Resource-Optimized Delivery of Web Images to Small-Screen Devices*

Yunnan Wu[a] and Daniel Lopresti[b]

[a]Dept. of Electrical Engineering, Princeton University, Princeton, NJ 08544. U.S.A.
[b]Bell Labs, Lucent Technologies Inc., 600 Mountain Ave., Murray Hill, NJ 07974. U.S.A.

ABSTRACT

A framework for real-time adaptive delivery of web images to resource-constrained devices is presented, bringing together techniques from image analysis, compression, rate-distortion optimization, and user interaction. Two fundamental themes in this work are: (1) a structured and scalable representation, obtained through content- and lower-level image analysis, that allows multiple descriptions of object regions, and (2) resource-optimized content adaptation in real time, facilitated by an algorithm for directly merging LZ77-compressed streams without the need for additional string matching.

Also introduced is a new distortion measure for image approximations based on a feature space distance. Using this measure, a color reduction algorithm is proposed. Simulation studies show that this algorithm can yield better results than previous approaches, both from a visual standpoint and in terms of feature space distortion.

Keywords: WWW, image transcoding, text detection, color reduction, Lempel-Ziv compression, resource-constrained device, PDA.

1. INTRODUCTION

1.1. Challenges in Pervasive Web Access

The needs for ubiquitous information access are expanding at a dramatic rate, as reflected in the growing popularity of portable small-screen devices, e.g., PDA's. Delivering web content to these "thin" clients involves many technical challenges. These devices are constrained in various resources:[1] (1) screen size and resolution, (2) color depth, (3) computing power, (4) memory and storage, and (5) bandwidth. Although some service providers customize web pages specifically for hand-helds, this solution is costly and does not scale given the enormous number of existing web pages designed for traditional displays and the increasing diversity of clients.

An attractive alternative is to transcode existing web pages at source servers, proxies, or clients. Much research has been conducted along these lines, e.g., from a systems perspective,[1] in terms of user interfaces,[2] for text summarization,[3] etc. Among the various components present in a typical web page, images convey the most explicit visual information. At the same time, they consume a significant portion of the "bit budget." Hence, effective handling of images is essential to improving the perceived quality-of-service by end users.

It has been observed that a considerable amount of information is embedded in the text in web images.[4] Quantitative studies have further confirmed this result.[5,6] In one survey,[5] 17% of the total number of words visible on a WWW page were in image form, and 76% of the words in image form did not appear elsewhere in the encoded text. In another study,[6] 42% of the images in the sample contained text, and 59% of the images with text contained at least one word that did not appear in the corresponding HTML file. Given limited bandwidth and display sizes, the text portion of an image is likely to be the most valuable for browsing. Motivated by these observations, in this paper we investigate the efficient delivery of web images to small-screen devices, with an emphasis on techniques to preserve and present the embedded text information.

* To be presented at *Document Recognition and Retrival X (IS&T/SPIE Electronic Imaging)*, Santa Clara, CA, January 2003.

1.2. Natural vs. Synthetic Images

Web images fall into two categories: natural images, such as photos, and synthetic images, such as graphics. This dichotomy is reflected in the different approaches to compression.

Natural images are characterized by their richness in color (typically 24-bit true color) and the smooth transitions between pixels. Prevailing compression algorithms (JPEG,[7] JPEG2000[8]) are composed of the following steps: transformation into the frequency domain, quantization, and lossless coding of the quantized coefficients. Synthetic images are usually created using graphics software. Their defining characteristics are a limited number of colors and an abundance of sharp edges. Representative compression schemes for synthetic images are GIF[9] and PNG,[10] both based on the lossless Lempel-Ziv compression[11] of the image in 1D raster scan format.

Whether an image is natural or synthetic, it may contain overlaid text. From the standpoint of resource-constrained devices (e.g., PDA's), synthetic images are more germane because they tend to be smaller and usually require fewer colors. Consequently, in this work our focus is on transcoding synthetic images, preserving the textual content while simplifying other portions of the image.

1.3. Scalability and Real-Time Adaptivity

Adaptively streaming web images to heterogeneous clients with various resource constraints requires per-client customization or transcoding. With the proliferation of client devices, it is important to reduce the processing and storage overhead. One way of achieving this objective is to maintain a scalable representation of source images and design real-time algorithms to compose efficient renderings adaptive to the given resource. Here the term *scalability* means that the image is compressed only once into a single bitstream, and parts of the bitstream can be extracted and decoded to generate representations commensurate with the proportion of the bitstream decoded. The term *real-time adaptivity* refers to the existence of fast algorithms to generate final output tailored to the resource constraints of each end user.

For natural images, JPEG2000 has been proven to be a successful example of supporting resolution/quality scalability. Scalability is enabled by the multi-resolution and space-frequency localization properties of the wavelet, independent block encoding of wavelet coefficients, embedded bitplane coding, and structured bitstream syntax. For real-time adaptivity, an interactive network image browsing system has been proposed mainly for viewing large images,[12] demonstrating the superior advantages of the scalable compression of JPEG2000.

Scalability in JPEG2000 is supported in the frequency domain. However, such an approach is not applicable to synthetic images due to the different compression paradigm. Likewise, interaction-driven adaptation[12] is not suitable for streaming most web images as they are typically small in size. In this paper, we seek to achieve content-level scalability and intelligent adaptation working in the spatial domain without much user interaction. We approach the scalability problem by combining image analysis and compression techniques. More specifically, we focus on text region processing as a concrete example of content-level image analysis. The original image is analyzed to generate a hierarchical structure of object regions, each having multiple descriptions. This structured representation is encoded once and adapted optimally on-demand to fit the bandwidth and display constraints. To address the real-time requirement, a fast algorithm is proposed for directly merging LZ77-compressed streams, thereby avoiding the need to decompress, merge in the time domain, and recompress.

1.4. Paper Organization

The remainder of this paper is organized as follows. In Section 2, we describe the overall system architecture. Sections 3, 4, and 5 are devoted to image analysis and compression, content adaptation, and flexible display, respectively. Experimental results are given in Section 6. Finally, Section 7 concludes the paper.

2. SYSTEM ARCHITECTURE

The proposed adaptive delivery system is comprised of three components: image analysis and compression, content adaptation, and flexible display, as illustrated in Figures 1(a), 1(b), and 1(c).

In Figure 1(a), at a proxy server, an input image is first analyzed to identify rectangular bounding boxes around text regions. Color reduction and down-sampling are then applied to each image region to form approximations for lower-quality rendering. After that, the approximations are compressed and their rate-distortion

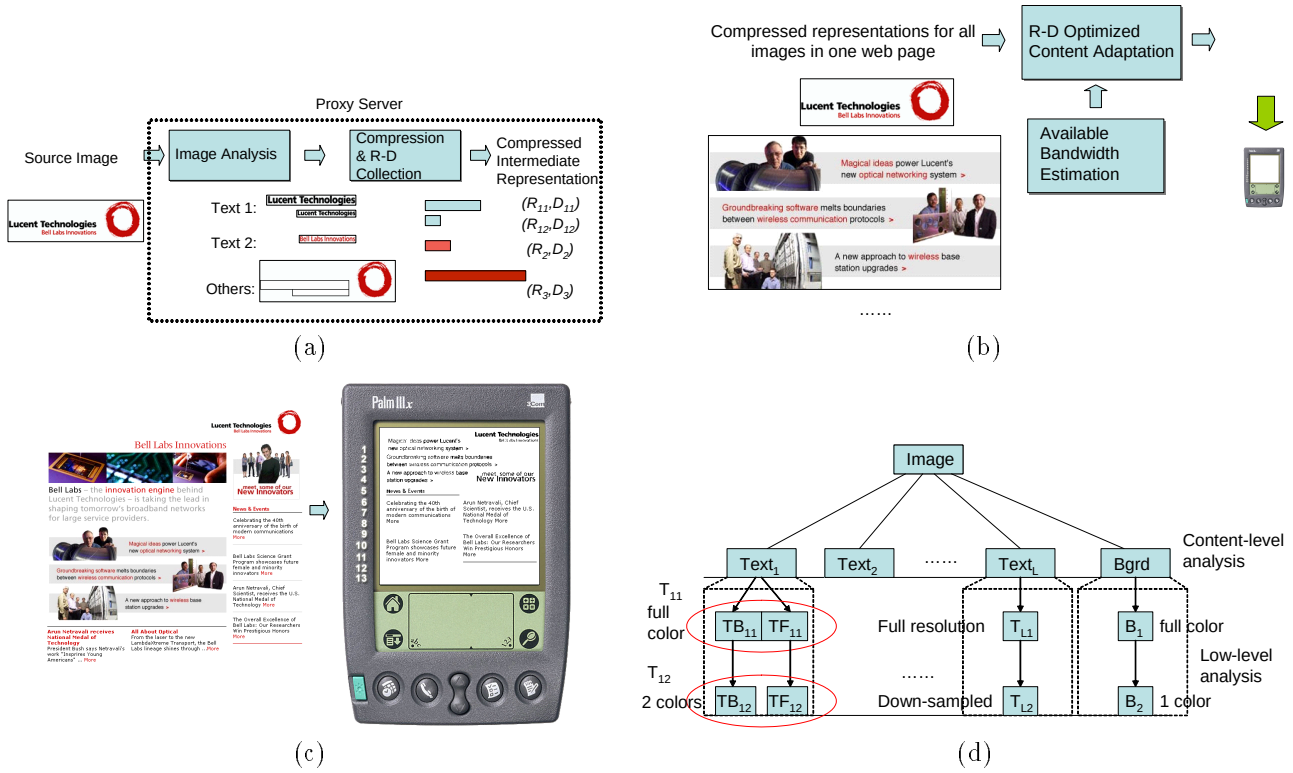


Figure 1. (a) Image analysis and compression, (b) Rate-distortion optimized adaptation, (c) Flexible interactive display, (d) The data structure.

information is collected. Each time a web page is accessed, the content adaptation process is invoked to optimally allocate the resources among the images and transcode them, as shown in Figure 1(b). This optimization relies on an external module to supply data on the available bandwidth.

Finally, at the client device, the decoder and rendering system interacts with the user and customizes the display. Figure 1(c) illustrates an example web page and what an adapted version might look like on a PDA. In this case, all of the non-text graphics have been discarded and the page layout has been re-organized. Since the large image in the upper-left corner of the original web page has no hypertext link associated with it, the user may click on it to remove it from the current window, as demonstrated in the adapted view in the figure.

Among the three components, image analysis and compression has the highest computational complexity. However, we note that this step needs to be carried out only once and may be done off-line at the server.

3. IMAGE ANALYSIS AND COMPRESSION

The essential goal of image analysis and compression is to develop a spectrum of approximations of the original image which require less bits and have fewer colors and/or coarser resolutions. In addition, the rate-distortion tradeoffs of the different approximations should be quantified for later optimal resource allocation.

3.1. Content-level Image Analysis

The objective of content-level image analysis is to extract structural information from a given structure-less image. This problem can be viewed as the inverse of image authoring/composition. During the authoring stage, most graphics software maintains a collection of independent objects and their respective shapes, textures, locations, and layers. In content-level image analysis, we wish to decompose the image into "objects" corresponding to semantically meaningful entities. A concrete example, which we employ in the sequel, is text regions defined by rectangular bounding boxes.

There is a rich literature on document layout analysis, and text localization in particular.[4, 13–18] However, traditional document image analysis is motivated by recognition tasks, e.g., OCR. Although encoded text is indeed a compact representation for the information contained in an image, full recognition is computationally demanding. Moreover, OCR errors may jeopardize a user's perception. Web images are particularly difficult as they typically employ lower spatial resolutions than scanned documents.

In contrast, the content-level image analysis pursued herein is driven by compression and delivery needs. Rather than attempt to recognize the text for the user, we simplify the image while preserving the text regions and leave recognition to the user. This difference allows us to take advantage of well-established pre-processing methods for OCR, such as text localization, without having to employ later-stage (and more problematic) techniques like character segmentation. Nevertheless, it is not our intent in this paper to discuss specific algorithms for text localization. Instead, we acknowledge their existence and focus on discussing how to make effective use of them in developing a scalable compression scheme and an adaptive delivery system.

3.2. Low-level Image Analysis and Transformations

In our goal of constructing image approximations, image analysis is broad in its meaning; it includes low-level image analysis as well as transformations. In this section, we consider color reduction and down-sampling.

Approximations techniques are always associated with a quality measure. Unfortunately, most existing quality measures are based solely on pixel-by-pixel differences and thus not indicative of human perception. A general rule of thumb can be stated as follows: a lower-quality image tries to approximate its original by maintaining color and spatial features. For a text region, preserving the strokes and edges is more important than keeping the pixel colors precise. Based on this observation, we propose a quality measure that combines color and spatial feature distance. Let the original image be $\mathbf{A}$ and the approximation be $\mathbf{B}$. Given a set of N feature definitions, both images are represented as a collection of feature vectors $\underline{a}_m \equiv (a_{m1}, a_{m2}, \cdots, a_{mN})^T$, $m = 1, \cdots, M$ and $\underline{b}_m \equiv (b_{m1}, b_{m2}, \cdots, b_{mN})^T$, $m = 1, \cdots, M$, where M can be less than or equal to the number of pixels in the image, depending on whether or not the feature vector is computed over a subset of pixel positions. We then measure the distance between two images as the distance in the feature vector space:

$$FD(\mathbf{A}, \mathbf{B})^2 \equiv \sum_{m=1}^{M} \| \underline{a}_m - \underline{b}_m \|^2 \,. \tag{1}$$

An example of feature definition: In the experiments for this paper, we adopt four features for each color component (R, G, B), each being a 2-D linear filter on a 3×3 window:

$$\mathbf{F}_0 \equiv \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{F}_1 \equiv \frac{1}{2} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \quad \mathbf{F}_2 \equiv \frac{1}{2} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad \mathbf{F}_3 \equiv \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{2}$$

The first feature, $\mathbf{F}_0$, is just the pixel intensity at the center. The second and third features, $\mathbf{F}_1$ and $\mathbf{F}_2$, are the horizontal and vertical Sobel edge detectors, which approximate the first derivatives of the image. The last feature, $\mathbf{F}_3$, is the Laplace operator which approximates the second derivative of the image. Their weightings as given above were selected through empirical evaluation.

3.2.1. Color reduction

One way to approximate an image is to reduce the number of colors it uses. If the feature definitions are the R, G, B components, this is simply an unsupervised clustering problem. Therefore, techniques like k-means can be applied. In order to bring spatial features into consideration, we adopt more general feature definitions and the cost function defined by Equation (1).

Given the target number of colors in the output image, we propose Algorithm 1 to reduce the colors in the feature space. The algorithm operates iteratively by alternating between updating the color association of each pixel and updating the assignment of color palettes. It is guaranteed to converge since each step can only reduce the cost function. Because we have assumed 2D linear shift-invariant filters with finite support (e.g., a 3×3 window), Algorithm 1 can be implemented efficiently with pipelining and using the linear superposition of impulse responses. Due to space limitations, these optimizations are not discussed here.

Algorithm 1 Color reduction on the feature space

Initialize the color assignment and association with the result of color reduction based on color space only.
repeat
  /* Sequentially update the color association. */
  for each pixel position **do**
    Given the up-to-date color assignment of all other pixels, choose the optimal color from the current palette which minimizes the summed distance of all the feature vectors which are affected by the current pixel.
  end for
  /* Sequentially update the color assignment. */
  for each color cluster **do**
    Consider the color assignment by averaging all the pixels using the same color association.
    if this average operation reduces the distortion measure **then**
      Adopt the new color assignment.
    else
      Retain the old color assignment.
    end if
  end for
until The feature domain distortion cannot be reduced further.

3.2.2. Image down-sampling

Image down-sampling is another well-known approximation for which many algorithms exist. For text bounding boxes, a suitable down-sampling ratio can be selected with heuristic knowledge of legible font sizes. However, we still require a systematic way to measure the reduction in quality. In this section, we demonstrate how to fit down-sampling into our optimization framework.

Our approach is based on a simple idea: since the receiver can always perform up-sampling, the distance between the down-sampled image and the original can be obtained by using the up-sampled image for comparison. Minimizing this measure immediately leads to an algorithm for down-sampling. The problem manifests itself as a structure-constrained optimization, i.e., the pixels in an up-sampled block are constrained to be of the same color. Iterative optimization is still applicable. In fact, the framework of Algorithm 1 can be easily extended by (1) initializing with a simple down-sampling operation, and (2) treating an up-sampled block as a unit and considering the change in the summed squared distance caused by a change in the color association for the unit.

3.3. The Data Structure

For the remainder of this paper, we shall call each region obtained through content-level analysis an *object*. Thus, for the application in question, there are two categories of objects: text and background. Note that the latter refers to the portion of the image with the text regions cropped out. An approximation for an object obtained through low-level analysis is called a *description*.

Content-level and low-level image analysis facilitates a hierarchical decomposition of the image into a tree-structured representation. Similar data structures have appeared elsewhere.[19] There, however, the decomposition is based solely on color reduction and involves no content-level information.

We illustrate this tree-structure with an example shown in Figure 1(d). The image first undergoes a content-level decomposition where the bounding boxes of text regions have been identified. The remainder of the image is represented as a node labeled *Bgrd*. Each region is then further decomposed with low-level techniques. For example, a full-colored description for text region $Text_1$ is first given as T_{11}. Then the foreground and the background TF_1 and TB_1 can be identified and each represented with a single color. Thus, the text region is reduced to a binary image T_{12}. Text region $Text_L$ is represented by a chain of reduced resolutions, with T_{L2} corresponding to a down-sampled version. The background region is represented by a chain of two nodes: B_1 corresponds to the full color representation, and B_2 corresponds to the single color version.

3.4. Compression and Rate-Distortion Evaluation

To save space, the tree-structured representation is encoded. Conceptually, compression here involves nothing more than processing each description with a general-purpose algorithm such as LZ77 and recording all of the structural information. However, we note that further enhancements are possible that take into account the correlations among the multiple descriptions. We leave the de-correlation strategy as a future investigation.

To enable optimal content adaptation, rate-distortion information needs to be collected. More specifically, the sizes of the LZ77 compressed nodes are used for rate information. The quality measure defined in Equation (1) serves as the distortion indicator. In order to achieve content-level quality evaluation, the feature space distortion measure is further weighted for different objects (e.g., text box or non-text region). The weights can be assigned using heuristics and reflect relative importance. For example, for a small image such as a stylish navigation icon, the single color version of the background object should be assigned a very low distortion. The weight for an image with an associated hyperlink should be higher than for those without links. In general, the background object receives a lower weight than text boxes. In the experiments for this paper, we set a weight 1.0 for text boxes and 0.25 for background objects.

4. CONTENT ADAPTATION

Image analysis and compression are performed once for each image. The resultant compact representations are stored at the proxy server. Whenever a request is made for a web page, all images on the page are optimally transcoded for efficient delivery to the small-screen device.

4.1. Available Bandwidth Estimation

The available bandwidth can be estimated by monitoring the recent history of the link throughput. The simplest approach is to observe a time window and compute an average for the bandwidth based on this. Subtracting the bits reserved for other resources gives the overall bit budget for images on the web page, which we denote as B.

4.2. Rate-Distortion Optimization

Recall that during the image analysis and compression stage, the rate-distortion information for all object descriptions has been collected. The optimization then seeks to find the best combination of descriptions within the bit budget constraints.

Let $\mathbf{B}_{ij}$, $i = 1, \cdots, I$ denote the j-th description for the i-th object, $\mathbf{A}_i$ be the associated original description, and w_i be the weight for the i-th object. Denote the rate for $\mathbf{B}_{ij}$ by R_{ij}. Given a fixed bit budget B, the optimal selection of the object descriptions is formulated as follows:

$$\min_{\{j\}} \sum_{i=1}^{I} w_i \cdot FD(\mathbf{A}_i, \mathbf{B}_{ij})^2 \quad \text{subject to} \quad \sum_{i=1}^{I} R_{ij} \leq B \ . \tag{3}$$

This can be solved exactly by dynamic programming[21] and approximately by Lagrange multiplier techniques.[20]

4.3. Transcoding and Bitstream Assembly

After selecting the appropriate description for each image, the compressed bitstream is constructed. There are two possibilities for transcoding. If object-based decompression can be supported at the client device, the bitstream segments can simply be concatenated. This gives the decoder the flexibility to adapt the content locally according to user preferences. Otherwise, a standard format such as PNG or GIF should be used for the final output image. Since this option requires no modification in the client, it is easy to deploy. In this section, we discuss fast algorithms to compose the selected descriptions for a given image to form the output.

A naive approach would be to: (1) decompress the selected description for each individual object, (2) compose these into one image, and (3) re-compress the composite image. Our objective here is to take advantage of the already-compressed bitstream segments to facilitate the creation of the final compressed stream. In other words, we are interested in compression using prior information. Without loss of generality, our discussion assumes that PNG format is used both for the input and the output.

4.3.1. Background on PNG: Lempel-Ziv Compression with Huffman Coding

The PNG format[10] is generated by compressing the 1D raster scan of the original 2D image with DEFLATE,[22] which is essentially LZ77 with a further Huffman coding of the LZ77 outputs. LZ77[11] is a universal compression algorithm, where source statistics are learned "on the fly" from a window of past symbols. A rough explanation is as follows: given a window of past symbols, the substring at the current position is searched against the recent history for the longest possible match. The match that is found is recorded as a tuple ⟨length, backward distance⟩. In this way, redundancy is used to compress the string.

To assist in the discussion that follows, we provide a description of the compressed stream here.† Let $\mathcal{A} = \{0, 1, \ldots, |\mathcal{A}|\}$ denote the alphabet of symbols, where $|\mathcal{A}|$ is the number of elements. For example, in the current DEFLATE implementation, $\mathcal{A} = \{0, 1, 2, \ldots, 255\}$, i.e., each symbol is represented using a byte. A string $\mathbf{s}$ of length $|\mathbf{s}|$ is $\mathbf{s}(1)\mathbf{s}(2)\ldots\mathbf{s}(|\mathbf{s}|)$, $\mathbf{s}(i) \in \mathcal{A}$. A substring of $\mathbf{s}$ which starts at position i and ends at j is denoted by $\mathbf{s}(i:j)$. The compressed data consists of a series of elements of two types:

1. **Literals**: A literal is a symbol $\mathbf{s}_p \in \mathcal{A}$. Typically, this suggests that a string match cannot be found in the window of past symbols.
2. **Pointers**: A pointer is a pair $\langle l, d \rangle$ where l is the length of the matching substring, and d is the backward distance. If p denotes the current position in the source string, then $\mathbf{s}(p-d : p-d+l-1) = \mathbf{s}(p : p+l-1)$.‡

4.3.2. Merging LZ77 Compressed Descriptions

In this subsection, we consider the problem of merging several compressed streams for the descriptions making up one image. LZ77 decompression involves simple string copying and hence is fast. For compression, the most time-consuming step is the string matching procedure. Therefore, our main objective is to avoid or at least minimize the string matching, given the already-compressed segments as input. The algorithm we propose takes several compressed streams of literals and pointers, updates the pointers, and merges them into one compressed stream. Merging is of course suboptimal relative to full string matching as the cross-object pattern repetitions are not explored. However, the complexity is reduced as a result. There is one subtlety to deal with in the merging procedure, though, which occurs at boundaries when objects are interleaved. Taking a pointer $\mathbf{s}(p-d : p-d+l-1) = \mathbf{s}(p : p+l-1)$ as an example, there are three types of boundary effect:

1. **Broken reference.** In this case, $\mathbf{s}(p-d : p-d+l-1)$ is split into two or more pieces.
2. **Broken source.** In this case, $\mathbf{s}(p : p+l-1)$ needs to be split into two or more pieces.
3. **Broken reference and source.** This occurs when both the reference $\mathbf{s}(p-d : p-d+l-1)$ and the source $\mathbf{s}(p : p+l-1)$ are broken.

A straightforward solution is just to compress $\mathbf{s}(p : p+l-1)$ as a string of literals. A better solution is to split $\mathbf{s}(p : p+l-1)$ (and $\mathbf{s}(p-d : p-d+l-1)$) accordingly. Note that the number of boundary points is proportional to the number of objects in the image. When the number of objects is small, even the straightforward solution should work well. The detailed specification of the algorithm is given below.

Let $\mathbf{s}_i(j)$ for $i = 1, 2, \ldots, S$ and $j = 1, 2, \ldots, |\mathbf{s}_i|$ denote the strings to be merged, and let $\mathbf{o}(k)$ for $k = 1, 2, \ldots, \sum_{i=1}^{S} |\mathbf{s}_i|$ denote the output string. Both $\mathbf{s}_i(j)$ and $\mathbf{o}(k)$ are in the time domain. Assume that a translation table $\mathbf{T}(i, j)$ for $i = 1, 2, \ldots, S$ and $j = 1, 2, \ldots, |\mathbf{s}_i|$ has been established, mapping the source symbol $\mathbf{s}_i(j)$ to the output symbol $\mathbf{o}(k)$. That is, $k = \mathbf{T}(i, j)$. Let the compressed stream corresponding to $\mathbf{s}_i$ be denoted as $\mathbf{LZ}_i$, which is a list of literals and pointers.

With the translation table $\mathbf{T}(i, j)$, the merging of the LZ77 streams is given in Algorithm 2. We maintain $\mathcal{L}$ as a queue of $\mathbf{LZ}_i$, $i = 1, 2, \ldots, S$, each a stream of literals and pointers. The merging can be done sequentially

†Note that this is the version used in practice and that it differs slightly from the original algorithm.[11]

‡In DEFLATE, $l \in \{L_{low} = 3, 4, \ldots, L_{upp} = 258\}$ and $d \in \{1, 2, \ldots, W\}$ where $W = 2^w$ is the history window size. The default window size is $W = 32{,}768$. Restricting the window size is useful in providing an upper bound on the memory usage at both the encoder and decoder.

Algorithm 2 Merging LZ77 Streams without string matching

Initialization: $p_i = 1$, $i = 1, 2, \ldots, S$; $k = 1$. Sort $\mathcal{L}$ in increasing order on $\mathbf{T}(i, p_i)$.
while $(k \leq \sum_{i=1}^{S} |\mathbf{s}_i|)$ **do**
  $\mathbf{LZ}_i = PopHead(\mathcal{L})$
  $\mathbf{cur}_i = PopHead(\mathbf{LZ}_i)$
  if $\mathbf{cur}_i = \mathbf{s}_i(p_i)$ **then**
    $InsertTail(\mathbf{LZ}_{out}, \mathbf{s}_i(p_i))$
    $p_i = p_i + 1$; $\mathbf{o}(k) = \mathbf{s}_i(p_i)$; $k = k + 1$
  else if $\mathbf{cur}_i = \langle l, d \rangle$ **then**
    $m^* = \max_{m \in \{1,\ldots,l\}} \{m \mid \mathbf{T}(i, p_i + m - 1) = \mathbf{T}(i, p_i) + m - 1,\ \mathbf{T}(i, p_i - d + m) = \mathbf{T}(i, p_i - d) + m - 1\}$
    $\hat{d} = \mathbf{T}(i, p_i) - \mathbf{T}(i, p_i - d)$
    if $(m^* = 1)$ **then**
      $InsertTail(\mathbf{LZ}_{out}, \mathbf{o}(k - \hat{d}))$
    else
      $InsertTail(\mathbf{LZ}_{out}, \langle m, \hat{d} \rangle)$
    end if
    $p_i = p_i + m^*$; $\mathbf{o}(k : k + m^* - 1) = \mathbf{o}(k - \hat{d} : k - \hat{d} + m^* - 1)$; $k = k + m^*$
    if $l > m^*$ **then**
      $InsertHead(\mathbf{LZ}_i, \langle l - m^*, d \rangle)$
    end if
  end if
  Insert $\mathbf{LZ}_i$ into $\mathcal{L}$ so that $\mathcal{L}$ is increasing in $\mathbf{T}(i, p_i)$.
end while

over the elements of $\mathbf{LZ}_i$. Literal bytes are simply copied into $\mathbf{LZ}_{out}$. Pointers are updated and split if necessary according to the discussion above. In Algorithm 2, p_i denotes the current position in $\mathbf{s}_i$, i.e., $\mathbf{s}_i(p_i - 1)$ has been merged, but $\mathbf{s}_i(p_i)$ has not been merged, while k denotes the current position in the combined output stream, $\mathbf{o}$.

4.3.3. Extensions and Possible Enhancements

Several extensions/enhancements are possible:

1. Note that a dual splitting process, that is, parsing out objects from a compressed image without fully decompressing it, can easily be constructed in a similar fashion. However, since image analysis requires decompression and objects are modified, this does not present any benefits for our application.

2. In DEFLATE, Huffman coding is applied to the LZ77 outputs to eliminate any remaining redundancies. Therefore, direct merging algorithms for Huffman-coded streams would further reduce the computational cost. Indeed, this should be possible as the statistics for the literals and lengths/distances can be summed to construct a combined Huffman tree and reassign the codewords. However, since the complexity of Huffman coding is low compared to LZ77 compression, this would yield only a minor improvement.

3. The compression for individual descriptions can be configured so that potential boundary effects are reduced by a constrained string matching procedure.

4. If further compression is desired, the output of Algorithm 2 can be treated as *suggestions* for the string matching process. The key observation is that search complexity is reduced if the initial solutions are good.

5. FLEXIBLE INTERACTIVE DISPLAY

The decoder and rendering system are ideal places to incorporate user preferences and interaction as the communication overhead is minimal. Since screen space and memory are assumed to be limited, only images currently being displayed or likely to be viewed in the near future need to be decompressed. The layout of the modified web page should be arranged by the rendering system based on user feedback. With our proposed transmission

scheme, it is likely that certain images contain blank regions. To use the display space economically, the user may set the rendering system to automatically detect the blank regions and use the space for other more important information. Alternatively, he/she may click on these regions to remove them manually. A hand-drawn example of a customized display is given in Figure 1(d).

6. EXPERIMENTS

The framework established in this paper has been fully implemented and tested. Here we report our findings on the following four aspects of the system:

1. Effectiveness of the new color reduction algorithm in terms of visual quality and feature space distance.

   For the test image in Figure 2(a), Figures 2(b), 2(c), and 2(d) present the binarized output using a k-means algorithm, a benchmark adaptive color reduction (ACR) algorithm,[19] and Algorithm 1. It can be observed that the approach we have proposed delivers the best visual quality among the three as the string "The GNU Image Manipulation Program" is more legible. The string "gimp-selection-..." in a smaller font is de-emphasized by Algorithm 1 since its response to the 3×3 feature detectors is not salient.

   Quantitative measures of the distortion are given in Figures 4(a) and 4(b), in terms of RGB space distance and feature space distance. It can be seen that Algorithm 1 performs the best in minimizing the feature space distortion. Note, however, that it exhibits the largest RGB space distortion among the three. Hence, it appears that the feature space distortion measure is more consistent with human perception, especially for embedded text which contains a distinct edge structure.

(a) (b) (c) (d)

Figure 2. (a) Test image "The Gimp" with text boxes marked, (b) Binarization with k-means, (c) Binarization with the ACR benchmark,[19] (d) Binarization with Algorithm 1.

Figure 3. Test image "Contest" with text boxes marked.

2. Compression efficiency of the tree-structured intermediate representation.

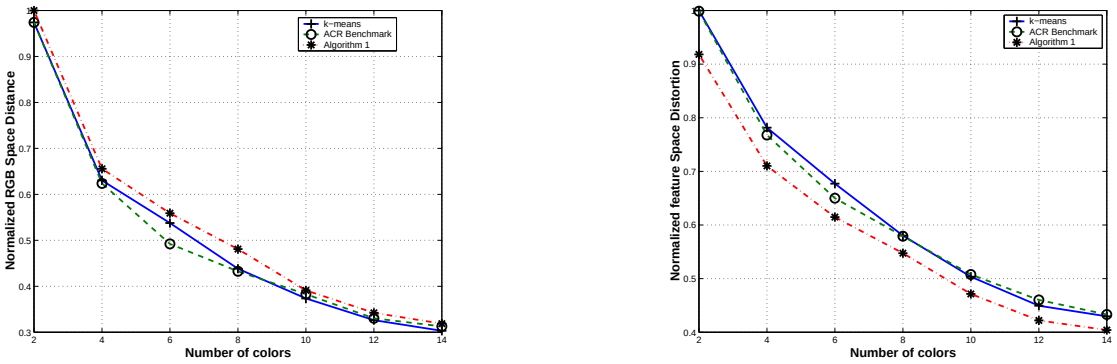


Figure 4. Color reduction performance in terms of: (a) RGB space distance, and (b) feature space distance.

Consider a simple web page with two images: "The Gimp"(Figure 2(a)) and "Contest"(Figure 3). The text bounding boxes are displayed as rectangles in the figures.[§] Originally, the compressed versions of "The Gimp" and "Contest" are 28.1 kbytes and 7.8 kbytes, respectively. The intermediate representations take 35.1 kbytes and 11.6 kbytes, corresponding to size increases of 24.9% and 48.7%. In general, the larger the input, the smaller the degree of expansion. Taking into account the additional structure we encode, the size increase does not seem significant. Moreover, the intermediate files are assumed to be stored on servers where there is usually sufficient space.

3. Effectiveness of the rate-distortion optimized image adaptation.

   Using the same simple web page as an example, the intermediate representations have a total length of 47.8 kbytes, while the original structure-less files are 35.9 kbytes. The rate-distortion curves for the two images are shown in Figure 6(a). The intermediate representation for "The Gimp" can produce 36 descriptions with different rate-distortion tradeoffs, illustrated by the rectangular datapoints in the figure. The 15 descriptions for "Contest" are indicated by diamonds.

   Given an available bit budget, we use the Lagrange multiplier approach to find the rate-distortion optimized adaptation. One restriction on this approach is that the points lying inside the convex hull (the solid curves in Figure 6(a)) cannot be reached during the search. Ruling out these interior points leaves 15 descriptions for "The Gimp" and 9 for "Contest."

   Figure 5 presents several sample images adapted with different budget constraints. It can be clearly observed that independent down-sampling and color reduction for each image object provides a flexible spectrum of possible descriptions. Sorting the line segments in Figure 6(a) by their slopes, altogether 24 $(= 15 + 9)$ output image combinations with different rate-distortions are feasible lying on the combined convex hull. Thus a given budget is quantized into one of the 24 rate intervals and the corresponding combination selected. In the figure, a number of these are shown along with their total file sizes, e.g., the second combination results in a file size of 359 bytes. In this example, the seventh output conveys the information most economically since the rate is low and nearly all of the embedded text is readable. In fact, this combination corresponds quite nicely to the "knees" in the rate-distortion curves (Figure 6(a)). This confirms that: (1) the feature space measure captures the visual distortion, and (2) the adaptation procedure is effective in utilizing the available resources.

4. Efficiency of the algorithm for merging LZ77-compressed streams.

   Algorithm 2 sacrifices some compression efficiency by not removing cross-object redundancies, but gains computational speed as a result. Figures 6(b) and 6(c) give the compressed file lengths for the 14 non-empty descriptions of "The Gimp" test image and the 8 non-empty descriptions of "Contest." (The first description has length 0, meaning that the object is omitted.) The expansion factors for Algorithm 2

[§] For the purposes of this study, the bounding boxes have been labeled by hand.

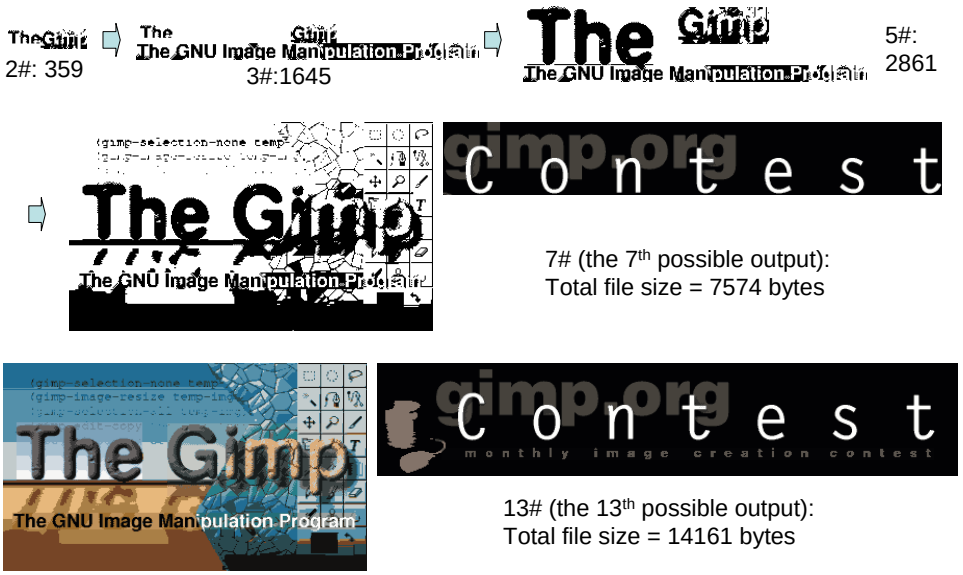
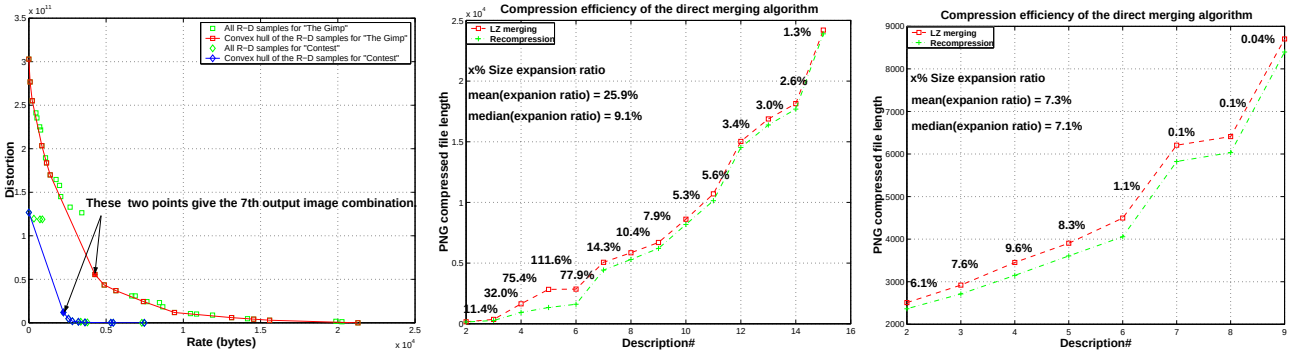


Figure 5. Adaptation results for a simple "web page" containing two images.



(a) (b) (c)

Figure 6. (a) Rate-distortion curves for the example in Figure 5, (b) Compression efficiencies of Algorithm 2 for "The Gimp," (c) Compression efficiencies of Algorithm 2 for "Contest."

are marked on top of the corresponding samples. The mean and median of the expansions for "The Gimp" are 25.9% and 9.1%, while for "Contest" they are 7.3% and 7.1%. Note that the 4th, 5th, and 6th descriptions for "The Gimp" have the largest expansion factors. These images are black-and-white with multiple embedded text objects, hence many patterns can be shared among the individual components. For images using more colors and/or larger dimensions, the size expansions are less significant, as shown by the other datapoints in Figures 6(b) and 6(c).

7. CONCLUSIONS

In this paper, a framework for the adaptive delivery of web images to small-screen devices has been presented. The proposed system provides content-level scalability by unifying image analysis, data compression, optimized content adaptation, and user interaction. Real-time adaptation is enabled by a fast algorithm that can directly merge multiple LZ77-compressed streams.

We described a way to establish rate-distortion information for each alternate description of an object. A feature space distance was introduced as a new distortion measure for image approximations. In addition, new

algorithms for color reduction and down-sampling were developed that minimize distortion in the feature space.

8. ACKNOWLEDGEMENTS

We would like to thank the providers of the zlib library, Jean-Loup Gailly and Mark Adler, for making their software available. The images used as examples in this paper are the property of their respective owners.

REFERENCES

1. R. Mohan, J.R. Smith, and C.-S. Li, "Adapting multimedia Internet content for universal access," *IEEE Trans. Multimedia*, vol. 1, no. 1, Mar. 1999.
2. G. Buchanan, S. Farrant, M. Jones, H. Thimbleby, G. Marsden, and M. Pazzani, "Improving mobile Internet usability," *10th International World Wide Web Conf.*, May 2001, Hong Kong, China.
3. O. Buyukkokten, H. Garcia-Molina, and A. Paepcke, "Seeing the whole in parts: text summarization for web browsing on handheld devices," *10th International World Wide Web Conf.*, May 2001, Hong Kong, China.
4. D. Lopresti and J. Zhou, "Locating and recognizing text in WWW images," *Information Retrieval*, pp. 177-206, 2000.
5. A. Antonacopoulos, D. Karatzas, and J. Ortiz Lopez, "Accessing textual information embedded in Internet images," *Prof. SPIE Internet Imaging II*, pp. 198-205, San Jose, CA, Jan. 2001.
6. T. Kanungo, C. H. Lee, and R. Bradford, "What fraction of images on the Web contain text?" *Proc. 1st International Workshop on Web Document Analysis*, Seattle, WA, Sept. 2001.
7. W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Compression Standard*, Kluwer Academic Publishers, Dec. 1992.
8. D. Taubman and M. Marcellin, "JPEG2000: standard for interactive imaging," *Proceedings of the IEEE*, vol. 90 no. 8, pp.1336-1357, Aug. 2002.
9. *GIF89a Specifications*, CompuServe Inc., July 1990.
10. G. Roelofs, *PNG: The Definitive Guide*, O'Reilly, 1999.
11. J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Information Theory*, vol. 23, no. 3, pp. 337-343, May 1977.
12. J. Li and H. Sun, "A virtual media (Vmedia) access protocol and its application in interactive image browsing," *SPIE, IS&T and ACM SIG Multimedia, Multimedia Computing and Networking 2001 (MMCN'01)*, vol. 4312, no. 10, San Jose, CA, Jan. 2001.
13. A. Antonacopoulos and D. Karatzas, "Text extraction from web images based on human perception and fuzzy inference," *Proc. 1st International Workshop on Web Document analysis*, Seattle, WA, Sept. 2001.
14. R. Lienhart and A. Wernicke, "Localizing and segmenting text in images and videos," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 12, no. 4, Apr. 2002.
15. Y. Zhong, K. Karu, and A. K. Jain, "Locating text in complex color images," *Pattern Recognition*, vol. 28, no. 10, pp. 1523-1536, Oct. 1995.
16. A. K. Jain and B. Yu, "Automatic text location in images and video frames," *Pattern Recognition*, vol. 31, no. 12, pp. 2055-2076, 1998.
17. V. Wu, R. Manmatha, and E. M. Riseman, "TextFinder: an automatic system to detect and recognize text in images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, pp. 1224-1229, 1999.
18. C. Strouthopoulos, N. Papamarkos, and A. E. Atsalakis, "Text extraction in complex color documents," *Pattern Recognition*, 35, pp. 1743-1758, 2002.
19. N. Papamarkos, A. E. Atsalakis, and C. Strouthopoulos, "Adaptive color reduction," *IEEE Trans. Systems, Man, and Cybernetics–Part B: Cybernetics*, vol. 32, no. 1, pp. 44-56, Feb. 2002.
20. D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Processing*, vol. 9, Issue 7, pp. 1158-1170, July 2000.
21. A. Ortega and K. Ramchandran, "Rate-distortion techniques in image and video compression," *IEEE Signal Processing Magazine*, pp. 23-50, Vol. 15, No. 6, Nov. 1998.
22. P. Deutsch, "DEFLATE compressed date format specification," RFC1951, May 1996. Available as ftp://ftp.uu.net/pub/archiving/zip/doc.