediction/coding context C based on the standard deviation of the 6 closest neighbouring pixels.
4. For each of the 3 HBB child units in C , compute the output pv'' using its 3 corrected sub-predictors.

5. Compute pv''' by feeding the child pv'' 's into the parent unit in context C.
6. Form another order-6 MAP context and correct pv''' to get pv'''' .
7. Compute pe as the difference between pv'''' and the actual pixel.
8. Normalize pe to give pe' .
9. Encode pe' using SMB context C.
10. Move to next pixel in the rain scan order.

8 Results

Tables 2 to 4 give results for the original 8 bpp (bit per pixel) greyscale JPEG test set, another larger set of well-known 8 bpp greyscale images², and a varied set of 12 bpp medical images.

The default parameters were used for CALIC³ and LOCO⁴. For HBB, we used $h = 32$ for the rain scan order, $\alpha = 0.98$ for the predictor's history fade factor, and the predictor update process was biased to minimize the MAE, as described in Section 4.2.

9 Discussion

The results show HBB to be very competitive with CALIC, beating it on the majority of the images. LOCO performs worse than both CALIC and HBB on all images, but this is to be expected as LOCO has traded compression for faster throughput and lower complexity.

For all the 8 bpp images, HBB is about 0.037 bpp better off on average. However, this comes at a computational cost estimated at one order of magnitude higher than CALIC, which in turn is one order of magnitude more complex than LOCO. The improvement is a higher 0.09 bpp for the 12 bpp medical images, which vary from very noisy (**c00156**, **f005**) to very smooth (**ref12b-0**, **skullq7-0**).

The improvements can be attributed to different features of HBB, depending on the images. For the smooth 12 bpp images, it is probably due to the improved scan ordering in conjunction with the adaptive entropy coder. For the noisy 12 bpp images, it is most likely the prediction stage which ensures that the amount of noise introduced into the predictors is minimized. One interesting result is the 0.13 bpp improvement on the **barb** image. We attribute this to the "edge" unit picking

²Some of the 8 bit images were obtained using anonymous FTP from <ftp://ip1.rpi.edu/pub/image/still/>

³Arithmetic coding CALIC executables were obtained from ftp://ftp.csd.uwo.ca/pub/from_wu/v.arith/.

⁴The LOCO-I/JPEG-LS V.0.823X executables were obtained from <http://www.hp1.hp.com/loco/>.

up on the strong 45° and 135° edges on the chair combined with the scan order following the same direction.

Preliminary experiments have shown that if the *average a* vector (the blending weights) over the whole image is computed, that the resulting predictor is very close to the bit-rate that a static image-specific *least-entropy* linear predictor [10] would give. This could be used as cheap way to compute *near* least-entropy predictors, or as a better starting position for a traditional search.

10 Acknowledgments

The authors would like to thank Binary Imaging Company for the **chestxray** image, Siemens Europe for the **ref** and **skull** images, and Lewis Berman of the National Library of Medicine - National Institutes of Health for the **c00156** and **100156** images.

References

- [1] N. Memon, V. Sippy, X. Wu, "A Comparison of Prediction Schemes for a New Lossless Compression Standard", *1996 International Symposium on Circuits & Systems*, Vol. II, 1996, pp. 309–312.
- [2] P. G. Howard, J. S. Vitter, "Fast and Efficient Lossless Image Compression", *Proc. Data Compression Conference*, 1993, pp. 351–360.
- [3] G. K. Wallace, "The JPEG Still Picture Image Compression Standard", *Communications of the ACM*, Vol. 34, No. 4, April 1991, pp. 30–34.
- [4] X. Wu, N. Memon and K. Saywood, "A Context-based, Adaptive, Lossless/Nearly Lossless Coding Scheme for Continuous-tone Images", ISO/IEC JTC 1/SC 29/WC 1 document, No. 202, July 1995.
- [5] X. Wu, N. Memon, "CALIC - A Context Based Adaptive Lossless Codec", *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol. 4, 1996, pp. 1890–1893.
- [6] M. J. Weinberger, G. Seroussi, G. Sapiro, "LOCO-I: A Low Complexity, Context-Based, Lossless Image Compression Algorithm", *Proc. Data Compression Conference*, 1996, pp. 140–149.
- [7] B. Meyer, P. Tischer, "TMW: A New Method for Lossless Image Compression", *Proc. Picture Coding Symposium*, Berlin, 1997.

Image	Width	Height	CALIC	LOCO	HBB
balloon	720	576	2.83	2.90	2.80
barb2	720	576	4.53	4.69	4.48
barb	720	576	4.41	4.69	4.28
board	720	576	3.56	3.68	3.54
boats	720	576	3.84	3.93	3.80
girl	720	576	3.77	3.93	3.74
gold	720	576	4.39	4.48	4.37
hotel	720	576	4.25	4.38	4.27
zelda	720	576	3.75	3.89	3.72
			3.93	4.06	3.89

Table 2: Original 8 bpp JPEG Test Set Results (bpp)

Image	Width	Height	CALIC	LOCO	HBB
airfield	512	512	5.47	5.57	5.49
airplane	512	512	3.54	3.61	3.51
camera	256	256	4.19	4.33	4.14
couple	512	512	4.58	4.68	4.58
crowd	512	512	3.76	3.92	3.78
harbour	512	512	4.44	4.50	4.42
lax	512	512	5.63	5.76	5.60
lenna	512	512	4.10	4.24	4.07
man	512	512	4.36	4.51	4.36
mandrill	512	512	5.88	6.04	5.80
peppers	512	512	4.20	4.29	4.15
sailboat	512	512	4.69	4.77	4.62
woman1	512	512	4.54	4.67	4.50
woman2	512	512	3.20	4.30	3.11
			4.47	4.66	4.44

Table 3: Other 8 bpp Image Results (bpp)

Image	Width	Height	CALIC	LOCO	HBB
c00156	1463	1755	5.82	5.93	5.71
chestxray	1024	1024	5.25	5.42	5.18
f005	1385	1799	6.25	6.35	6.16
l00156	2048	2487	5.87	6.03	5.78
ref12b-0	512	512	2.77	2.90	2.68
ref12q5-0	512	512	2.83	2.95	2.66
skullpa-0	512	512	3.32	3.44	3.33
skullq7-0	512	512	3.33	3.40	3.23
			4.43	4.55	4.34

Table 4: Medical 12 bpp Test Set Results (bpp)

- [8] P. Pirsch, "A New Predictor Design for DPCM Coding of TV Signals", *ICC Conference Report (International Conf. on Communications)*, Seattle, WA, 1980, 31.2.1–31.2.5.
- [9] T. Seemann, P. E. Tischer, "Generalized Locally Adaptive DPCM", Technical Report No. 97/301, Monash University, March 1997.
- [10] P. E. Tischer, "Optimal Predictors for Image Compression", Technical Report 94/189, Monash University, 1994.
- [11] R. T. Worley, P. E. Tischer, "An Alternative to Gray Coding for Bit-Plane Compression", *ACSC 17*, Christchurch, New Zealand, 19–21 January 1994.
- [12] G. G. Langdon, "Sunset: A Hardware Oriented Algorithm for lossless compression of grayscale images", *Medical Imaging V : Image Capture, Formatting and Display*, Volume 1444, SPIE, 1991, pp. 272–282.