Compression of Palettized Images with Progressive Coding of the Color Information

Uwe Rauschenbach

Computer Science Department, University of Rostock,
D-18051 Rostock, GERMANY
Email: urausche@informatik.uni-rostock.de

ABSTRACT

This paper introduces a new compression method for palettized images, which supports progressive refinement of the color information in contrast to the resolution refinement used in standard methods like interlaced GIF. Such, fine image details can be recognized after decoding only a small part of the compressed image data. Achieved compression ratios are comparable to those of interlaced GIF or PNG. The method combines color map sorting with bitplane by bitplane prediction and Golomb coding of the pixel field.

Keywords: Color Palette, Progressive Coding, Image Data Compression

1. INTRODUCTION

The graphical nature of the World Wide Web requires the transmission of raster images over slow network connections. In the context of the ever-growing mobile access to the Internet via low-bandwidth mobile phones, users have to cope with long image transmission times. Progressive refinement while loading an image can support faster browsing, since the user may recognize the picture before all image information has been transmitted.

Quite often, computer generated imagery with few colors is used for graphical Web site navigation or for diagrams. These images can efficiently be stored as *palettized* or *colormapped* images. Palettized images are usually transmitted in the interlaced GIF or interlaced PNG[1] format, which refine the *image resolution* progressively by encoding an image in multiple passes, skipping a number of pixels in each pass. This scheme introduces a blocky structure during early transmission steps, and fine details are not recognizable until late transmission stages. When browsing the WWW over a slow link, this may require the user to wait for a long time until site navigation images (often containing small font text) become usable. Since the human visual system is very sensitive to shape and contrast, an alternative scheme of progressive *color information* refinement could solve this problem. Used instead of or combined with resolution refinement, it would offer the opportunity to recognize fine details with high contrast very early during transmission. However, for palettized images this topic has not yet been addressed in the literature.

In this paper, a new image compression method for palettized images with a small number of colors (up to 256) is introduced, which combines progressive coding of the color information with compression ratios comparable or superior to those achievable using interlaced GIF and PNG. The compression method builds upon a combination of different known techniques.

This work's main contribution is to introduce the concept of progressive color refinement into the domain of colormapped images and to show that it can be realized without sacrificing compression efficiency.

The rest of the paper is structured as follows: First, a review of related work is given. Second, the new progressive coding method is explained. Third, its performance is evaluated and compared to the existing methods PNG and GIF. Last but not least, possible optimizations and extensions are discussed.

2. RELATED WORK

Pictures with thousands or millions of colors are represented as true color images, storing the exact color for each pixel. In contrast to that, pictures containing only a few (typically up to 256) colors can be stored more efficiently as colormapped images. All used colors are kept in a color table (the palette), and the color of each pixel is represented by an index into that table. That's why the pixel values usually lack correlation with any metric (e.g., lightness) applicable to define an order on a set of colors.

Methods for the progressive refinement of the color information exist by now only for greyscale and true color images (see section 2.1), where this correlation is naturally present. Converting a palettized image into true color in order to be able to use such a scheme is in general not a good solution, since this leads often to a larger image file.

The missing correlation presents problems not only in the context of progressive color refinement but also in applying lossy compression methods to the index array. Sorting the color map can establish the desired correlation, which has been already researched in the context of the lossy compression of colormapped images (see section 2.2).

2.1. Progressive Refinement of the Color Information

Bell and Maeder[2] propose a progressive region based image compression method which they are extending by a scheme for color refinement. This scheme encodes the color information in several steps, but it needs already 128 standard colors for the first step. The scheme has been developed for true color images. For refining colormapped images, its granularity is too coarse.

The PROTRAC progressive image transmission system[3] decomposes true color images channel by channel into bitplanes. This leads to eight refinement levels, each containing one bitplane of the R, the G and the B channel. Per bitplane, eight consecutive bits are combined into one symbol. The symbol stream is LZW encoded.

None of these two systems is suitable for colormapped images. The lossless, progressive transmission of true color images is considered not very important in the targeted domain of online applications, since there exist methods like progressive JPEG which deliver higher compression ratios with comparable visual quality for screen display.

2.2. Lossy Compression of Colormapped Images

Colormapped images can be compressed using two basic methods. One opportunity is to interpret the pixel values as characters and to apply a lossless compression method like LZW (as in the GIF format) to them. For images with only a few colors, this yields high compression ratios. The other opportunity is to convert the picture into a true color image by replacing the index value at each pixel by the corresponding color map entry and then to apply a method suitable for this image class (e.g., JPEG[4]) to it. This creates three color channels (YC_bC_r), however, and leads to decreased compression efficiency since they must be compressed separately. Since the DCT coefficients are quantized, the decoded image may contain colors not present in the original.

Different authors[5–9] have investigated how colormapped images can be compressed using lossy techniques without conversion into true color. In order to establish the missing correlation between the index values, a *color map sorting step* is used prior to compression. This step sorts the color map such that similar colors are assigned similar indices, and that the redundancy between neighboring pixels is high.

The works differ in the sorting method used. Zaccarin and Liu[5] use the order of the colors along the luminance axis of the YC_bC_r color space as the sorting criterion. Waldemar and Ramstad[7] apply a color correlation measure which determines the similarity of colors by how often they occur in neighboring pixels. Po, Tan and Cham[8,9] use the so called *Closest Pair Method* in *RGB* space. Here, the color map is initialized with the color closest to black ($R = G = B = 0$) in terms of Euclidean distance. After that, the remaining colors are inserted step by step, always adding the color which is closest to the one most recently inserted. Chen, Peterson and Bender[6] divide the colors of the palette into groups of similar color, where the similarity of two colors is computed as the distance in the perceptually uniform CIE L*a*b* space.

After sorting, the pixel array (often named a pseudo gray level image) is now – like a "normal" gray scale image – encoded using JPEG,[5,6] a subband encoder[7] or a combination of DPCM, run length and Huffman coding.[8,9] Some authors use additional side information to handle the problems introduced by quantization: indices not present in the original image or ambiguous indices addressing different colors addressed by the same index value.

An alternative method is proposed by Chiang and Po.[10] They encode a palettized image using a lossy LZW algorithm producing an output stream which can be decoded by a standard GIF decompressor. An adaptive similarity

threshold is used to classify pixels with minimally varying colors as having the same color. This method does not modify the color map, however.

None of the described methods supports the progressive refinement of the color information.

3. DESCRIPTION OF THE NEW CODING METHOD

A method for progressively coding the color information of palettized images has to meet the following requirements:

1. Fine details with high contrast must be already recognizable after decoding a small portion of the compressed stream.
2. By decoding a bigger portion of the stream, the color information must refine progressively up to lossless recovery.
3. The scheme has to offer a compression performance comparable to the methods interlaced GIF and PNG, which are used for colormapped images today.

The new method MCQ (**M**oVi* **C**olor **Q**uantized format) has been designed to meet these requirements. After providing an overview over the algorithm, the steps of the method will be explained in detail.

Prior to encoding, the missing correlations between pixel values and color properties have to be established by *sorting* the color map using some of the schemes described in section 2.2. This step results in a pixel array representing similar colors by similar index values and dissimilar colors by very different index values, allowing progressive color refinement by means of progressively refining the index values. After sorting, the indices are encoded *bitplane by bitplane* in order to realize the desired refinement. Before each bitplane can be encoded, a palette of *composite colors* for all valid index values must be computed and written to the data stream.

Inter-pixel redundancy can be exploited to compress the pixel array. To efficiently represent homogeneous or nearly-homogeneous areas, *quadtrees* or *prediction* may be used. The first method compactly encodes homogeneous square areas in the image by using a tree structure, and is even able to exploit inter-bitplane redundancy when the BCQ method (Bitwise Condensed Quadtree[11]) is used. Quadtree generation is followed by an arithmetic coding step for increased compression. The second opportunity is to use already known pixel information to predict the still unknown value at the next pixel position. Only the difference between the predicted and the true value is stored, which results in many small or zero values. These can be compressed by a following run length encoding step. We compared *BCQ followed by arithmetic coding* with *prediction followed by Golomb (or RLR†) encoding.* BCQ delivered compressed files which were about 60% larger, why we have chosen to use prediction plus Golomb coding.

An *interlacing scheme* can be used to further reduce the size of the stream portion which must be decoded for a first approximation of the whole image. It is known that interlacing trades this advantage for a decrease in the overall compression ratio. Experiments with a test image set (see table 2) have shown that PNG without interlacing generates files of 67.7% the size of their interlaced PNG pendants. Thus, it must be carefully decided how many interlacing steps should be used. Because of requirement 1, MCQ uses optionally a simple line interlacing which encodes in a first pass only pixels in even rows; a second pass traverses the remaining odd rows.

Summarized, the encoder performs the following steps:

1. Sort the color map.
2. Apply the prediction method.
3. For each bitplane:
   (a) Blend the colors of the global color map into a composite color map, which contains one entry for each color index possible after decoding the current bitplane.
   (b) Encode the composite color map for the current bitplane.
   (c) RLR-encode the current bit of the prediction error of all pixel values. This step is split into two passes when using the optional interlacing scheme.

*MoVi (Mobile Visualization) is the name of the project in which MCQ has been developed.

†RLR: Run length / Rice encoding. Such an encoder has been described, e.g., by Malvar[12] in the context of wavelet coding.

3.1. Sorting the Color Map

In section 2.2, some sorting methods from the literature have been introduced. Unfortunately, there are no investigations which compare these methods directly. Thus, *Y sorting*[5] and *closest pair sorting*[8,9] have been implemented and compared visually. Using Y sorting, the colors of the color map are converted into the YC_bC_r color space, and the Y component is used as sorting criterion. Since Y represents the luminance of a color, this method is suitable to meet the first design requirement. The closest pair sorting method (called briefly *CP sorting* from now on) works in RGB space and stores similar colors in neighboring color map entries.

Tests using 34 colormapped images (cartoons, screen dumps, WWW navigation bitmaps) showed that no general statement can be given which method is suited best, but that the selection has to be made depending on the image. For most images can be stated, however, that the recognizability of the first few bitplanes is better using Y sorting (cf. figure 3 top), and that the CP scheme results in less color distortion in the lower bitplanes (figure 3 bottom).

Because of these results, a *hybrid sorting method* (called *YCP sorting* from now) has been introduced. This method can be defined for an image with n colors by a parameter y as follows: first, the color map is sorted using the Y method for the highest y bitplanes. For the lowest $n - y$ planes, the color map is split into sub-tables, which are each sorted using the CP method. Thus, the YCP method combines the advantages of Y and CP sorting and allows sorting control using the parameter y. As special cases, Y sorting ($y = n$) as well as CP sorting ($y = 0$) are possible. The optimum value for y is image dependent; as a pragmatic solution, $y = 1$ delivers satisfactory results in many cases.

3.2. Prediction

Prediction schemes exploit inter-pixel redundancy in order to increase compression. The value of the current pixel is predicted using the values of pixels visited earlier during the encoding process[‡], and only the error of this prediction must be encoded. Since the error is small (ideally zero), a subsequent coding step which efficiently encodes long runs of zeros yields high compression ratios. A widely used prediction method is DPCM (*Differential Pulse Code Modulation*), which represents a digital signal by the true value of the first sample, followed by a series of difference samples. The PNG graphics format combines this scheme with a selection of a set of *predicting pixels* depending on image characteristics in order to increase the compression ratio. For calculating the difference to the current *predicted pixel*, PNG computes a *predictor* selecting one of the following methods:

- the value 0 (no prediction)
- the value of the pixel *left* of the predicted pixel
- the value of the pixel *top* of the predicted pixel
- the average value of the pixels *left and top* of the predicted pixel
- the value of one of the pixels *left*, *top* or *diagonally left and top* of the predicted pixel. Pixel selection is based on a difference measure published by Paeth[13]

Method selection is performed for each pixel row separately, and the choice is encoded as side information in the header of the compressed image file.

For our method, we adapt the prediction scheme from PNG by using a different predictor than DPCM. The problem with DPCM and bitplane by bitplane transmission is, that difference values are approximated when only the higher bitplanes are available. This may lead to error accumulation, resulting in decoded color indices which have no corresponding color map entry. That's why we need a predictor which handles each bitplane separately. If the current bit is the same in the predicting and the predicted pixel, the predictor should deliver a 0-bit, otherwise a 1-bit. This property is offered by the *XOR* Boolean function, which can be applied to compute the prediction as well as to revert it.

In order to decide which of the five alternatives results in the best compression, we apply all five methods to the pixels of each row and simulate the encoding of the result, counting output bytes instead of outputting them. Then, the alternative resulting in the smallest number of simulated output bytes is selected for the current row. If interlacing is used, prediction using pixels *atop* the current pixel must take into account that during the first interlacing pass only even rows are available.

[‡]At the decoder side, these pixels must be known when decoding the current pixel.

3.3. Computing the Composite Color Map

Before the coded representation of a bitplane in the pixel array can be written to the stream, the specification of a valid colormap for this bitplane is necessary. These color maps have $m_i \leq 2^i$ entries, where $i \geq 1$ represents the number of the bitplane, starting with $i = 1$ for the most significant bit. Given $m_{i_{max}}$ colors in the image, the last (global) color map includes $m_{i_{max}}$ entries; and $i_{max} = \lceil \log_2 m_{i_{max}} \rceil$ bitplanes must be coded. The x-th entry ($0 \leq x < m_i$) in the i-th ($1 \leq i < i_{max}$) color map is generated by compositing the entries with indices $2 \cdot x$ and $2 \cdot x + 1$ of the $i + 1$-th colormap. Each entry is weighted using the number of pixels of the corresponding color.

If the index $2 \cdot x + 1$ points to an entry after the end of the $i + 1$-th color map, the entry $2 \cdot x$ is inserted unmodified from the $i + 1$-th into the i-th colormap. This can lead to a problem with images where $m_{i_{max}}$ is marginally larger than a power of two. For example, if $m_{i_{max}} = 2^{(i_{max}-1)} + 1$, the color with the largest index $x_{max} = 2^{(i_{max}-1)}$ does not find a "partner" during compositing and occurs unaltered throughout all composite color maps, while all other colors are blended. The highest bitplane colormap m_1 then contains the blend of all colors as one entry and the x_{max}-th color as the other. If this color occurs seldom in the image or is very similar to the blend of all other colors, the highest bitplane image shows poor contrast or unimportant details. A pragmatic solution to that problem is reverting the colormap order directly after sorting, such that now the darkest color is at position x_{max}. To decide whether to revert or not, we use a contrast measure. It is computed as the sum of differences between neighboring pixels over the first bit of all index values.

After decoding the first few bitplanes, *color distortions* are present in the image, although the shapes can already be recognized. For shape recognition, color information is not necessary. If the color distortions are unacceptable for the user, the colors in the according composite palettes are replaced by gray levels, such that first a gray level image is decoded which is later enriched by color when enough bitplanes are available. Currently, the bitplane c where to switch from gray scale to color is selected by the image author. An automatic selection based on a distortion measure is considered as future work.

3.4. Bitplane Encoding

The encoding of the pixel array is carried out bitplane by bitplane in order to support color refinement. To achieve a high compression ratio, we can exploit the fact that the highest bit with a value different from zero appears where predicted and predicting pixel differ for the first time. Because of the inter-pixel redundancy, this is often a low bitplane, which results in many 0-bits and few 1-bits. For the following bitplanes, 0-bits and 1-bits are nearly equally distributed. A separation of the bits into *significance* and *refinement* information like in EZW[14] helps to exploit these different characteristics. Significance data inform the decoder about the highest 1-bit of a pixel with a value still known to be zero, and refinement represents a further bit for a pixel value already known to be different from zero.

The significance information is run length encoded in order to represent the long runs of zeros efficiently. This is done by using the RLR encoder exploited by Malvar[12] to encode the significance of wavelet coefficients. The output of this encoder is known as *Golomb code*. Encoding is adaptive with a parameter k ($k > 0$). As code word, a 0-bit is output if a sequence of 2^k insignificant pixels has been found. A 1-bit followed by the k bit binary representation of the number l is output if a sequence of l ($l < 2^k$) insignificant pixels is terminated by a significant pixel. When coding significance, all pixels which have already been encoded significant in previous bitplanes are skipped, and their current bit is buffered as refinement information. After outputting a code word, the bits from the refinement buffer are appended to the data stream without encoding them. If interlacing is used, each bitplane is encoded in two passes, first considering the rows with even Y coordinate, then the ones with uneven Y.

In order to adapt the parameter k to the characteristics of the data source, Malvar[12] proposes to increment k after outputting a 0 and to decrement it after outputting a code word starting with 1. We have carried out experiments using a set of 34 test images in order to check which integer increment P_{inc} ($1 \leq P_{inc} < 7$) and decrement P_{dec} ($0 \leq P_{dec} < 9$) yields the best compression. A value of $P_{dec} = 0$ represents a variable decrement: k is set to the value which would have been needed to output the current l without wasting bits. To be able to plot the results in 2D, both parameters have been "folded" into a combined parameter $P_{comb} := 10 \cdot P_{inc} + P_{dec}$. The curves of the compressed file size as a function of P_{comb} are similar for all test images. Figure 7 shows an example. File size peaks are at $P_{dec} = 1$; between peaks, the curve is relatively flat. The global minimum is located at values $P_{comb} \leq 29$ for all images. Since the minimum was in many cases located at $P_{comb} = 14$ and the curves are rather flat, $P_{inc} = 1$ and $P_{dec} = 4$ are used as default values in order to save the computing time for finding the optimal solution for each

image individually. Experiments with 516 clipart images have shown that this choice only led to an increase of the sum of all file sizes by 0.7% with interlacing and 0.9% without interlacing compared to the optimal choice.

4. PERFORMANCE EVALUATION

In this section, the results achievable using the proposed coding technique will be presented, and its performance will be compared with the standard methods interlaced GIF and both interlaced and non-interlaced PNG.

For comparison, the visual quality in the course of a simulated transmission of the images over a slow network and the achievable overall compression ratio are considered as criteria. Furthermore, the dependency of the compression performance on the number of colors in the image and the vulnerability against dithering noise will be evaluated.

4.1. Visual Quality

It can be stated, that the design goal of recognizability of fine details after decoding a small portion of the image has been met. Using MCQ, fine details are visible after decoding a much smaller portion of the image than using interlaced PNG or GIF.

The figures 4 and 5 show an example. Four image formats are compared: MCQ without interlacing (top left), MCQ with interlacing (bottom left), interlaced PNG (top right) and interlaced GIF (bottom right). Figure 4 shows the image quality after decoding the first 2048 bytes, which corresponds to 2.3 seconds transmission time when accessing the image over the Internet via a GSM mobile phone. Using interlaced MCQ, information about the first bitplane is available for all pixels in a somewhat lower resolution, such that the shape of the logos can be recognized. In contrast, in the PNG and GIF images it is even impossible to recognize the logos. In figure 5, the first bitplane has been transmitted completely using non-interlaced MCQ. Here, all texts are readable after 3.8 seconds of simulated transmission time (3456 bytes). The PNG and GIF versions still do not even allow the recognition of all logos, and no text is readable. Compared to the 3.8 seconds using MCQ, it takes 13.6 seconds of simulated transmission time until all text in the corresponding GIF image can be read.

Although interlacing allows to display information about all pixels in the image after decoding a smaller part of the file than without interlacing, compression efficiency suffers. In figure 5, the first bitplane has been completely decoded using non-interlaced MCQ, while interlaced MCQ still has to decode some data for that bitplane in the bottom image parts where lower resolution text can be seen.

On web pages, images are often generated from higher resolution versions by downscaling and smoothing, which results in a larger number of colors in the smoothed image. Figure 6 shows such an image. It can be seen that GIF compresses this image better than MCQ (the MCQ file size is 119.4% the size of the GIF), but MCQ delivers a similar visual quality if approximately the number of bytes corresponding to the GIF file size is decoded.

4.2. Compression Ratio

The proposed image format allows the earlier recognition of fine details compared to the established formats GIF and PNG. Although this is a useful feature, a new image format has to offer compression ratios comparable to those of established methods. In this section, the compression ratio achievable using MCQ is compared with that of PNG and GIF.

Table 1. Bitplane number histogram of the test image set.

Number of bitplanes	2	3	4	5	6	7	8	Total
Number of images	64	213	174	36	17	5	7	516

In order to perform the compression tests, a set of 516 colormapped images has been composed of 482 clipart images[§] and 34 maps, screen dumps and navigation bitmaps from the Internet. Table 1 shows the composition of the test set with respect to the number of bit planes. Most images contain three or four bitplanes, i.e. 5 to 8 or 9 to 16 colors.

[§] All images have been taken from the subdirectories "cartoons" and "the_east" of the CorelDraw8 CDROM.

All files have been compressed using the methods iMCQ¶, MCQ, iGIF, iPNG and PNG, and the total size of the compressed files has been computed for each method. Table 2 lists the results of the tests. Compared to interlaced PNG, the new method compresses better; delivering files of 79.9% resp. 68.3% the size of their iPNG pendants. MCQ with interlacing delivers compression ratios comparable to that of interlaced GIF; without interlacing, MCQ compresses approximately 15% better.

Comparing MCQ with non-interlaced PNG is not especially fair since the latter method does not provide progressive refinement. Nevertheless, it can be seen that MCQ (without interlacing) delivers comparable compression ratios while additionally supporting refinement.

In general, it can be said that using interlacing in MCQ sacrifices approximately 15% of compression.

Table 2. Comparison of the total compressed file size using different compression methods.

Method	iMCQ	MCQ	iGIF	iPNG	PNG
Total file size [bytes]	2198307	1877666	2230259	2750039	1859936
Percentage of iGIF	98.6%	84.2%	100.0%	123.3%	83.4%
Percentage of iPNG	79.9%	68.3%	81.1%	100.0%	67.7%

4.3. Influence of the Number of Colors and of Dithering Noise on the Compression Ratio

When evaluating the visual quality, it has been found that MCQ delivers high compression ratios on images with a small number of colors; the compression ratio seems to decrease faster than the compression ratio achievable using, e.g., iGIF, for increasing numbers of colors. To quantify that effect, two experiments have been carried out.

In a first test, the set of 516 test images has been split into subgroups according to the number of bitplanes, and for each group, the total compressed file size using iMCQ and MCQ has been computed and related to iGIF as 100%. The results are plotted in figure 1. This test suggests, that compression drops faster with an increasing number of colors using MCQ than using iGIF. For images with a small number of colors, MCQ is more efficient than iGIF. For many colors, the performance of iGIF and MCQ is comparable.

For a second test, a 244-color image (shown in figure 6) has been created by downscaling and smoothing an 18-color image. This image can be better compressed using iGIF and iPNG than using MCQ. It has been successively quantized to decreasing numbers of colors, and the quantized versions have been encoded using the four methods. Figure 2 shows that for many colors, iPNG and iGIF outperform MCQ, but that for small numbers of colors the MCQ variants compress better. An interesting fact is that larger "jumps" in the compression ratio occur near but not exactly at bitplane changes (at 130, 65, 32 and 16 colors).

Table 3. The influence of dithering on the compression performance.

	Method	iMCQ	MCQ	iGIF	iPNG	PNG
Dithered images	Total file size [bytes]	818137	755946	653888	811193	610420
	Percentage of iGIF	125.1%	115.6%	100.0%	124.1%	93.4%
	Percentage of iPNG	100.9%	93.2%	80.6%	100.0%	75.2%
Un-dithered images	Total file size [bytes]	877510	745447	866549	1221850	812085
	Percentage of iGIF	101.26%	86.02%	100.00%	141.00%	93.71%
	Percentage of iPNG	71.82%	61.01%	70.92%	100.00%	66.46%

Another negative influence on the compression ratio is the presence of dithering noise in the image. In order to quantify that influence, the contents of the "cartoons" subdirectory of the CorelDraw clipart CD has been split into a set of 117 dithered and 417 un-dithered images. Table 3 lists the achieved compression ratios. In the presence of dithering patterns, MCQ compresses both with and without interlacing worse than iGIF; but comparable with iPNG (iMCQ) or slightly better (MCQ). Un-dithered images are compressed better than iGIF (MCQ) or comparable to it (iMCQ). The un-dithered set is compressed by both MCQ modes better than by iPNG.

¶An abbreviation starting with "i" stands for the interlaced mode of the according technique.

Since the negative influence of dithering patterns is stronger than that of many colors, dithering should be avoided by using more colors.

5. IMPROVEMENTS

It has been found that interlacing offers an advantage only for the first bitplane; in lower bitplanes, it decreases compression. Thus, an improved version of MCQ should only encode the first (highest) bitplane in two passes, all other bitplanes should be encoded without interlacing.

The vulnerability to dithering noise points to another opportunity for improvement. Dithering patterns decrease the length of the sequences of 0-bits, but methods which encode symbol patterns efficiently (like LZ77[15]) may be used to increase the compression ratio.

Preliminary tests with four images have been promising. In an experiment, three alternatives have been used to output significance as well as refinement data: raw output without compression, RLR encoding and `gzip` (LZ77) compression. For an image without dithering, RLR coding of the significance information and raw output of refinement bits was confirmed to be the optimum choice for most bit planes. Images with dithering patterns tended to use LZ77 for the refinement of all bitplanes and for the significance information of the first bitplanes; the remaining significance bits were RLR encoded. Using this scheme, the compressed file size could be decreased by 10 to 23% for the dithered images; for un-dithered images the decrease was marginal.

6. CONCLUSIONS AND FUTURE WORK

This paper has presented MCQ, a new progressive coding method for palettized images. While existing methods for colormapped images use interlacing to refine the resolution of an image, MCQ refines the color information bitplane by bitplane. This allows the recognition of shapes with fine details (e.g., text) in the image after decoding a small portion of the compressed stream, making the method especially suitable for navigation bitmaps in low bandwidth online services.

To be able to apply color refinement to colormapped images, a sorting method is applied to the colormap. By using prediction and RLR encoding, MCQ achieves compression ratios comparable or superior to the existing methods GIF and PNG.

The proposed method offers potential for improvement in two respects. First, the compression ratio achievable using MCQ decreases faster with increasing number of colors than that of PNG and GIF. Second, MCQ is more vulnerable to dithered images than the two classic methods. In section 5, an idea has been presented to improve MCQ's performance under these conditions.

Future work includes to realize the sketched improvements and to implement the method in Java in order to make it available in a wide range of browsers. Furthermore, automatic mechanisms will be looked at which ease the creation of MCQ files. Currently, the author has to set the parameters y for switching between Y and CP sorting and c to switch from gray scale to color manually when saving the image. Chen, Peterson and Bender[6] exploited the CIE L*a*b* color space for sorting and reported results superior to that using the closest pair method in RGB space. That's why evaluating the use of this color space for sorting is planned, too.

ACKNOWLEDGMENTS

This work was carried out in the Mobile Visualization (MoVi) project supported by The German Science Foundation under contract no. Schu-887/3-3. I wish to thank Mathias Balzer and Christian Schmidt for implementation work and my advisor, Heidrun Schumann, for her steady encouragement and support. Special thanks to the Warnow Regional Transport Board and the Nitschke Advertizing Studio for kindly permitting to use their map image underlying figure 6.

REFERENCES

1. "PNG specification." http://www.wco.com/~png/, http://www.w3.org/TR/REC-png.
2. A. Maeder and D. Bell, "Modeling colour refinement for progressive image coding," in *Proc. Second IS&T/SID Color Imaging Conference*, pp. 111–114, November 1994.
3. V. Lalich-Petrich, G. Bhatia, and L. Davis, "Progressive image transmission capability (PROTRAC) on the World Wide Web," in *Proc. 3rd Int. WWW Conference*, (Darmstadt, Germany), 1995. Posters and Demos, http://www.umiacs.umd.edu/users/lpv/HTML/PROTRAC/protrac.html.
4. W. Pennebaker and J. Mitchell, *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, New York, 1993.
5. A. Zaccarin and B. Liu, "A novel approach for coding color quantized images," *IEEE Transactions on Image Processing* **2**(4), pp. 442–453, 1993.
6. Y. Chen, H. Peterson, and W. Bender, "Lossy compression of palettized images," in *Proc. IEEE Conference on Acoustics, Speech & Signal Processing (ICASSP)*, vol. 5, pp. 325–328, 1993.
7. P. Waldemar and T. Ramstad, "Subband coding of color images with limited palette size," in *Proc. IEEE Conference on Acoustics, Speech & Signal Processing (ICASSP)*, vol. 5, pp. 353–356, April 1994.
8. L. M. Po, W.-T. Tan, and C.-H. Chan, "Address predictive color quantization image compression for multimedia applications," in *Proc. IEEE Conference on Acoustics, Speech & Signal Processing (ICASSP)*, vol. 5, pp. 289–292, 1994.
9. L. M. Po and W.-T. Tan, "Block address predictive color quantisation image compression," *IEE Electornic Letters* **30**, pp. 120–121, 20 Jan 1994.
10. S. W. Chiang and L. M. Po, "Adaptive lossy LZW algorithm for palettized image compression," *IEE Electronics Letters* **33**, pp. 852–854, May 8 1997.
11. M. Dürst and T. Kunii, "Progressive transmission increasing both spatial and gray scale resolution," in *Proc. International Conference on Multimedia Information Systems*, pp. 175–186, McGraw-Hill, (Singapore), 1991.
12. H. S. Malvar, "Fast progressive wavelet coding," in *Proc. Data Compression Conference (DCC99)*, pp. 336–343, (Snowbird, Utah), March 29 – 31 1999. http://www.research.microsoft.com/ ~malvar/papers/index.htm.
13. A. Paeth, "Image file compression made easy," in *Graphics Gems II*, J. Arvo, ed., Academic Press, San Diego, 1991.
14. J. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions on Signal Processing* **41**, pp. 3445–3462, Dec. 1993.
15. J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory* **23**, pp. 337–343, May 1977.

APPENDIX A. IMAGES AND DIAGRAMS

The figures 3, 4, 5 and 6 should have been reproduced in color. Color versions of these figures can be downloaded from `http://wwwicg.informatik.uni-rostock.de/Projekte/MoVi/Publications/VCIP2000/images.html`.

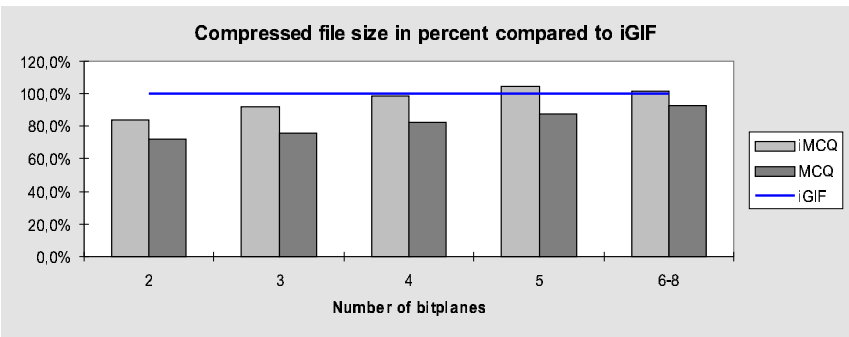

Figure 1. Compressed file size using iMCQ and MCQ compared to iGIF depending on the number of colors for the test image set (cf. table 1).

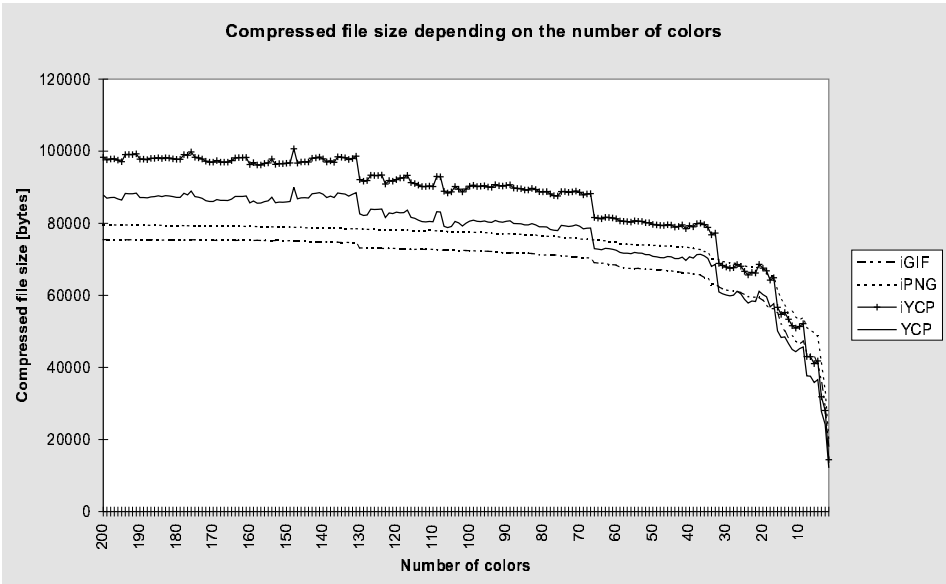

Figure 2. Compressed file size depending on the number of colors. Image: see figure 6.

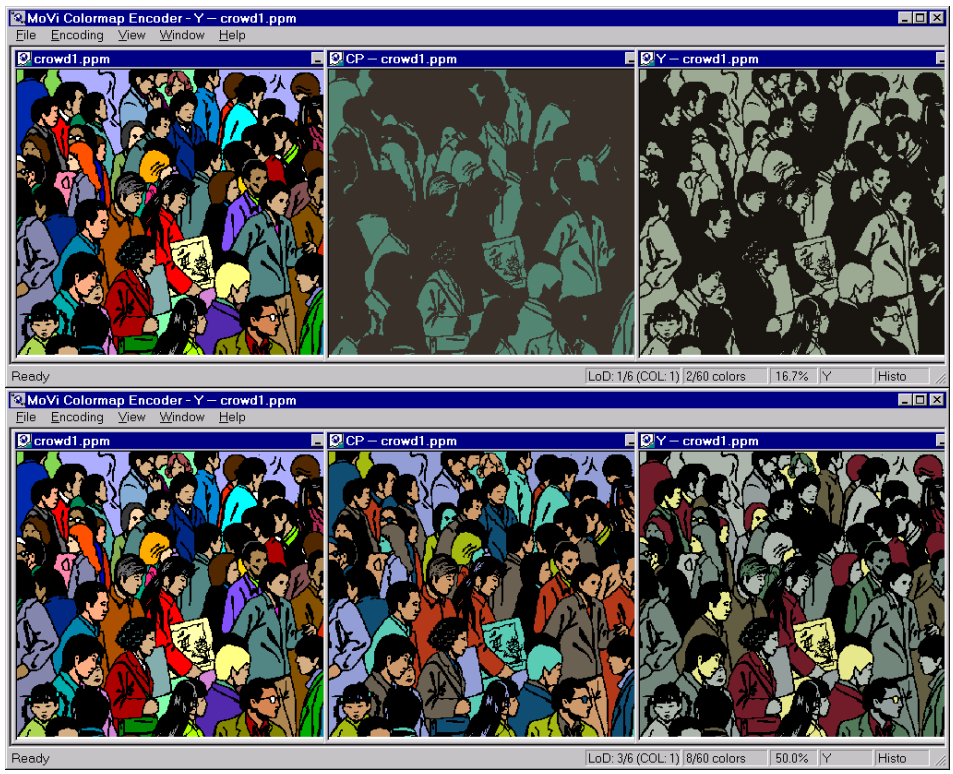

Figure 3. Comparison of the image quality after decoding of the first (top) and third (bottom) bitplane using Y sorting (center) and CP sorting (right). The original image is shown in the left pane. This image must be reproduced in color in order to be meaningful; see `http://wwwicg.informatik.uni-rostock.de/Projekte/MoVi/Publications/VCIP2000/images.html`.

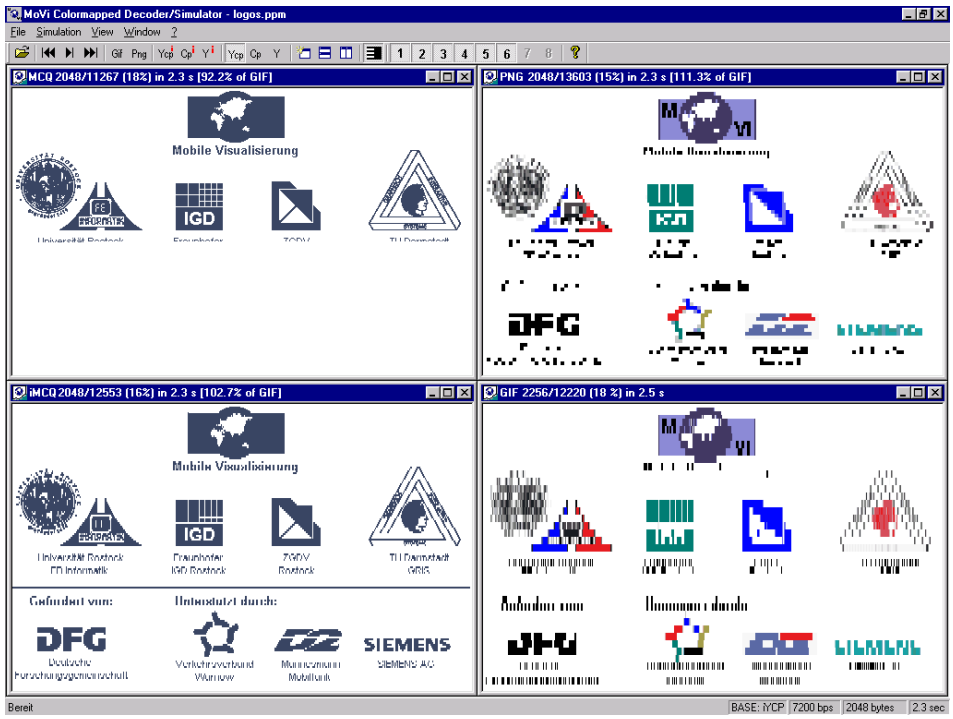

Figure 4. Comparison of interlaced MCQ and non-interlaced MCQ using YCP sorting with interlaced PNG and interlaced GIF. Image: sample navigation bitmap with 38 colors. Decoding of first pass through first bitplane finished using interlaced MCQ.

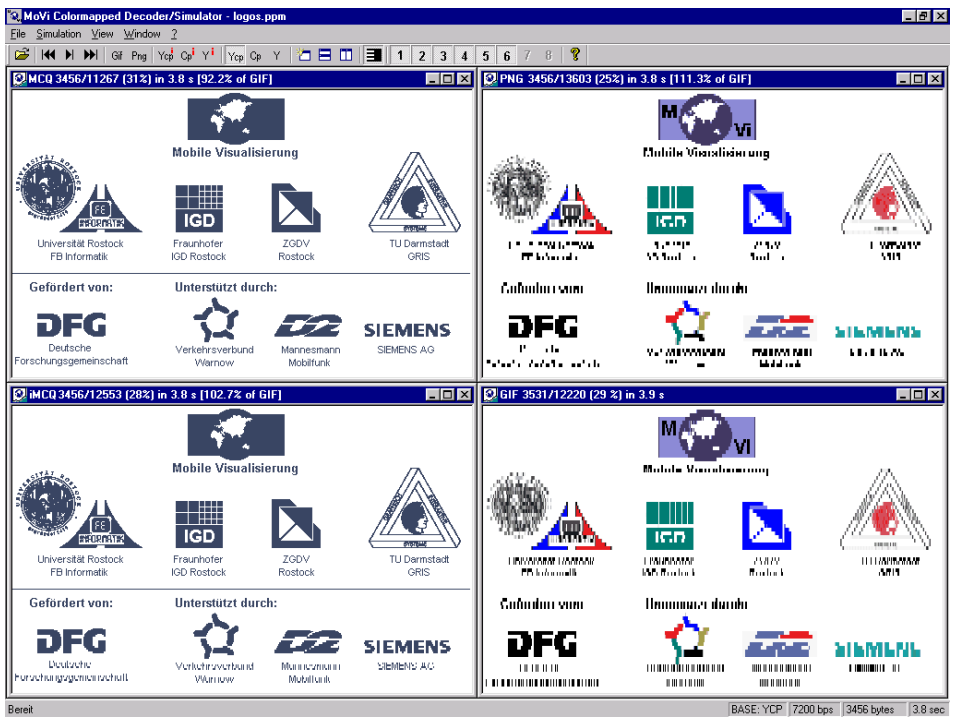

Figure 5. Comparison of iMCQ and MCQ with interlaced PNG and interlaced GIF. Configuration see figure 4. Decoding of first bitplane finished using non-interlaced MCQ.

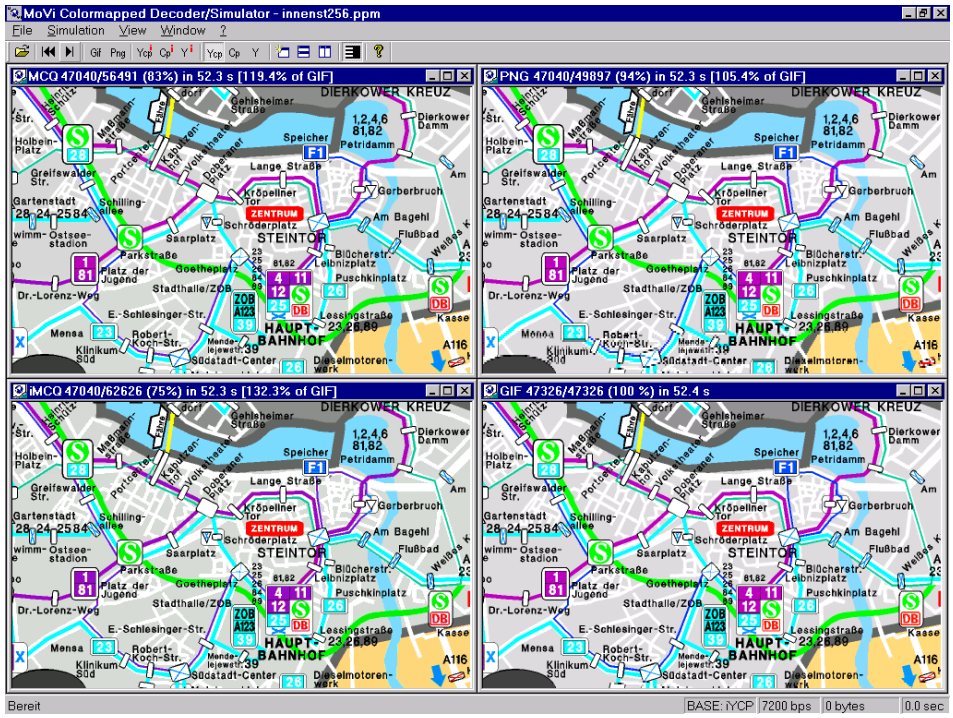

Figure 6. Comparison of iMCQ and MCQ with interlaced PNG and interlaced GIF. Image: City map with 244 colors generated by downscaling and smoothing from an 18 colors image. Decoding the same amount of data for MCQ and iGIF. This image must be reproduced in color in order to be meaningful; see `http://wwwicg.informatik.uni-rostock.de/Projekte/MoVi/Publications/VCIP2000/images.html`.

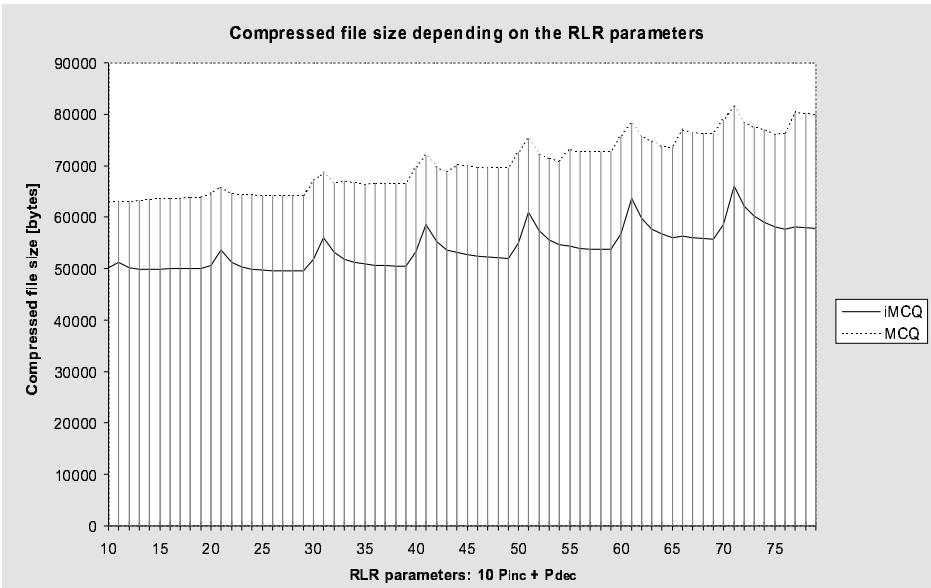

Figure 7. Compressed file size depending on the parameters P_{inc} and P_{dec} for one test image.