Context Models for Palette Images

Paul J. Ausbeck Jr.
Research Fellow
Department of Computer Engineering
University of California
Santa Cruz, CA 95064
paula@cse.ucsc.edu

Abstract

A family of two dimensional context models appropriate for palette images is described. The models are designed for use with a binary arithmetic coder. A complete image encoder/decoder using three models from the family is disclosed. The new coder is compared against five alternate coding methods: JBIG bit plane coding, CALIC predictive coding, CALIC plus palette ordering, and two dictionary methods, GIF and PNG. The aggregate compression achieved by the new method on a corpus of fifteen palette images is 25% better than the best alternate method. The appropriateness of the corpus is validated by the similar aggregate compression achieved by the alternate methods even though compression varies widely from image to image. Remarkably, the new method achieves 20% better compression than a composite coder formed from the best alternate method for each image.

1. Introduction

A palette image is composed of two components: color information contained in a lookup table or *palette*, and image information composed of a series of palette indices. In modern computer systems, palette images are ubiquitous. For example, the user interface elements of most windowing operating systems are composed of palette images. Black and white documents are a simple form of palette image. Almost every page on the Worldwide Web contains one or more palette images. Figure 1 is a grayscale version of the palette image serving as the banner of the Web site http://www.yahoo.com in October of 1997.

Figure 1
Typical Palette Image

In spite of their widespread use, an image model tailored for palette images has yet to be devised. Palette images generally contain too few colors to make effective use of the linear predictive models used in JPEG–LS and contain too many colors to avoid the sparse context problem that arises when using neighborhood color models such as those of JBIG. Table 1 shows the results of applying various coding methods to the grayscale

image of Figure 1. The first method is to separately code the image bit planes with JBIG, the second method is CALIC predictive coding[1], and the last method is dictionary coding via GIF. Perhaps surprisingly, the one dimensional LZW model used in GIF performs better than either of the alternate two dimensional models. For palette images that have not been converted to grayscale, the superior performance of dictionary coding methods continues to hold true. For typical palette images a text model turns out to perform better than the wrong image model!

Uncoded	Bit Planes	Predictive	GIF
27,182	9,266	8,348	6,923

Table 1
Motivational Coding Example

This paper proposes a new family of *piecewise–constant* image models that are optimized for describing palette images. The new models contain efficient mechanisms for describing pixel locations where color changes occur and a novel method for guessing unknown colors using known surrounding colors for probability estimation. The models are fully adaptive and have relatively few parameters.

Three specific members of the family that efficiently represent a wide range of palette images are also described. A new image coder based on these models is then compared to several alternate techniques on a corpus of typical palette images. Analysis and conclusions follow the experimental results.

2. A Palette Image Model

Whether synthetically produced or derived from continuous tone pictures, palette images are characterized by the following three properties.

- They tend to contain far fewer colors than pixels.
- Pixels of the same color tend to be contiguous.
- The color of a pixel is statistically related to its surrounding colors.

The first two palette image properties lead to a characterization of such images as *piecewise–constant*. For typical images, the constant color pieces or *domains* can be either rectilinearly or diagonally connected. If *boundaries* between domains are known, establishing the color of one pixel in a domain establishes the color of all pixels in the domain. One role of a piecewise–constant image model is to efficiently describe boundaries between arbitrarily shaped domains.

The statistical relationship of pixel colors propounded in property three is, for typical palette images, **not** a linear predictive relationship. Because of the lack of a simplifying model property, the sparse context problem makes it prohibitive to keep track of complete neighborhood color statistics. In an image with 256 colors, even a first order color model requires 256×255 model parameters. However, complete statistics are often unnecessary.

Given a first order context, typically only one or a few following colors predominate. This leads to the idea of using previously determined colors in the same context as *guesses*. To maintain one guess for each context of a 256 color first order model, requires only 256 model parameters.

Boundary delineation and color guessing can only determine whether or not an image pixel has the same color as some other pixel. Introduction of new colors or *innovations* into the image model requires some other mechanism. The exact form of this mechanism is unrestricted and can be accomplished via a standard technique such as linear prediction.

A language for describing piecewise–constant images is shown in Table 2. It consists of a sequence of questions posed by a piecewise–constant image decoder. The questions are either binary decisions or can easily be decomposed into a sequence of binary decisions. As such, the language is designed for use with a binary arithmetic coder.

Q1	Is the current pixel's color identical to that of a specified rectilinearly connected neighbor?
Q2	Is the current pixel's color identical to that of a specified diagonally–connected neighbor?
Q3	Is the current pixel's color identical to a guessed value?
Q4	What is the current pixel's color?

Table 2
Piecewise–Constant Image Model Language
Elements

The piecewise–constant modeling language is designed so that affirmative answers to **Q1**–**Q3** eliminate the need for a decoder to pose **Q4**. On images that match the model, **Q1**–**Q3** are answered predominantly in the affirmative and dominate both coding time and resulting codestring length. When **Q1**–**Q3** all fail the model reverts to the mechanism used to answer **Q4**.

Any mechanism can be used to answer **Q4** including linear predictive coding. In such a case the piecewise–constant model can be viewed as a preprocessor to a linear predictor. If adaptive context models are used for **Q1**–**Q3**, the worst case compression overhead introduced by the piecewise–constant model is largely proportional to the number of parameters used to estimate probabilities for **Q1**–**Q3**. The worst case performance overhead is proportional to the number of times that **Q1**–**Q3** are posed. Both of these topics are covered more extensively in the following discussion.

3. Context Models for the Piecewise–Constant Language

An excellent context model for **Q1** was proposed by Tate[2] in his work on coding edge maps resulting from image segmentation of satellite imagery. The technique is a variation of the neighborhood template model of Langdon[3]. Instead of using neighboring pixels on a black/white image, neighboring edge segments are used to determine a decision context. The construction is shown in Figure 2.

To completely specify an image partition, two edge locations are assigned to each pixel. For boundary discovery scans in normal raster order, either the north and west or the south and east edges are assigned to each pixel and the presence or absence of an edge segment is determined by a **Q1** decision. The following discussion defaults to north and west assignment.

For each location **L**, **Q1** is posed against the vertical dotted edge location on Figure 2 under the context determined by the eight solid edge segments on the figure. The western edge together with the same surrounding edges form a context to pose **Q1** against the northern edge. The number of model parameters associated with this scheme is $256 + 512 = 768$, and the number of decisions is two per pixel.

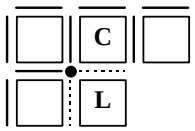

Figure 2
Rectilinear
Context

The number of model parameters and decisions of the Tate model can be reduced by taking advantage of boundary connectivity constraints. Since in a piecewise–constant model the boundaries separate contiguous domains, each end of a boundary segment must connect with another edge segment. Because of this constraint, many pixel locations require only one decision to determine domain connectivity.

For example, if the answer to **Q1** for the western edge of pixel **L** is yes, and no other edges impinge on the lattice intersection labeled with a dark circle associated with **L** on Figure 2, the answer to **Q1** for the northern edge of the location must always be yes and a coding decision need not be made. The average number of **Q1** decisions needed to establish rectilinear connectivity in an image varies between one and two decisions per pixel and for typical images approaches one. The connectivity constraints also decrease the number of model parameters associated with **Q1** to 512.

On bilevel images, connectivity constraints are even more severe. Since only two colors are available, a boundary lattice can never have an odd number of impinging edges, only zero, two or four. This constraint reduces the number of **Q1** decisions per pixel to one and the number of **Q1** model parameters to 256. Since the number of model parameters is halved on two color images, and since there are only two possible colors, better compression is achieved by including the color of one surrounding pixel in the context model. For example, if the color of the pixel labeled **C** on Figure 2 is added to the **Q1** context model, the number of model parameters is restored to 512. This formulation is equivalent to a nine pixel color model.

Q2 decisions are used to establish diagonal connectivity in a piecewise–constant model. Diagonal connectivity is only defined at lattice intersections where there is no rectilinear connectivity. Figure 3 shows the two causal orientations of diagonal connectivity at a lattice intersection, **L**, that has four impinging boundary segments. Each potential diagonal connection requires one **Q2** decision. The number of diagonal connections considered by the model drops quite rapidly as the edge density decreases and is typically less than 0.5 decisions per pixel.

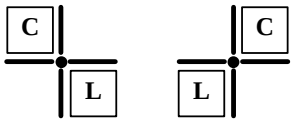

Figure 3
Diagonal Contexts

Domain color is usually more important than domain shape in conditioning **Q2** decisions. For this reason **Q2** decisions should only be made once the color to be propagated across an diagonal connection is known. Connection orientation is also an important conditioning criteria in many images. Using both orientation and color in a context model for **Q2** decisions requires two model parameters for every color used by an image. On Figure 3 the two orientations are represented by the left and right glyphs and the propagating color is labeled **C.**

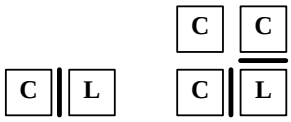

Figure 4
Guess Contexts

Color guessing, language element **Q3,** is designed to model the neighboring color relationships in an image while using a controlled number of model parameters. A guess is simply some color that has occurred previously in the coding process. The size of a guess model is proportional to D^S where D is the palette depth of the image and S is the number of neighboring colors used in the model. To maintain a reasonably sized model, the number of neighboring domain colors used to condition **Q3** must be carefully limited. For 256 color images it is usually only profitable to include one neighboring color in the conditioning context. The left glyph of Figure 4 shows a known pixel, **C**, used as a conditioning context for the unknown pixel, **L**, to its east. The right glyph of the figure shows three neighboring colors used as a conditioning context. The three color configuration is normally only useful for palette images of depth four (sixteen possible colors).

The exact size of a guess model is determined by the number of guesses for which statistics are maintained. One possibility is to maintain statistics for every possible color occurring in each context. The size of this straightforward guess model is D^{S+1}, no different from a complete neighborhood color model. With such a large model, many guesses are not very useful in determining color. Compression suffers because of the large number of mostly useless parameters to be learned. Coding speed suffers because a large number of mostly useless decisions are made.

One way to solve both of these problems is to limit the number of guesses maintained simultaneously by the model to some fixed number. When limiting guesses, a mechanism is needed to maintain only *good* guesses: guesses that are mostly correct. One way to achieve this is through *guess competition* in a *guess pool*.

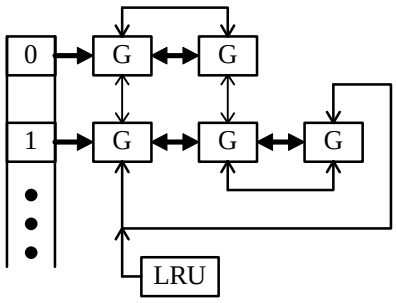

Figure 5
Guess Pool Structure

Any competitive mechanism can be used in the pool. One such mechanism is the least recently used (LRU) chain. In this application a context is moved to the front of the LRU any time its associated guess is correct. When a new guess is added to the pool and the pool has reached its maximum size, the guess at the end of the LRU chain is sacrificed. Figure 5 shows the guesses of a guess pool chained from lists determined by a context identifier. The average number of guesses per context is the size of the guess pool divided by the number of guess contexts. It should be emphasized that the guess pool is global. Competition occurs both between guesses in the same context and guesses in other contexts. The complete guess pool operation is as follows:

- When **Q4** decision is made, the result is added to the head of the guess pool LRU and to the tail of the appropriate decision context's guess chain.
- When a **Q3** decision is correct, its associated context is moved to the head of its guess chain and to the head of the guess pool LRU.

- When the guess pool is full and a guess must be sacrificed, it is removed from both its associated guess chain and the end of the LRU.
- The statistics of a **Q3** context are only updated when they are actually used to make a decision. No attempt is made to keep accurate conditional probabilities.
- The statistics of a sacrificed **Q3** context are decayed and kept as a prior for its new role.
- Guesses that are known to be impossible from prior **Q1** and **Q2** decisions are ignored.

4. An Example Piecewise–Constant Image Coder (PWC)

The piecewise–constant model components can be assembled in various ways to handle different types of images. The following discussion describes a coder that uses three variations on the piecewise–constant model to efficiently compress palette images of depth one, four, and eight. The number of parameters used by each model variation is summarized in Table 3.

Palette Depth	Model Parameters			
	Q1	**Q2**	**Q3**	**Q4**
1	512	0	0	1
4	512	0	256	64
8	512	512	1024	2048

Table 3
Model Parameters

The depth one model used by the example coder uses the single color augmented eight edge **Q1** context. The model does not use **Q2** and **Q3** since they provide no additional image information. **Q4** is only needed to determine the color of the first two domains in the image. Thereafter, color is propagated via color swap. Because **Q4** is only posed two times, the simplest **Q4** model, one parameter, is used.

At depth four, the eight edge model without a neighboring color is used for **Q1**. The three neighboring color model is used for **Q3**. Since this model has enough extent to provide most of the same information provided by **Q2**, **Q2** is not used. Although there are 4096 contexts, the size of the **Q3** guess pool is only 256, providing an average of 0.0625 guesses per context. Since on typical images the majority of possible neighboring color combinations never occur, the average number of guesses associated with each active context is much higher. A predictive model with four activity classes similar to that described by Don Speck[4] is used for **Q4**. The northern pixel is used for prediction and the magnitude of the maximum difference in the western, northwestern, and northern pixels is used to establish the activity class. The depth four coder performs a first raster scan and establishes rectilinear–connectivity via **Q1**. A second scan establishes color information via **Q3** and **Q4**.

At depth eight, the edge only model is again used for **Q1**. The orientation augmented single neighboring color model is used for **Q2**. The single neighboring color model with the neighboring color obtained from the western pixel is used for **Q3**. The size of the **Q3** context pool is 1024, providing four guesses per color on average. The predictive model used for **Q4** has eight activity classes. The prediction and activity class are established as with the depth four coder. The two pass operation of the coder is also the same.

It should be noted that for many images the **Q3** mechanism in the piecewise–constant model obviates the need for a sophisticated model for **Q4**. In fact in the experiments of

the next section, a constant prediction in one activity class produced essentially the same results as neighboring color prediction and multiple activity classes. The more sophisticated **Q4** mechanism only yields improvement if the image palette has significant linear predictive structure that is not caught by the piecewise–constant model.

5. Experiments

A standard corpus for evaluating palette image coding models is not currently available. Therefore significant effort was expended compiling a fairly representative group of test images. The resulting fifteen images are summarized in Table 4.

Image	Depth	Colors	Size	Source
benjerry	8	48	466x60	WWW – Ben and Jerry's ice cream ad
books	4	7	179x318	MS Access – bookshelf motif
ccitt01	1	2	1728x2339	CCITT – reference document #1
cmpndn	8	223	512x768	JPEG–LS cmpnd1 nearest color reduced
cmpndu	8	246	512x768	cmpndn plus error diffusion
flax	4	3	170x102	MS Access – textile simulation
gate	8	84	564x108	SF Chronicle home page – banner
music	4	8	111x111	MS Office clip art – music motif
netscape	8	32	612x100	Netscape home page – banner
pattern	1	2	135x137	MS Access – noisy tiled pattern
sea_dusk	8	46	484x325	MS Access – synthetic sky & city
stone	4	3	169x133	MS Access – stone texture
sunset	8	204	640x480	MS Scenes – sunset, ice and mountains
winaw	4	10	633x465	pcAnywhere – startup banner
yahoo	8	229	460x59	Yahoo home page – banner

Table 4
Palette Image Corpus

The number of colors used in the test images varies from two to 246. The test set contains completely synthetic images, error diffusion and nearest color quantized images, and compound images containing both synthetic and natural elements. An attempt was made to balance the number of bits of each source type. Some particular emphasis was placed on obtaining palette images from popular sites on the Worldwide Web. The **yahoo** image of line thirteen was shown previously in grayscale form in Figure 1.

Table 5 shows the results of the application of the piecewise–constant coder (PWC) described here and four alternate coding methods to the test images. All data are file byte counts and include header information. The header of the piecewise–constant file is 16 bytes long. The GIF, PNG, and PWC results contain either 6, 48, or 768 bytes of palette color information. The JBIG Planes and CALIC results do not.

Image	JBIG Planes	GIF	CALIC	CALICO	PNG	PWC
benjerry	3,697	4,401	4,193	4,193	4,571	3,018
books	13,343	11,177	14,033	9,740	10,831	8,624
ccitt01	12,884	38,862	18,146	18,146	28,910	12,892
cmpndn	69,098	62,682	56,951	52,085	56,397	40,790
cmpndu	81,705	76,759	71,109	63,430	69,438	54,550
flax	226	846	379	194	318	175
gate	25,589	23,313	20,555	18,494	20,124	16,318
music	1,563	1,987	1,648	1,219	1,647	744
netscape	16,489	17,442	14,302	13,746	15,879	11,326
pattern	1,190	1,782	1,755	1,755	1,928	1,203
sea_dusk	1,647	6,362	1,446	1,069	2,540	1,289
stone	4,460	4,753	8,440	4,917	3,906	3,992
sunset	121,020	100,186	113,710	82,109	81,794	52,608
winaw	27,830	18,559	21,686	16,886	18,732	11,439
yahoo	8,126	7,097	6,884	6,884	6,275	4,492
total	388,867	376,208	355,237	294,867	323,290	223,460

Table 5
Comparison of Coding Methods

The results in the column labeled JBIG Planes were obtained by applying JBIG[5] to the bit planes of each image. Empty and duplicate planes were eliminated from each image total. For each image only enough planes were kept to uniquely determine the number of colors used. To correct for duplicate file headers, eighty bytes were subtracted from each JBIG total for each retained plane other than the first.

The third column of results were obtained using the arithmetic CALIC[1] predictive coder. Of the JPEG–LS contributors, CALIC appears best suited to palette images. The publicly available implementation obtained from http://www.csd.uwo.ca/~wu/index.html was used. The results labeled CALICO were obtained by combining CALIC with palette ordering[6]. The GIF and PNG results were obtained using the Paint Shop Pro image utility[7].

Image	JBIG Planes	GIF	CALIC	CALICO	PNG	PWC
pc	123,788	376,482	205,000	186,157	225,898	98,997

Table 6
Coding Results for the JPEG–LS **pc** Image

The data of Table 6 are coding results for a palletized version of the JPEG–LS test image, **pc**. They are shown because **pc** is a widely known test image, contains only six colors, and can be converted to a depth four palette image without loss. This image is so large that it would skew the corpus, so it is shown separately. The proposed JPEG–LS standard with palette feature codes this image using 322,628 bytes. JBIG bit planes does well on this image but is still 20% worse than PWC.

Table 7 shows some key PWC coding statistics for the fifteen images of the test corpus. The column labeled “Domains” is a count of the number of rectilinearly connected domains in each image. The columns labeled with **Q?** are counts of the number of each type of decision made during coding. The columns labeled H? are the average entropies for each decision type.

As described previously, **Q2** decisions are only made for depth eight images. **Q3** decisions are only made for depths greater than one. Additionally, depth one images require only two **Q4** decisions. Due to the method used to obtain the measurements, the accuracy of the tabulated entropy values degrades substantially for those decision buckets containing fewer than several hundred decisions.

Image	Domains	**Q1**	H1	**Q2**	H2	**Q3**	H3	**Q4**	H4
benjerry	1704	32647	0.330	1022	0.845	7257	0.645	257	6.008
books	14527	84877	0.663	0	0.000	18418	0.593	453	2.808
ccitt01	1513	4041791	0.025	0	0.000	0	0.000	2	1.000
cmpndn	34445	462288	0.324	40380	0.592	155665	0.471	10603	6.873
cmpndu	61019	486937	0.297	88124	0.690	247177	0.480	15554	6.825
flax	16328	33906	0.018	0	0.000	16328	0.011	14	4.571
gate	15497	87841	0.625	19954	0.725	79404	0.462	2916	6.255
music	948	15244	0.283	0	0.000	1082	0.665	111	3.532
netscape	10385	83235	0.571	10862	0.769	36975	0.694	557	5.056
pattern	7142	28133	0.319	0	0.000	7137	0.010	7	4.571
sea_dusk	211	158628	0.020	4	1.000	1067	0.330	60	7.067
stone	6827	36070	0.793	0	0.000	7687	0.347	41	3.317
sunset	36925	390890	0.637	39594	0.664	122527	0.698	7158	7.540
winaw	9984	321561	0.255	0	0.000	11956	0.628	504	3.000
yahoo	3002	33519	0.444	3581	0.809	9425	0.609	815	7.539

Table 7
PWC Coding Statistics

6. Performance

Using a research grade implementation, the PWC encoded images of the previously described test set are decoded in 8 seconds on a 200 MHz Intel Pentium processor. This is approximately 225 Kbit/sec decode performance, sufficient to keep up with a 128 Kbit/sec ISDN connection. Encoding is about 25% slower due to increased memory use.

7. Conclusion

A new piecewise–constant image model and a language for describing such models has been proposed. The model is appropriate for application to palette image coding. An example piecewise–constant image coder demonstrated remarkable compression on a corpus of 15 palette images. The demonstrated compression is 20% better than a composite coder formed by selecting the best of JBIG bit plane coding, CALIC

predictive coding with palette ordering, and PNG LZ77 deflation for each image of the corpus. The appropriateness of the corpus was validated by the similar aggregate compression achieved by the competing methods even though their performance varied widely on individual images of the corpus.

The data show that the PWC coder is robust across a wide variety of images. The compression achieved is as good (2% worse on stone) or better than the best of the alternate methods on every image in the test set. Its aggregate compression is 31% better than PNG, and 24% better than the combination of CALIC with palette ordering. The other alternate methods are even less competitive.

8. Acknowledgments

The arithmetic coder used in the example coder is the carry free coder designed and written by Don Speck[4]. The Golomb tree coder[8] used for coding prediction errors was also done by Don. Nasir Memon graciously provided the palette ordering code used in the comparison experiments.

9. References

1 Xiaolin Wu, "An Algorithmic Study on Lossless Image Compression", Data Compression Conference Proceedings, March 31–April 3, 1996, IEEE Computer Society Press, Los Alamitos, California.

2 Stephen R. Tate, Lossless Compression of Region Edge Maps, CS–1992–9, Department of Computer Science, Duke University, Durham, NC, 1992.

3 Glen G. Langdon, Jr. and Jorma Rissanen, "Compression of Black–White Images with Arithmetic Coding", IEEE Transactions on Communications, Vol. COM–29(6), pp. 858–867 (June 1981).

4 Don Speck, "Local Activity Level Classification Model for Continuous–tone Coding", document N198 submitted to ISO/IEC JTC1/SC29/WG1 June 29, 1995.

5 Markus Kuhn, Version 0.9 of the JBIG–KIT, available via anonymous ftp at ftp.informatik.uni–erlangen.de/pub/doc/ISO/JBIG/jbigkit-0.8.tar.gz.

6 N. D. Memon and A. Venkateswaran, "On Ordering Color Maps for Lossless Predictive Coding", IEEE Transactions on Image Processing, 1996, Vol. 5, No. 11, pp. 1522–1527.

7 Jasc, Inc., http://www.jasc.com.

8 Don Speck, "Clarifying Some Details of ALCM", June 1996, document N351 submitted to ISO/IEC JTC1/SC29/WG1.