Presented at: WSCG '97 Winter School of Computer Graphics, Plzen, Czech Republic, February 10-14, 1997

Adaptive Image Transmission

Uwe Rauschenbach, Heidrun Schumann

Email: {urausche,schumann}@informatik.uni-rostock.de
WWW: http://www.informatik.uni-rostock.de/{~urausche,~schumann}

University of Rostock, Computer Science Department
D-18051 Rostock
Germany

Abstract. This paper presents an image transmission control mechanism which – in contrast to existing systems – adapts dynamically to the capabilities of the client display as well as to the available transmission bandwidth. This mechanism is based on existing image compression algorithms whose strengths and weaknesses are summarised. For JPEG image compression, the predictability of the compression ratio depending on image content and compressor parameterization are discussed. An experimental image transmission system is presented which shows the feasibility of adaptive image transmission.

Keywords: Adaptive image transmission, Image compression, Image quality, Image compressor configuration

1 Introduction

Globally distributed information spaces (e.g., the WWW) make extensive use of images, which, however, demand high bandwidth when transmitted over the network. This problem gets even harder when using mobile computers and wireless data transmission. To decrease the bandwidth demands of image transmission, lossless and lossy image compression methods are widely used.

Lossless image compressors preserve all the information in the image and only remove redundancies by using entropy encoding methods like LZW. Most image file formats (e.g., GIF or PNG [Cro95]) perform some kind of lossless image compression in order to save storage capacity. Additionally, dedicated lossless compression methods have been developed for special image classes, e.g., JBIG for monochrome images [CCI92].

Lossy algorithms try to remove information from the image which is „unimportant“ with respect to the human eye. Thus, they achieve much higher compression ratios than lossless methods. There are three basic approaches to lossy image compression. The ISO standard *JPEG* [PMi93] divides an image into blocks of 8 by 8 pixels, which are then converted into the frequency domain using, e.g., the discrete cosine transform (DCT). The loss of information is introduced by quantizing the DCT coefficients. After quantization, an entropy encoding step is performed. *Wavelet* based compression (c.f., e.g., [HJS94]) becomes more and more popular in image and video coding. This compression method is based on the wavelet transformation, usually implemented by quadrature mirror filters containing a high-pass and a low-pass filter. By successive filtering and subsampling, the image is partitioned into a hierarchy (pyramid) of smaller images. The loss of information is introduced by quantisation. *Fractal* image compression [Jaq90], [Fis95] is based on iterated function systems and exploits self-similarities in the image. Image blocks of different size are considered similar to other blocks or their translated, rotated, mirrored and/or scaled copies according to a similarity function. The error metric of this function determines the loss of information. However, searching for similar blocks in an image is a process of high time complexity.

In most distributed information systems, images are compressed off-line with a fixed compression ratio and then put into the information space. This *static approach* does not cater for the needs of clients connected to the system by links with different bandwidth. Furthermore, the varying display capabilities of different client computers are not considered accordingly. Thus, unnecessary data transfer often leads to long response times and transmission costs, especially in low bandwidth networks. To overcome these deficiencies, we developed an *adaptive approach* to image transmission.

2 A Concept for Adaptive Image Transmission

2.1 Overview

Adaptive image transmission can stretch the limits imposed by low bandwidth, restricted client display resources and processing power: it can save resources by using image compression methods and adapting the images to the client's display capabilities.

The basic idea is to do all necessary reduction at the *server* side. In a first step called *image reduction*, all those parts of the image can be removed which can't be displayed at the client. This includes reduction of the resolution as well as reduction of the number of colours – controlled by *context parameters* (e.g., screen resolution and number of displayable colours) sent by the client. Although straightforward, this method can already decrease the image size by a considerable amount. Take, for instance, a 1280x1024 pixel true colour image, which has a size of 3.75 Mbytes. Adapting it to a 640x480, 256 colour laptop display reduces the image data size to 300 Kbytes – without using a dedicated image compression method[1]!

In the following *compression* step, lossy or lossless data compression can be applied in order to decrease transmission bandwidth demands further. As different image compression algorithms are best suited for different types of images, the best compressor must be *selected* automatically depending on the image and on further *context parameters* such as bandwidth and maximum time the user is willing to wait for the image.

The loss of information introduced by lossy image compressors can be controlled by a parameter set. In order to provide the user with response times conform with his expectations, it would be desirable to know which compression ratio corresponds to a certain parameter value. However, the problem is that there is no direct match between a particular parameter value and the compression ratio achieved using this parameter. A method for *compressor configuration* which approximates this match is required, but this problem has shown to be non-trivial.

Adaptive image transmission means to carry out the adaptation to the client's display, the selection and the configuration of the compression algorithm automatically.

2.2 Evaluation of Image Compression Algorithms

In order to select compression methods to use in adaptive image transmission, we looked at some lossless image compressors (PNG, GIF and JBIG) and conducted experiments to evaluate three lossy compressors based on JPEG, fractal compression and wavelet compression. Table 1 shows the evaluation criteria and summarises the results. In figure 1, the compression ratio as a function of a control parameter is plotted for six different test images. As it can be seen, JPEG is the *least image dependent* one of the three lossy compressors evaluated[2].

[1] Of course, this step can also be seen as a lossy compression method, as it leads to a loss of image information.
[2] These experiments and their results are described in detail in [RSS96].

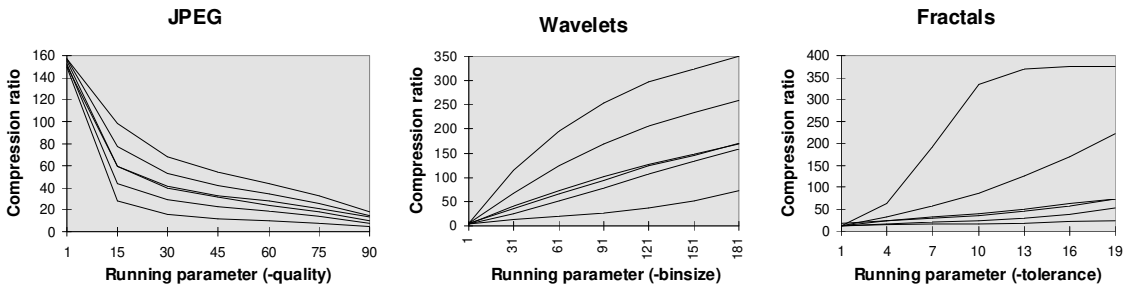

Figure 1: The compression ratio of lossy compressors as a function of a control parameter for six test images

Based on the results of our evaluation, we chose JBIG, GIF, PNG and JPEG for the use in adaptive image transmission. JBIG has been chosen because it offers better compression on monochrome images than GIF and PNG, GIF has been selected as the least common denominator of Web browser image formats, PNG as a better-compressing and patent-free alternative to GIF, and JPEG because it is the fastest, least image dependent and most widely supported lossy compressor.

	Criteria							
Methods	Lossy	Preferred image type	Availability in current web browsers	Support for progressive encoding	Speed	Image content dependence	Configurable[3]	Symmetric[4]
JBIG	No	1bit	Rare	Yes	Fast	n/a	No	Yes
GIF	No	8bit	All	Yes	Fast	n/a	No	Yes
PNG	No	8bit-24bit[5]	Some	Yes	Fast[5]	n/a	No	Yes
JPEG	Yes	24bit	Most[6]	Yes	Fast	medium[7]	Yes	Yes
Wavelet	Yes	24bit	Rare / None	Yes	Slow	high[7]	Yes	No
Fractal	Yes	24bit	Rare	No	Slow[8]	high[7]	Yes	No

Table 1: Characteristics of image compression methods

2.3 Compressor Selection Depending on Image Content and Bandwidth

After the step of image reduction, a compressor has to be selected to compress the reduced image. Two factors influence this decision: the properties of the reduced image and the context parameters *maximum waiting time T* and *available bandwidth B*. Having these parameters, the *target file size F* in bytes is computed as $F=T \cdot B/8$.

Now, we look for the compressor which is able to compress the image down to the target file size at the maximum possible quality (i.e., lossless compressors are preferred). Depending on the number of colours in the image, a set of candidate compressors is selected for the image (see column „Preferred image type“ of table 1).

[3] A compressor is called *configurable* when it offers a set of configuration parameters to control the compression ratio.
[4] A compressor is called *symmetric* when the computing time for compression and decompression is nearly equal.
[5] 24bit colour images are supported, but PNG is only fast enough for interactive applications on 8bit images.
[6] Most browsers support baseline JPEG. Progressive JPEG support is being added to more and more browsers at the moment. There are no browsers which support hierarchical JPEG.
[7] See figure 1.
[8] Additionally to being slow, the compression time of a fractal encoder depends to a high degree on the compression ratio (the higher the compression ratio the faster the algorithm, see [RSS96]).

One could think that the lossy compression algorithm JPEG is always superior over lossless algorithms – but that is not the case. As JPEG has been developed for continuous-tone true colour images, it delivers best compression on this image class. On images with 256 or less colours, lossless compressors like PNG are often superior over JPEG. We conducted a test which illustrates this fact: Six images were quantised to three up to nine bit planes. The resulting images were compressed using PNG on the one hand and JPEG with different quality parameter values on the other hand. The curves in figure 2 represent the average compression ratio for the six images. Furthermore, two curves have been added depicting the compression ratio of the images best and worst compressible using PNG. It can be easily seen that the compression ratio of JPEG is independent from the number of colours, and that for images with only a few colours PNG is superior over JPEG. However, the break-even-point between JPEG and PNG can not be derived from the graph as the variation between individual images is quite high.

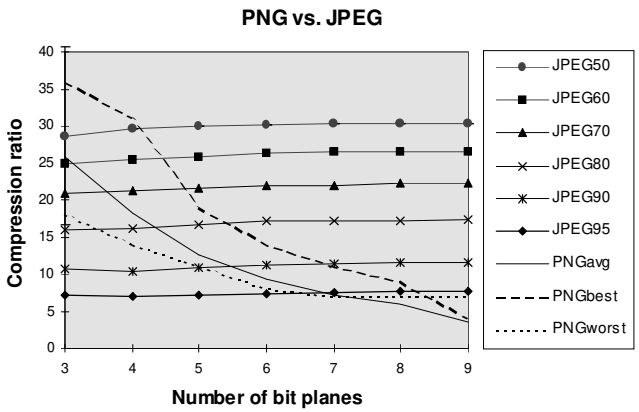

Figure 2: PNG vs. JPEG

These results suggest, that a compression algorithm has to be selected depending on the content of the image. However, the magnitude of the variation makes it impossible to do this just by lookup in pre-recorded tables. Instead, image analysis is required to compute a predictor which predicts the achievable compression ratio. We tried several simple predictors (like the variance of the Y channel of the image), but they did not deliver the required accuracy. At the moment, more complex predictors are being evaluated.

Meanwhile, we adopt a simple approach: each requested image is compressed using all lossless methods applicable to it (that are three at most), and the compressed images are stored in a cache. As a result of the compression, we know the compression ratio of that image achievable using lossless algorithms. As all three algorithms are fast[9] and we can reuse the compressed images, this method seems feasible.

2.4 Configuration of the JPEG Image Compressor[10]

2.4.1 Basic Idea

For lossless image compressor selection, the method described above works. When extending this selection process to cater for lossy image compressors too, there is some more work

[9] Compression time is as follows for a 512 by 512 pixel image on an SGI Indy workstation: 8bit image – 1.7s PNG, 1.1s GIF; 1bit image: 0.4s PNG, 0.8s GIF, 0.4s JBIG.

[10] We use the JPEG implementation of the Independent JPEG Group, see [IJG96].

involved. The problem here is that the compressor can be parameterized to control the loss of information, which affects the achieved compression ratio. In the case of JPEG, this is done by using a `quality`[11] parameter. In order to satisfy the bandwidth constraint given by the context parameters, we must determine which parameter value to use in order to achieve the required compression ratio. As the compression ratio varies between different images for the same parameter value (c.f. figure 1), a prediction method is needed which compensates this variation.

For approximating the function $R_u(q)$ (the compression ratio R of an unknown image u as a function of the parameter q), we propose the following method: Taking a set of n test images with different content, each image i is compressed iterating the parameter q from 1 to 100 and the achieved compression ratio $R_i(q)$ is recorded. For each parameter value q, the average $A(q)$ and the standard deviation $D(q)$ of the compression ratio is then computed from the $R_i(q)$. Two tables are the results of this preparation step which can now be used to approximate $R_u(q)$.

Having the unknown image u, we first do a sample compression[12] of u using a known parameter value q_o and record the achieved compression ratio $R_u(q_o)$. The difference $\Delta_o = R_u(q_o) - A(q_o)$ is computed and normalised according to the value of the standard deviation at q_o: $\Delta = \Delta_o / D(q_o)$. $R_u(q)$ can now be approximated by adding the standard deviation scaled by Δ to the average values: $\forall q{:}1 \le q \le 100{:}\, R_u(q) = A(q) + \Delta \cdot D(q)$, resulting in a new table. From this table, the required parameter value q_x can be looked up. Figure 3 illustrates the idea.

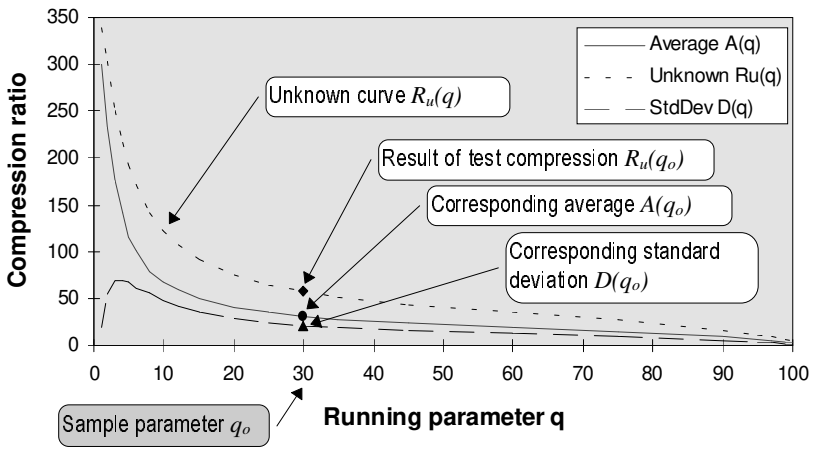

Figure 3: Parameter value prediction

2.4.2 Methods for Increasing Accuracy

There are several opportunities to select the sample parameter q_o (see figure 4). The initial *one-step* approach (using one prediction step after the sample compression) taken was just to use a fixed value for q_o (like 30 in figure 3). However, the variation of the prediction accuracy was quite high using this approach - prediction was best in the neighbourhood of the sample parameter value.

In order to decrease the difference between q_o and q_x, we experimented with a *two-step* prediction, which uses two prediction steps – one before the test compression and one after it. First, we look up the parameter q_o using the method described above, setting $\Delta = 0$. This value of q_o – which will lead to the required compression ratio when used as parameter for

[11] The term *quality parameter* is somewhat misleading as this parameter does not influence the image quality directly. It has an indirect influence, however, by controlling the quantisation.
[12] This takes one second for a 512 by 512 pixel image on an SGI Indy workstation.

compressing the "average image" – is used as a first approximation for q. A better approximation for q is computed by doing a sample compression using q_o and proceeding as in the one-step approach.

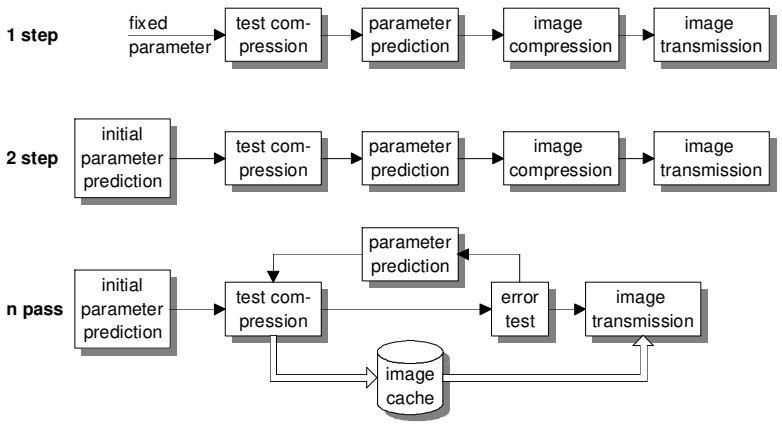

Figure 4: One step, two step and n pass method

An opportunity to increase accuracy further is to use a *multi-pass* method, which is an extension of the two step method. To refine the result of the initial sample compression, multiple sample compressions are performed when the error of the previous prediction is too high. The predicted parameter of one step is used as the sample parameter of the next step. Iteration must be stopped when the time needed for additional test compressions exceeds the possible gain in transmission time. The result of the last test compression is cached and can be immediately transmitted when the accuracy is high enough.

2.4.3 Experimental Evaluation

We conducted experiments to evaluate the methods described above. For this reason, we extended our test image database from [RSS96] to include images of different classes and numbers of colours: scanned photographs and paintings, computer-generated imagery, drawings, cartoons, maps and screen dumps. Furthermore, we included downscaled and cropped versions of each image into the test image set. On each of these images, we ran the one-step, the two-step and a two-pass method with requested compression ratios from 1:2 up to 1:300. We recorded the predicted quality parameter and compression ratio as well as the compression ratio achieved. Records where excluded when lossless compression methods were more suitable for the image than JPEG or when the requested compression ratio could not be reached by the image.

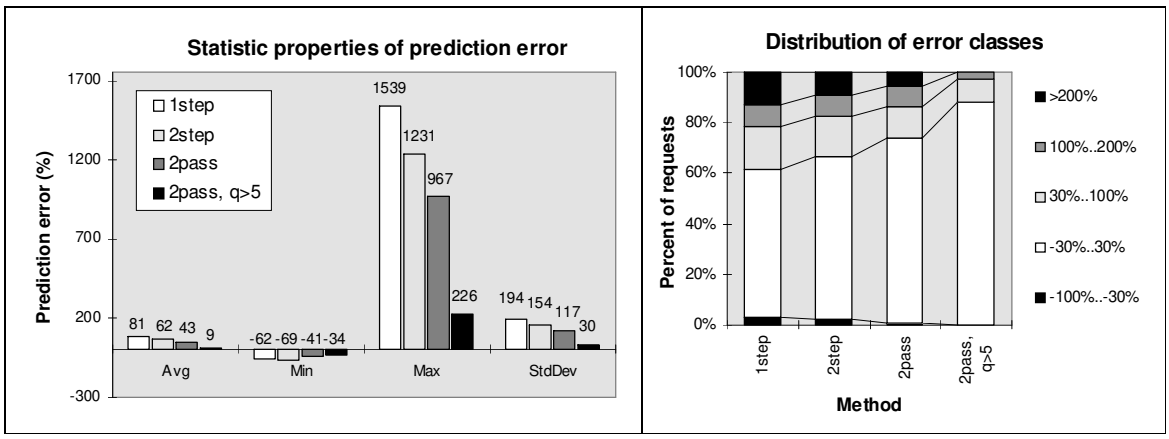

Figure 5: Average, minimum, maximum and distribution of the relative error for the different methods

2.4.4 Discussion of the Results

The results of the evaluation are shown in the figures 5 to 7. As a measure of quality of our prediction, we computed the relative error[13] e of the predicted compression ratio R_p to the achieved compression ratio R_a as follows: $e:=(R_p-R_a)/R_a*100\%$. Figure 5 shows the summarised results. It can be seen that the two pass method gives the best results - the average error is 43% with a minimum of -41%, a maximum of 967% and a standard deviation of 117%. Although for the latter method, 74% of the predictions do not deviate by more than 30% from the accurate value, this error margin is quite high, and we tried to find the reason for that and methods to decrease it.

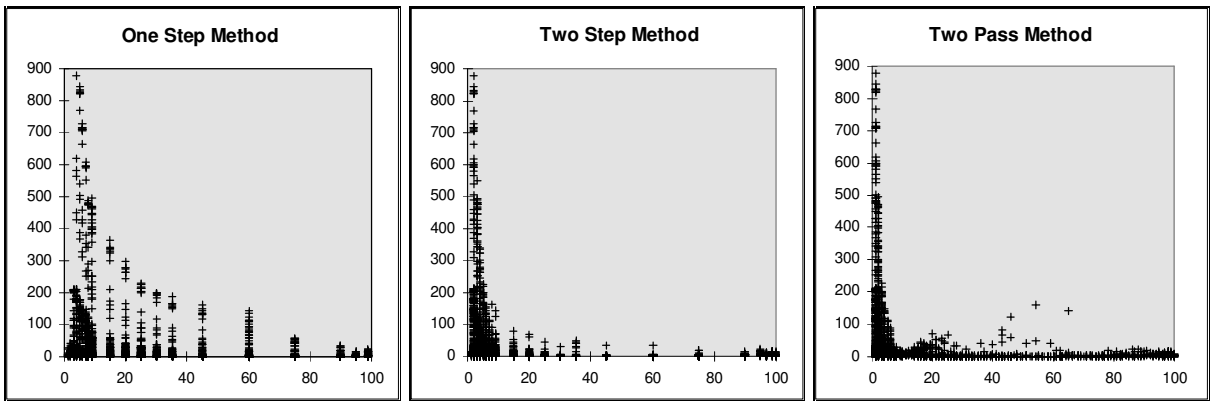

Figure 6: Scatter plot of relative prediction error (y axis) and `-quality` parameter (x axis)

Figure 6 shows scatter plots of the predicted `-quality` parameter and the relative error. It can be seen that the two pass method delivers the best accuracy and the one step method the poorest. Furthermore, the biggest errors and largest variation occur at `-quality` values less than six. Figure 7 depicts the reason for that: the variation of the compression ratio between individual images is very high at low `-quality` values, a fact which our initial experiments (see figure 1) did not capture.

For the `-quality` range from 6 to 100, the two pass method works acceptably well, as figure 5 (2pass, q>5) shows: the average error drops to 9%, the standard deviation to 30%. 88% of the predictions do not deviate by more than 30% from the correct value. Lower `-quality` settings than 6 lead to very poor image quality, and are frequently recommended

[13] A relative error of 100% means that the compressed image is twice as large as requested (taking twice the time to transmit).

not to be used (for instance by the Independent JPEG Group, see [IJG96]). However, as these parameter values lead to a high compression ratio, it may sometimes be desirable to use them in order to get a first impression of an image. To cope with the high error in this parameter interval, *multi-pass prediction* must be used in order to increase accuracy at the cost of a higher pre-processing time. Especially for negative errors, the decrease of the error (and thus of transmission time) in each pass must be monitored and compared to the additional time required for a new iteration. If the benefit drops under the cost of a new cycle, the image should either be transmitted using the approximation found so far (for rather small errors), or the *user* should be informed if the image transmission will take a much longer time than requested (for large errors). Alternatively, a notion of *quality of service* can be used which allows the user to control the error margin: „guaranteed“ will try to decrease the error to nearly zero at the cost of a long pre-processing time, "best effort" will use the multi-pass method with an error threshold in order to deliver a moderate prediction error, and "no guarantee" will deliver inaccurate but fast results using the two step method.

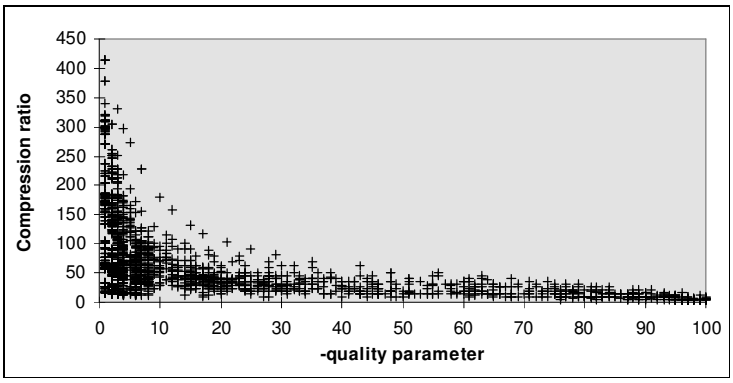

Figure 7: Scatter plot of compression ratio and `-quality` parameter

3 An Experimental System

3.1 Overview

We built an experimental system using World Wide Web technology. The steps performed by the image server in order to process an image request are illustrated in figure 8.

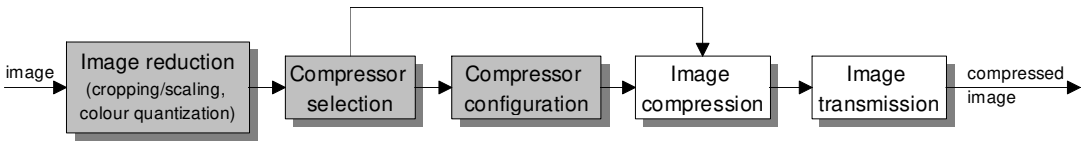

Figure 8: Tasks of the image server

An *image request* sent by the client to fetch an image specifies an *image identifier* and the *context* the image has to be adapted to. Figure 9 and figure 10 show the user interface. The *display context* and the preferred resize method (images are either scaled or cropped) control the *image reduction* step. The *network context* and the *maximum transmission time* determine the requested image size and thus control the *compressor selection* step and the *compressor configuration*. Last but not least, the *software context* is used as a filter during compressor selection.

3.2 A Demo Scenario

To demonstrate how our prototype selects a compressor and configures it depending on the image content, we prepared an image in the image base such that it contains areas of different number of colours. In the upper left corner of a 512 by 512 pixel 64K colour image, a small four colour picture of the globe has been placed.

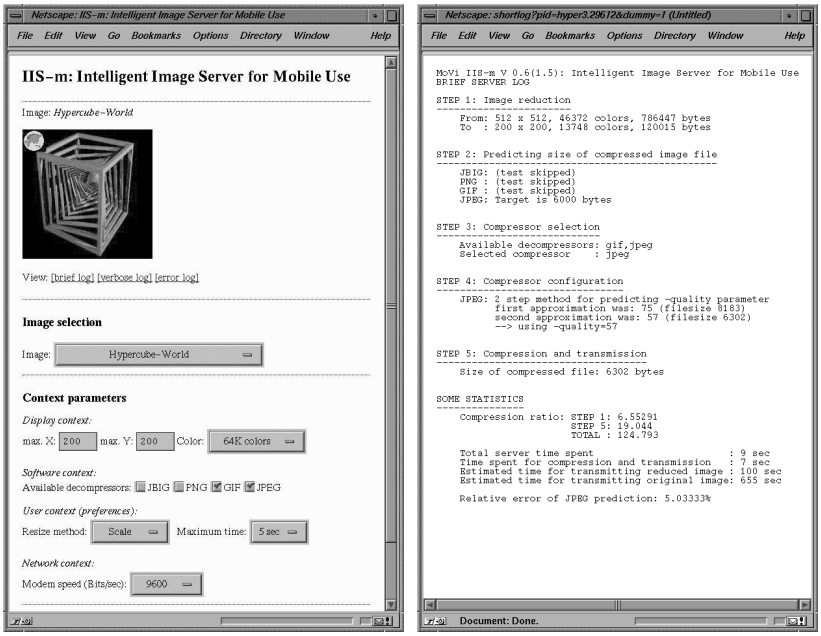

Figure 9: Retrieving a 64K colour image

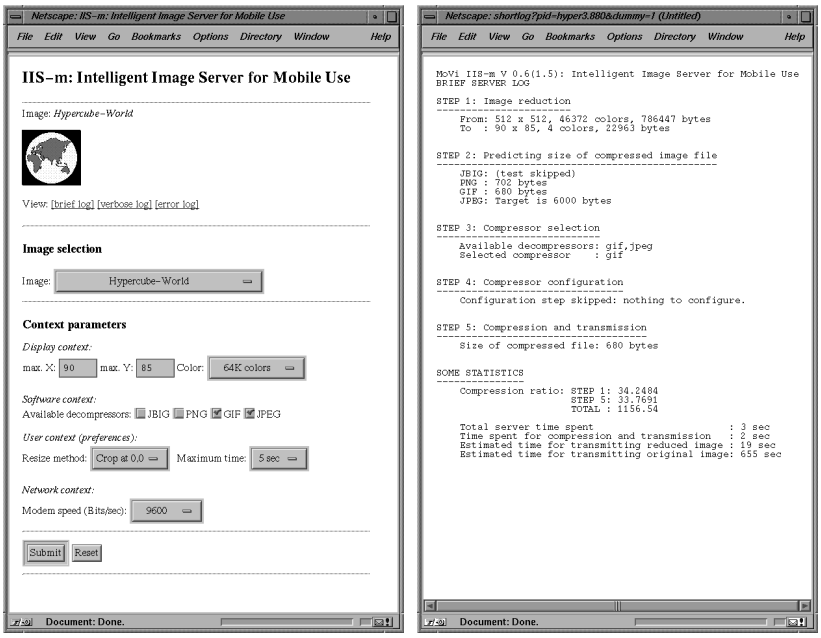

Figure 10: Retrieving a part of the image which only contains four colours

The request shown in figure 9 retrieves a downscaled version of the whole image, specifying a transmission time of five seconds and a bandwidth of 9600Kbit/s. Due to the high number of colours, JPEG is the only candidate compressor. A suitable value for the `quality` parameter is computed using the two step approach, and the server log shows a moderate error.

Figure 10 shows another request which cuts out the four colour drawing in the upper left corner of the image just by specifying a new image size and cropping at the origin as the resize method. In this case, GIF, PNG and JPEG are tested as candidates. As PNG and GIF deliver smaller files for this image than JPEG at 100% quality, they are used as candidates for compressor selection. Since the viewer can only display GIF images, GIF is selected and the cached image file from the GIF test compression is sent to the client.

Note that this different handling of the images is fully dynamic and based on the image properties after the reduction step. This dynamic adaptation of the image to the properties of the client computer and the transmission link guarantees a flexible image handling in environments with dynamically changing resources.

4 Conclusions and future work

To provide a means for compensating the shortcomings of static image transmission used today in very dynamic, globally distributed information infrastructures, we introduced the

concept of adaptive image transmission, presented a prototypical system and discussed the results of an experimental evaluation of that system.

We found that adaptive image transmission can save a substantial amount of bandwidth, thus allowing faster image transmission. However, the experimental evaluation showed a large variation of the parameter value prediction accuracy. We suggested two simple methods consistent with our approach for improving accuracy.

Instead of using the indirect prediction approach presented in this paper, future work will focus on a direct approach to control the compression ratio of JPEG to improve accuracy and increase speed. After the discrete cosine transform (DCT) step, an analysis step will be introduced which computes statistic properties of the DCT coefficients and accordingly controls quantisation, as it is already done in the bitrate control of real-time video compressors (see, e.g., [KSK95]). Taking this a step further, *levels of detail* can be generated based on the analysis by using the JPEG progressive mode. Breaking an image up into levels of detail supports the transmission of higher quality levels *on demand* to refine an already transmitted degraded quality version of an image without having to re-transmit data already transferred.

Ongoing work on an Image Compression and Analysis Workbench is expected to provide better predictors for the lossless algorithms which will hopefully allow us to drop the test compressions done currently.

Acknowledgements

The authors wish to thank Randolf Schultz and Ralf Jubin for their contribution of ideas, the conduct of experiments and for design and implementation work. The work presented is part of the MoVi project supported by the German Science Foundation (DFG) under contract no. Schu 887/3-1.

References

[CCI92] CCITT Draft Recommendation T.82, ISO/IEC Draft International Standard 11544 Coded Representation of Picture and Audio Information - Progressive Bi-Level Image Compression, 1992

[Cro95] Crocker, L.D.: PNG: The Portable Network Graphic Format, in: Dr. Dobb's Journal, Vol. 20, July 1995

[Fis95] Fischer, Y. (ed): Fractal Image Compression: Theory and Application, Springer, New York, 1995

[HJS94] Hilton, M.L.; Jawerth, B.D.; and Scngupta, A.: Compressing Still and Moving Images with Wavelets, Multimedia Systems, Vol. 2, No. 3, 1994

[IJG96] Manual page of the public domain JPEG compressor of the Independent JPEG Group, Release 6a, available via ftp://ftp.uu.net/graphics/jpeg/

[Jaq90] Jaquin, A.: Image Coding based on a Fractal Theory of Iterated Contractive Function Systems, SPIE Vol. 1360, Visual Communication and Image Processing, 1990.

[KSK95] Keesmann; Shah; Klein-Gunnewieck: Bitrate Control for MPEG Encoders; in: Signal Processing: Image Communication, Vol. 6, 1995

[PMi93] Pennebaker, W.B.; and Mitchell, J.L.: JPEG Still Image Data Compression Standard, Van Nostrand Reinhold, New York, 1993.

[RSS96] Rauschenbach, U.; Schultz, R.; and Schumann, H.: Quality and Resource Controlled Transmission of Images; in: Urban, B. (ed.): "Multimedia '96" Proceedings of the EUROGRAPHICS workshop in Rostock, Germany, May 28-30, 1996, SpringerWienNewYork, 1996, ISBN3-211-82876-1.