Proceedings of USITS' 99: The 2nd USENIX Symposium on Internet Technologies & Systems

Boulder, Colorado, USA, October 11–14, 1999

JPEG COMPRESSION METRIC AS A QUALITY AWARE IMAGE TRANSCODING

Surendar Chandra and Carla Schlatter Ellis

JPEG Compression Metric as a Quality Aware Image Transcoding

Surendar Chandra and Carla Schlatter Ellis
Department of Computer Science, Duke University, Durham, NC 27708
{surendar,carla}@cs.duke.edu

Abstract

Transcoding is becoming a preferred technique to tailor multimedia objects for delivery across variable network bandwidth and for storage and display on the destination device. This paper presents techniques to quantify the quality-versus-size tradeoff characteristics for transcoding JPEG images. We analyze the characteristics of images available in typical Web sites and explore how we can perform informed transcoding using the JPEG compression metric. We present the effects of this transcoding on the image storage size and image information quality. We also present ways of predicting the computational cost as well as potential space benefits achieved by the transcoding. These results are useful in any system that uses transcoding to reduce access latencies, increase effective storage space as well as reduce access costs.

1 Introduction

The advent of inexpensive hardware such as powerful personal computers, digital cameras, scanners and other easy-to-use technologies is making it easier for the average user to dabble in multimedia. The phenomenal growth of Internet technologies such as the Web and electronic mail allows users to disseminate and share these multimedia objects. By some estimates [19], about 77% of the data bytes accessed in the web are from multimedia objects such as images, audio and video clips. Of these, 67% of the data are transferred for images. While this ability of users to share multimedia objects makes the Internet more valuable to consumers, the underlying capabilities of the system are not always able to keep up with this shifting usage.

Users access these multimedia objects from a wide variety of devices with different resource constraints. Users are not only accessing the rich multimedia objects from traditional desktops that are better able to render these objects, but they are also using mobile devices such as a palmtops and laptops as well as newer devices such as webtops and navigation systems that are resource constrained in terms of the processing, storage and display capabilities. The network capabilities used in accessing these multimedia objects vary widely from wired networks such as high speed LANs, ISDN, DSL, cable and telephone modems, as well as wireless technologies such as cellular, CDPD, Ricochet and GSM networks. Depending on the technology used, the networks can be slow, unreliable and expensive.

In such an environment of varying network, storage and display capabilities, one size does not fit all. Consumers using expensive networks want to download multimedia images for the lowest possible cost. Consumers using high speed networks and high quality displays want to view the multimedia images at the highest quality.

In such an operating environment, transcoding can be used to serve the same multimedia object at different quality levels to the different users. Transcoding is a transformation that converts a multimedia object from one form to another, frequently trading off object fidelity for size. By their very nature, multimedia objects are amenable to soft access where the user is willing to compromise object fidelity for faster access. From the server's perspective, transcoding may be employed in an attempt to achieve greater scalability, as exemplified by AOL's across-the-board transcoding of all requested JPEG and GIF images [1].

Transcoding of multimedia objects can be performed in any of the following scenarios:

static environment where the information provider transcodes the object to a variety of formats (e.g., thumbnails along with full scale images) so that the consumer can download the appropriate form for the current operating environment.

streamed environment where the network infrastructure (e.g., a network proxy) can transcode the objects on the fly to compensate for network latencies.

Previous work by [19, 8, 11] has used transcoding in this fashion.

store and forward environment where the system can transcode the objects to a smaller size to increase the effective size of the local image storage space. Users of a digital camera may transcode a previous picture to a lower quality version to create space for a new picture.

For transcoding to be useful, we need to quantify the loss in information so that an informed decision can be made on choosing the aggressiveness of transcoding. We need to understand the computational costs and storage benefits in order to decide if a particular transcoding is worth the effort. For example, without such understanding, systems that have used JPEG Quality Factor as a transcoding metric typically transcoded the images to a drastically lower JPEG Quality Factor.

Our work addresses the problem of how to characterize the quality-size tradeoffs involved in transcoding JPEG images so that the effectiveness of a particular level of transcoding can be predicted efficiently. This implies several subproblems.

The first problem is to more precisely define what we mean by an "effective transcoding". Thus we introduce the notion of "quality aware transcoding" that is meant to capture a direct and measurable relationship between reduction in perceived image quality and savings in object size. In order to measure whether we achieve that goal for a particular image and a particular transformation, we need to be able to quantify the reduction in quality caused by the operation. This requires that we have a metric that corresponds, in some sense, to the user's perception of quality and that we can consistently measure the starting and ending quality levels in order to determine the loss. We analyze the use of the JPEG compression metric for that purpose. The size reduction component is straight-forward.

Next we must determine the computational cost required to perform the transcoding. Thus, we ask whether the transcoding is easy to compute and whether it can be streamed.

Finally, it is highly desirable to easily predict the outcome of a possible transcoding operation in terms of the size reductions, quality lost, and computation cost to determine if it is worth the effort for a particular case. Since transcoding may not provide space benefits for all images, we need to analyze if we can predict whether a particular transcoding will provide space benefits for the particular image. Thus we explore prediction algorithms that can estimate those outcomes. For our study, we analyze a number of JPEG images available from a number of Internet Web sites. This establishes how our methods apply to typical workloads (of JPEG images). Similar work has to be undertaken for other image, audio and video multimedia types.

Step	**Compression**	Step	**Decompression**
1	Convert ColorSpace	5	Convert Colorspace
2	Downsample	4	Upsample
3	Forward DCT	3	Inverse DCT
4	Quantize	2	De-Quantize
5	Entropy Encode	1	Entropy Decode

Table 1: JPEG Compression and Decompression

In this paper, we show that, for a transcoding that changes the JPEG compression metric (referred to as the JPEG Quality Factor), the change in JPEG Quality Factor directly corresponds to the information quality lost. To measure this change, we describe an algorithm to compute the Independent JPEG Group's (IJG) [15] equivalent of the JPEG Quality Factor of any JPEG image. To understand the overhead involved in performing a transcoding, we develop a predictor that predicts the computational overhead with a high degree of accuracy. We also develop a predictor for predicting if an image will transcode efficiently. We define transcoding efficiency of an transcoding algorithm by the ability to lose more in storage space for a particular loss in information quality. We show that we can predict efficient images at a significantly better rate than the base case. We validate these results with a number of JPEG images.

The results from this paper were utilized in a companion paper [4] that describes the utility of quality aware transcoding for serving multimedia objects to mobile clients.

The remainder of this paper is organized as follows: Section 2 describes the JPEG compression metric and how it can be computed for a given JFIF image. Section 3 describes the workload used for evaluating our transcoding technique. Results of transcoding the images for this workload are presented in Section 4. Section 5 discusses related work in the area of transcoding and image quality metrics and Section 6 summarizes the work.

2 JPEG Compression Metric

JPEG [20] is the Joint Photographic Experts Group lossy compression scheme for still images. JFIF [10] is the

JPEG File Interchange Format used for exchanging image files compressed using JPEG compression. JPEG compression is based on psycho-visual studies of human perception. To convert to a smaller file size, this type of compression drops the least-noticeable picture information. Human visual system response is dependent on the spatial frequency. JPEG is a lossy image compression algorithm that uses Discrete Cosine Transform (DCT). DCT provides a good approximation to allow one to decompose an image into a set of waveforms, each with a particular spatial frequency. This allows us to successively drop frequency components that are imperceptible to the human eye.

The different steps in compressing and decompressing an image using JPEG are outlined in Table 1. To compress an image, the color space of the image is first transformed to the YCbCr [2] color space, followed by any smoothing operations, followed by Minimum Code Unit (MCU) assembly and forward DCT computation, followed by quantization and entropy encoding. Since the human eye is less sensitive to chrominance values than to luminance values, different color components can be downsampled differently: Cb and Cr components are usually downsampled by a 2x1 factor, while the Y component is scaled using a 1x1 factor. Decompression process essentially reverses these steps: entropy decoding is performed, followed by dequantization and inverse DCT, followed by upscaling and color space conversion to get back to an RGB image.

In JPEG, the compression ratio of the output image is controlled solely by the quantization tables used in step 4. An all-1's quantizer achieves a compression ratio that is comparable to a lossless compression algorithm. The JPEG specification [3] section K.1, provides standard luminance and chrominance quantization tables that provide good results for a variety of images. The standard quantization tables can be scaled to vary the compression ratios. Most JPEG compressors allow the user to specify a range of values for the scaling factor, by specifying a compression metric called Quality Factor. This Quality Factor is an artifact of JPEG compression. Different software implementations use different values for Quality Factor. Quality Factors are not standardized across JPEG implementations. The IJG Library [15] uses a 0-100 scale, while Apple formerly used a scale running from 0 to 4. Recent Apple software uses an 0-100 scale that is different from the IJG scale. Paint Shop Pro's scale uses a 100-0 scale, where lower numbers imply higher quality. Adobe Photoshop gives discrete *maximum*/ *high*/ *medium*/ *low* choices.

The JPEG Quality Factor can form the basis for a transcoding that progressively reduces the Quality Factor of an image to achieve better compression ratios. A transcoding that uses JPEG Quality Factor, can transcode an image, either to an absolute Quality Factor value or to a percentage of the original quality.

2.1 Initial Quality Factor

In order to determine if the JPEG compression metric forms an effective transcoding, we need the ability to measure the JPEG Quality Factor of an image. Without the ability to measure the initial Quality Factor used to produce an image, the transcoding algorithm might transcode an image with a low initial Quality Factor to an apparently higher Quality Factor value by manipulating the quantization tables, even though such an operation does not increase the information quality of the output transcoded image. The resulting output image from such an operation is bigger than the original "lower quality" image. For example, transcoding a JPEG image of Quality Factor 20 and size 37 KB to a JPEG image of Quality Factor 50 produces an image of size 42 KB. Systems that have used JPEG compression metric as a transcoding metric, such as [8, 11], have avoided this problem of not knowing the initial JPEG Quality Factor by transcoding the images to a sufficiently low quality value, such as Q=5, so that they can transcode all images to a smaller size.

The quantization table that was used to compress an image is stored in the JFIF header, but the JPEG Quality Factor that was used to generate the quantization table is not stored along with the image and hence the original JPEG Quality Factor is lost. Different JPEG compressors use different quantization tables. Since there is no reliable way to recognize the software that produced a given JPEG image, we develop a predictor to measure the IJG equivalent of the JPEG Quality Factor of an image.

By default, the IJG implementation uses the standard quantization tables, computes the Scaling Factor(S) from the Quality Factor(Q), and then uses the Scaling Factor to scale the standard quantization tables (*stdQuantTbl*). Finally it computes the quantization tables for the image (*imgQuantTbl*) as follows

$$S = (Q \geq 50)?(\tfrac{5000}{Q}) : (200 - 2Q)$$
$$imgQuantTbl[i] = \tfrac{stdQuantTbl[i]*S+50}{100}$$

We reverse the steps used by IJG in computing the quantization tables to compute the quality value as follows:

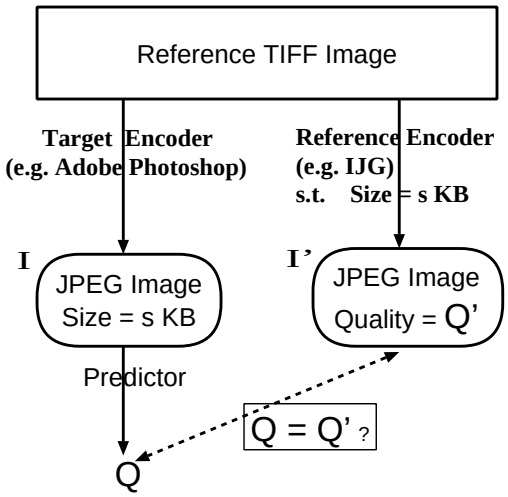


Figure 1: Evaluating JPEG Predictor

$$S' = \frac{imgQuantTbl[i] \star 100 - 50}{stdQuantTbl[i]}$$
$$Q' = (S' \leq 100)?(\frac{200 - S'}{2}) : (\frac{5000}{S'})$$

The JPEG Quality Factor predictor function involves integer computations on the quantization tables that introduce integer rounding errors. Hence, even for images produced by IJG's software, the Quality Factors computed can be off by a point or two.

To validate the correctness of our JPEG Quality predictor, we first tested the predictor on images produced by IJG software. Using IJG, we created a JPEG image with a specified JPEG Quality Factor, and verified if our predictor could predict the correct Quality Factor that was used. As expected, the results were accurate within a few points; which is attributable to integer round-off errors.

Next, we tested the effectiveness in predicting the JPEG Quality of images produced by Adobe Photoshop and Paintshop Pro. We evaluated the effectiveness of our predictor in two steps. The steps are illustrated in Figure 1. First we produced a JPEG image (I) from a reference TIFF image using the target encoder (e.g. Photoshop). We ran our predictor on the resulting JPEG image to estimate its IJG-equivalent Quality Factor (Q). Then, in step two, we produced another JPEG version of the reference TIFF image (I') but used the IJG software using a Quality Factor value such that the resulting image matches the size of the JPEG image produced in Step one. Finally we compared the predicted Quality Factor and the value used in producing the IJG JPEG version. The values of Q and Q' should be identical for a successful predictor.

First we tested the effectiveness of our compressor for images produced using Adobe Photoshop. We compressed a reference TIFF image using the four quality settings allowed by Photoshop. Our predictor estimated the qualities for the four different Adobe settings to be 37, 62, 82 and 92 respectively. The sizes of the files derived from Adobe's software's four compression settings were similar to IJG compressed images of Quality Factors 55, 74, 89 and 97 respectively.

Next, we tested the effectiveness of our predictor for images produced using Paint Shop Pro. We compressed a reference TIFF image using Paint Shop for a range of Quality values (Q') between 10 through 90. Our predictor estimated the Quality Factor values at a value equal to (100 - Q'). The file sizes also corresponded to JPEG Quality values of (100 - Q'). In fact, the file size values matched so perfectly, we suspect that Paint Shop Pro internally uses IJG software, but chooses to invert the meaning of IJG Quality Factor values.

Hence, our JPEG Quality Factor predictor closely predicts the JPEG Quality Factor for images compressed using IJG and Paint Shop Pro and underestimates the JPEG Quality Factors for images compressed using Adobe Photoshop. For our purposes, it is preferable to under predict the image quality. Otherwise, applications might try to transcode an image to a Quality Factor that is lower than an over-estimated Quality Factor, but is in fact higher than the correct Quality Factor of the image (which would produce a JPEG image that is bigger than the original un-transcoded image). Since we do not know the *real* Quality Factors, and since the values are either under predicted or predicted accurately, the predictors' performance is acceptable as a predictor of the IJG equivalent of Quality Factor of the given JPEG image.

2.2 Perceived Information Quality

When we transcode an image, we want to quantify the loss in information as well as the space savings achieved. To quantify the loss in information we need to measure the image quality, as *perceived by an observer*. Then, we need to address whether JPEG Quality Factor captures perceived quality adequately.

The perceived information quality of an image depends on a variety of factors such as the viewing distance, ambient light, the display characteristics, the image size, the image background etc. Images can be objectively measured using *distortion metrics*, which are functions that find the difference between two images, *fidelity metrics*, which are functions (or series of functions) that describe the *visible* difference between two images and

subjective measures such as *quality metrics* that are numbers, derived from physical measurements, which relate to perceptions of image quality.

Distortion metrics, such as tone, color, resolution, sharpness and noise, describe a measurable change in an image, although this change need not necessarily be visible. Distortion metrics do not take into account the characteristics of either the output device or observer, although some attempts have been made to include these factors.

Fidelity metrics, such as the Visual Differences Predictor (VDP), produce difference images that map perceptible differences between two test images. While this approach is useful for diagnostic tasks, a single value representing 'quality' is often more useful. Ahumada and Null [13] note that the relationship between fidelity and quality varies with the observer's expectations of the intended use for the image, a feature which is also dependent on factors such as past experience. The subjective assessment of quality also varies with the pictorial content of the test stimuli, even when image fidelity remains constant.

In his Ph.D. thesis, Adrian Ford [7], tried a series of image quality and color reproduction measures to quantify the effects of lossy image compression on baseline JPEG images using the IJG software. Objective measurements such as tone reproduction, color reproduction (CIE $\Delta E^*$and Color Reproduction Index), resolution, sharpness (MTF) and noise (Noise Power Spectra) along with quality metrics, such as 'Barten's Square Root Integral with Noise' and 'Töpfer and Jacobson's Perceived Information Capacity' were used to evaluate the overall image quality. The image quality metrics were implemented with the aid of a published model for the human eye and a model of the display system based on experimental measurements. Measured image quality was compared with subjective quality assessments performed under controlled conditions. He concluded that JPEG compression metrics such as JPEG Quality Factor outperformed other measures in predicting the quality of an image.

Since the workload and assumptions of Adrian Ford's thesis closely matches the kinds of images available on the internet, we adopt his conclusion that JPEG Quality Factor is a good representation of the subjective image quality. Reduction in JPEG Quality Factor directly translates to loss of image information quality.

3 Workload

The effectiveness of our study depends on the realism of the JPEG images that are used to validate the results. To better understand the qualities of the JPEG images available on the web, we classify web sites into four distinct categories with respect to the site's usage of JPEG images. These categories are

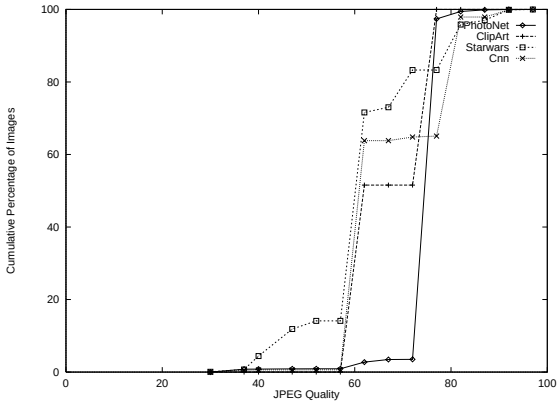
News Site. News sites use images to reinforce some news story. The images are secondary to the news being delivered. Hence we expect the images to be small and of low quality. For our experiments, we used 6217 JPEG images downloaded from Cnn.com [5].

Image Site. The sole purpose of online art gallery sites is to deliver high quality images. For our experiments, we used 4650 JPEG images downloaded from Photo.net [9].

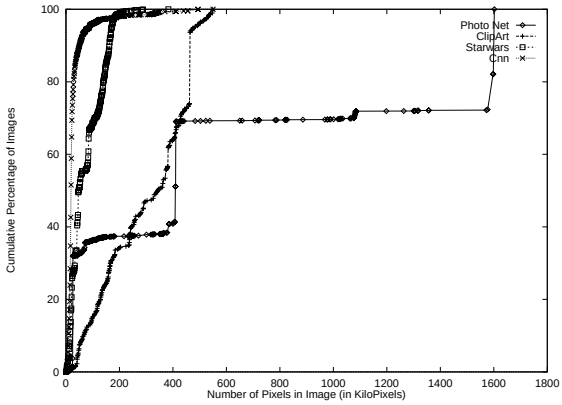
Commerce Site. The primary purpose of commerce sites is to sell their wares. These sites would like to deliver big, high quality images to promote their merchandise without turning away users with high access latencies. For our experiments, we used 1248 JPEG images downloaded from Starwars.com [22].

ClipArt Site. JPEG encoding is optimized for realistic full color images and is not especially appropriate for compressing images with few colors such as cartoons and line drawings. Still, such ill-advised uses do exist in the Web. We use 485 JPEG images from a image collection from Media Graphics [17], which provides clipart collections that are intended to be used by a Web site designer.

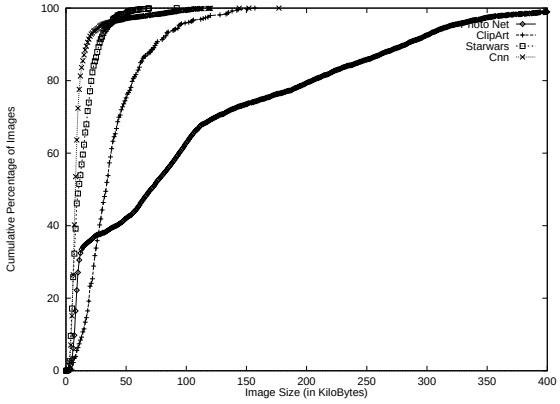
We plot the cumulative distribution of JPEG Quality, the image size and image geometry in Figure 2. From Figure 2(a), we note that 60% of images in the CNN workload have Quality Factor less than 60, while 80% of images in Starwars workload have Quality Factor less than 60. Most of the images in PhotoNet have Quality Factor of 77. From Figure 2(b), we note that most of the images in CNN and Starwars workload have small image geometries. Correspondingly, from 2(c), we note that images in the CNN and Starwars occupy smaller file sizes, with most of the images being less than 20KB. PhotoNet on the other hand has a number of very big images that are greater than 100KB. ClipArts consists of fairly big images with 25% of the images being over 50KB.



(a) Image JPEG Quality



(b) Number of Pixels in image



(c) Image file size distribution

Figure 2: Workload Characteristics

Workload	Reduced Image Quality		
	25%Q	**50%Q**	**75%**Q
Overall	3.2%	47.4%	66.1%
CNN	2.9%	40.2%	99.6%
PhotoNet	1.8%	66.3%	25.1%
ClipArt	2.3%	22.7%	53.6%
Starwars	6.8%	18.9%	56.9%

Table 2: % Efficient images for a given loss in image quality

4 Results

4.1 Efficient, Quality Aware Transcoding

Once we quantify the loss in information, we can measure if the transcoding produces significant saving in space for a corresponding loss in image information quality.

We define transcoding efficiency of a transcoding operation by the ability to lose more in size for a particular loss in information quality. For example, if an image was transcoded to lose 50% of information quality, the output image size should be less than 50% of the original image for it to be an efficient transcoding. For some applications, this restriction may be too restrictive and these applications may relax the constraint to accept output size of, say, 55% instead of 50%.

From Section 2.1, we note that our initial JPEG Quality Factor predictor under-estimates the Quality Factor of images compressed using Adobe Photoshop. As a consequence, the quality loss may be similarly underestimated which has the effect of allowing smaller size reductions to be classified as efficient. This errs on the side of capturing *all* efficient transcodings at the cost of including some borderline inefficient ones.

In order to measure the overall efficiency of the transcoding algorithm, we performed experiments on all the images in our workloads. We measured the original JPEG Quality Factor of the images using the algorithm described earlier, reduced the image qualities to 75%, 50% and 25% of the original Quality Factor values, using fixed Huffman tables[1], and then measured the resulting image sizes. The change in image size, from the original image size, for a given reduction in image qual-

[1] It has been observed that using custom Huffman tables may yield better compression. For the images in our workload, we empirically measured that, on average, using custom tables produces images that are 11% smaller than using fixed tables. However, this comes at a cost of consuming 32% more memory and taking 56% more time to transcode. Our use of fixed tables means even more savings are possible.

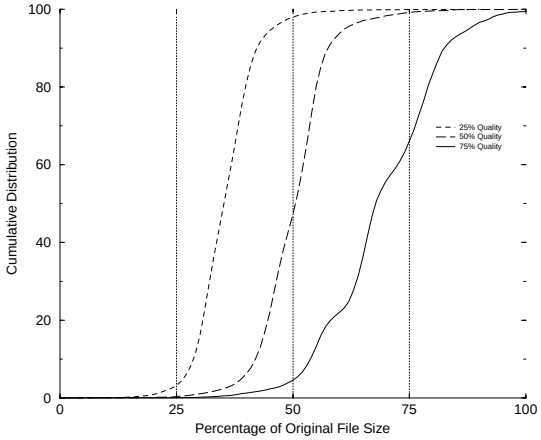


Figure 3: Cumulative Distribution of transcoding efficiency for all the images in the workloads

ity, is plotted as an cumulative distribution in Figure 3. For a given reduction in quality, any change in file size that is less than the change in quality (values to the left of the vertical line representing the loss in quality) is an efficient transcoding as the resulting file size is smaller than the corresponding loss in quality. Any value to the right is inefficient as the resulting file size is bigger than the corresponding loss in quality.

From Figure 3, we can see that reducing the initial quality to 75%, 50% and 25% was efficient for 66.1%, 47.4% and 3.2% of the images respectively. The results for the different workloads, were tabulated in Table 2.

From Table 2, we can see that images reduce efficiently for smaller drops in quality; a big drop in image quality to 25% of original quality does not lead to an efficient transcoding for most of the images. For images from PhotoNet, only 25% of the images reduced efficiently for a Quality Factor reduction to 75%, while 66% of the images reduced efficiently for a Quality Factor reduction to 50%.

In exploring this behavior, we discovered that images in the PhotoNet collection were initially entropy encoded using custom Huffman tables. Hence, for a small drop in image quality, the relative space gain from transcoding was offset by the increase in image size from using (less optimal) standard Huffman tables. For a more significant loss in the Quality Factor, the space gain from transcoding offsets this increase in image size.

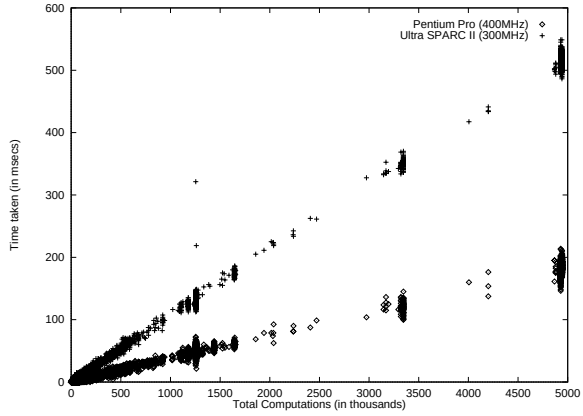


Figure 4: Computational Cost to re-quantize the JPEG images

4.2 Computation Cost

The next measure of the effectiveness of a transcoding is an analysis of the computational cost involved in performing a transcoding operation so that an intelligent decision can be made about the applicability of the transcoding for a particular operating environment.

The JPEG Quality Factor value of an image can be reduced without completely uncompressing the image (Table 1), by first performing an entropy decoding, followed by dequantization and requantization followed by entropy encoding. The computational cost does not depend on the quality of the image being transcoded, nor on the quality value being used for transcoding the image. Quantization and dequantization involve integer multiplication and division operation on each DCT image block, which can be collapsed into a single integer multiplication on each DCT image block. The basic computational block is of the form

$$dst[i] = (src[i] * reQuantizedScale) \gg scale\,(0 < i < 63)$$

For an image with i color components, and a size of $n_i \times m_i$ blocks per color component, we need to perform $\sum_i n_i * m_i * 64$ computations. For example, transcoding a 480×480 full color image (with 3 color components YCbCr) with no component scaling involves $60*60*3*64$ or 691200 computations. For a 480×480 image with 2×1 scaling for chrominance values, this corresponds to $(60 * 60 + 60 * 30 + 60 * 30) * 64$ or 460800 computations. Transcoding a progressive JPEG image requires expensive Huffman encoding for re-compression as the default tables are not valid for progressive formats. Hence, for our experiments, we automatically transcode progressive JPEG images to the simpler sequential format.

The image component dimensions are available in the JFIF headers and hence the number of basic computational blocks required to transcode an image can be computed for the particular image. Since the time taken to compute the basic computational block can be statically determined for a particular computing device, we want to validate that we can use the number of basic computations as a measure of the computational cost for a particular transcoding.

For our experiments, we used the IJG software, which is not particularly optimized for the compression operation. We used a Pentium II 400MHz machine with 128MB memory and an Ultra Sparc II 300MHz machine with 512MB of memory, both running Solaris 2.6 for our experiments. Based on the basic computational blocks described above, the time taken to transcode the images from our workloads are plotted in Figure 4. We observe that the number of basic computation blocks and the time to transcode an image are linearly correlated. The results indicate that the observed data fits the linear equation $time = const * basicBlks$ (where *const* $\simeq 37$, for Pentium II and *const* $\simeq 105$, for Ultra Sparc II). The value $time$ is measured in msecs and $basicBlks$ are measured in millions of basic computational blocks. The linear correlation coefficient (ρ) between the computational blocks and the time taken to transcode was 0.99 for both the machines. The 95% confidence interval, computed using standard deviation about regression, was measured at 15.3 msecs for the Ultra Sparc II and 7.9 msecs for the Pentium Pro II. These confidence intervals are quite adequate for most purposes.

Hence we conclude that the time required to perform a image transcoding using the JPEG Quality metric can be predicted accurately by computing the number of basic computational blocks using the image components dimensions. The time taken to compute a basic computational block, which affects the coefficients of the linear equation described above, can be computed statically for a particular computing device.

4.3 Predictable Size Tradeoff

Another objective in using JPEG Quality Factor as a transcoding algorithm is the ability to predict the output size of an image for a particular transcoding.

Previous work by Han et al. [11] attempted to predict the image size using the image pixel counts as a metric. They transcoded sample images to JPEG Quality Factor values of 5 and 50. For these transcodings, they measured the linear correlation coefficient between the

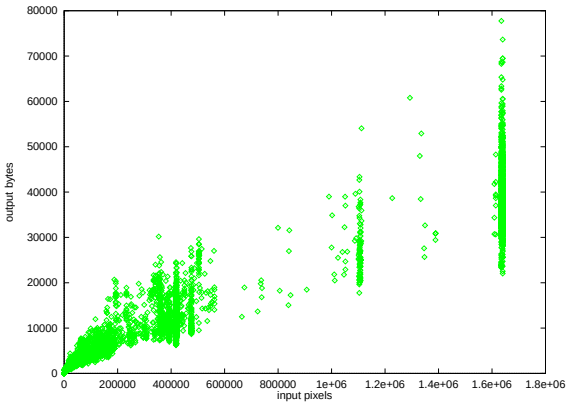


Figure 5: Correlation between output size and input pixels (Q=5)

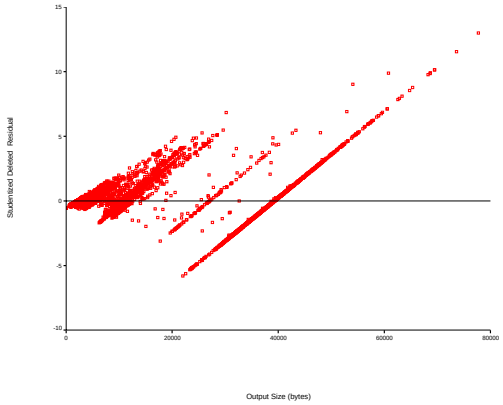


Figure 6: Studentized Residuals with i^{th}observation deleted (Q=5)

transcoded output JPEG image size and the number of pixels in the image at 0.94 and 0.92 respectively. For the images in our data set, we plotted (Figure 5) a scatter plot of the input pixels vs. the output image size. The data exhibits heteroscedasticity [14] of the variance, wherein the variance of the output size is not the same for all values of input pixels. The variance is higher for images with higher input pixels. The studentized residual for transcoding images to a Quality Factor value of 5, with the i^{th}observation deleted, were plotted as a scatter plot in Figure 6. Similar results were observed for transcoding images to Quality Factor value of 50. In order to make inferences from the data sample using linear statistical models, the plot should contain a horizontal band of points with no hint of any systematic trend. The residual values for our data, as seen in Figure 6, suggests that the output size depends on multiple (unknown) parameters and not just on the input pixels. Hence the input pixels cannot be used as a useful predictor of the output image size for our set of images.

Intuitively, this makes sense as the type of input images can vary from photographic images to clip arts to images

of random noise. Since JPEG encoding is optimized for realistic images, random noise will not be encoded as efficiently as a photographic image. There is no easy heuristic to identify if an image is a photographic image. It is unlikely that the exact output size of an image for a particular transcoding would be predictable a priori without an understanding of what the image represents.

Even though the ability to predict the exact output image size would be valuable, for a transcoding to be useful, it is usually sufficient to analyze if an image will transcode efficiently. Hence, we try heuristics to predict if an image will transcode efficiently.

To better understand what factors might affect the transcoding efficiency, we need to understand the characteristics of the JPEG entropy coding algorithm. Since re-quantization only involves re-encoding of entropy parameters, DCT operations do not play a part in affecting the efficiency of a transcoding that lowers the JPEG Quality Factor.

JPEG baseline algorithm computes the entropy encoding using Huffman encoding with a standard Huffman table. The JPEG quantization step strives to increase the number of zeros in the image, starting with the least perceptible high frequency components. The default Huffman table stores zero value using 2 bits. Since typical images that are to be entropy encoded tend to be predominantly zero, the JPEG algorithm employs three additional optimization steps to improve the compression ratios.

1. The DC components are stored as a difference value from the DC coefficient from the previous image block, rather than by a straight Huffman code.
2. Sequences of zeros are encoded by a single byte which contains the count of zeros in the upper four bits with the lower four bits being zeros. For example, a sequence of 16 zeros are stored using 8 bits, instead of 32 bits required by pure Huffman encoding.
3. If the trailing entries in a image block are zeroes, the JPEG algorithm stores an End of Block (EOB) marker and avoids encoding those trailing zeros explicitly. Hence, if an image block had 60 trailing zeros, they will be encoded using 0 bits, instead of 120 bits that would be required by a pure Huffman encoding.

Empirically, we found that we only need to analyze the luminance values of an image to predict its transcoding efficiency. This makes sense, since human eyes are less sensitive to chrominance values; the default quantization tables reduce the chrominance values to zeros faster. Also, empirically, we found that images that typically transcode efficiently tend to have higher coefficients for the low frequency components. Since the high frequency components are reduced quicker by the quantization tables, their contribution does not affect the final image size. We also noticed that images of higher initial quality tended to transcode efficiently.

Prediction Criteria	% for Criteria	75% Q	50% Q	25% Q	Q= 75	Q= 50	Q= 25
None	*100*	*67*	*46*	*2*	*89*	*81*	*41*
$P = 0$	21	57	37	1	89	78	44
$P > 1$	29	85	58	6	86	81	30
$P > 3$	15	97	82	8	87	81	27
$P > 5$	9	99	83	12	89	84	30
$Q > 80$	20	98	83	9	87	86	30
$R < 10\%$	54	68	51	4	89	81	40
(Q > 80) or (P > 3)	21	96	82	9	87	85	30

Table 3: % of Accurately Predicted Images (C%)

Prediction Criteria	% for Criteria	75% Q	50% Q	25% Q	Q= 75	Q= 50	Q= 25
None	*100*	*1*	*6*	*82*	*8*	*9*	*38*
$P = 0$	21	0	2	18	2	2	8
$P > 1$	29	0	2	23	3	3	13
$P > 3$	15	0	0	12	2	2	6
$P > 5$	9	0	0	7	1	1	4
$Q > 80$	20	0	0	14	2	2	8
$R < 10\%$	54	0	2	43	5	5	20
((Q > 80) or (P > 3)	21	0	0	14	2	2	8

Table 4: % of Egregiously Mispredicted Images (E%)

Taking these observations about JPEG images as well as the Huffman optimizations into account, we hypothesize that an image will transcode efficiently, if:

There are sufficiently many high magnitude low frequency components in the luminance blocks of the images. The alternative, which are images whose magnitude of low frequency components are low, will already have lost much of the quality represented in the low frequency components that are most noticeable to the human eye and hence further compression is inefficient.

We define low frequency factor (P) as the percentage of low frequency components in the luminance blocks of an image that have coefficients whose absolute value is over a certain threshold. The number of frequency components to analyze is a compromise between choosing predominantly low frequency component images such as cartoons and predominantly high frequency component images such as detailed photographs. We experimentally varied the number of low frequency components to consider at 3, 5 and 7 (odd number of components

are required to maintain equal weighting for horizontal and vertical frequency components) and the magnitude thresholds to 64, 80, 96 and 128. We found that the values of 80 for the threshold and 5 for the number of components reasonably captured our intuition. Hence, for our experiments, we computed P as the percentage of frequency components 1 through 5 (which translates to 1, 8, 16, 9 and 2 in the natural order) over all luminance blocks that have coefficients with an absolute value greater than 80.

There are sufficiently few luminance color blocks within the image in which all of the highest frequency components beyond some point have coefficients that are all zero. Such images still present adequate opportunity for further efficient compression.The alternative, which is an image with many such blocks having zero coefficients for all of the high frequency components, will already be efficiently compressed using the JPEG Huffman optimizations described earlier.

We define a high frequency factor metric (R) as the percentage of luminance blocks within the image which have zero coefficients for *all* of the highest frequency components beyond a certain threshold. The choice of the threshold is a compromise. We tried zero-coefficient runs of the 48, 63, and 58 highest frequency components on test images. For our study, we chose 58 ($max - P\ or\ 63 - 5$) as the threshold, in order to avoid counting coefficients already captured by the low frequency factor (P), described above. Hence, for our experiments, we computed R as the percentage of luminance image blocks which have zero coefficients for all of the 58 (or more) highest frequency components.

The image is of high initial quality. We consider images produced with a JPEG Quality Factor value (Q) over 80 as high quality.

The parameters P and R can be computed easily by traversing a JPEG image's luminance DCT component blocks once. P can be computed by accessing the first few components in each DCT component block. Usually, these parameters can be computed as a by-product of transcoding an image locally or while measuring the initial JPEG Quality Factor of an image.

Using these parameters, we categorized the images and measured the number of images that transcoded efficiently in each category. For a particular criteria to be useful in predicting if an image will transcode efficiently, it needs to:

Apply to a significant percentage of images. For example, it is undesirable to have a prediction algorithm that makes predictions for only about 2% of the images. For our study, we analyze prediction criteria that at least predict 10% of images in a workload.

Accurately predict at a higher percentage than the general population. The percentage of images that are predicted efficient should be better than the percentage of efficient images in the original workload. For example, if 25% of all the images transcoded efficiently, we prefer a prediction that categorizes the images such that 80% of the predicted images transcode efficiently over a criterion that categorizes 30% of the efficient images.

Mis-predict egregiously inefficient images at a lower percentage than the general population. We define a pathological image as an image that is bigger than 125% in size than the loss in information quality. For example, if an image loses 50% Quality, we categorize an image that is bigger than 125% of 50%, i.e. 62.5%, as pathological.

We first transcoded the images in our workload to Quality Factor values of 75%, 50% and 25% of the original Quality Factor values as well absolute Quality Factor values of 75, 50 and 25. Using various predictor criteria values, we separated the images from our workload into images that were predicted as efficient images by the particular criteria. For the images that were predicted efficient, we measured the percentage of images that actually transcoded efficiently as well as the percentage of pathological images that were mispredicted as efficient. The goal was to increase the percentage of good predictions. The percentage of correct predictions (C%) and the percentage of egregious mispredictions (E%) were tabulated in Table 3 and 4 respectively. The results were tabulated as follows:

The first column in the table shows the criteria that were used to separate the images. We call the criterion *None*, which includes all the images in the workload as the base case.

The second column specifies the percentage of images that satisfy the criteria specified in the first column. For our study, values less than 10% were ignored because they categorize an insignificantly small percentage of images to be useful as a prediction criterion.

The subsequent columns show the results for transcoding the images to 75%, 50% and 25% of the original JPEG Quality Factor as well as absolute JPEG Quality Factor values of 75, 50 and 25.

The goal is to choose results that produce significantly higher C% and lower E% than the base case. The results for all the images in our workload are shown in Table 3 and 4 respectively. Similar trends were observed for the individual workloads.

From Tables 3 and 4, we note that $P > 5\%$ predicts less than 10% of the images (9%) and hence is ignored from further consideration. For transcoding to Quality Factor values of 75%, 50% and 25% of the original JPEG Quality Factor, the criterion $(Q > 80) \,||\, (P > 3\%)$ predicts efficient images correctly and mispredicts images at a rate better than the base case. For transcoding to JPEG Quality Factor values of 75, 50 and 25, none of the criteria predicts at a significant rate, even though they mispredict images at a better rate than the base case. The value of R is not a useful predictor of the efficacy of an image for transcoding using the JPEG compression metric.

We cannot make conclusions regarding images that weren't predicted as efficient. We tried parameters such as number of colors, pixel density etc, but could not find a reliable way of predicting the efficiency of images that have $P < 3$ and $Q < 80$. We repeated the experiments by varying the definition of P to include images frequency components over absolute values of 64, 96 and 128 as well as changing the number of frequency components to 3 and 7. The end results were similar and the values that we chose better predict efficient transcodings by a slight margin.

5 Related Work

Fox et al.[8] used transcoding to render an image on a PDA such as Palmpilot, as well as to offset access latencies from slow modems. Noble et al. [18] manipulated the JPEG Compression metric as a distillation technique for a web browser that adapts to changing network environments. Mazer et al. [16] describe a framework for allowing users to specify their own transcoding transducers for a application-specific proxy that acts on the HTTP stream to customize the stream for a particular client. Ortega et al. [19] have used JPEG progressive encoding to recode images to lower resolutions, thereby increasing the effective cache size.

Commercial products such as WebExpress [6] from IBM, QuickWeb technology [12] from Intel, Fastlane [21] from Spectrum Information technology and Johnson Grace ART format [1] from AOL have used various forms of compression and transcoding operations to improve web access from slow networks.

Even though transcoding has been widely used in a number of systems to deal with network access latencies, display characteristics or storage space requirements, there has been little formal work in conducting a systematic study to measure the information loss associated with a given transcoding. As previously mentioned, Han et al.[11] attempted a similar effort to characterize image transcoding delay as well as the transcoded image size.

The concept of information quality and measuring the objective and subjective quality of an image has been well researched and understood. The ability to quantify the information loss in rendering an image on a particular hardware is central to measuring the performance of rendering devices such as monitor, plotters, printers etc. CIE publishes guidelines on measuring the color reproduction characteristics. The International Telecommunication Union (ITU) publishes recommendations for subjective assessment of quality of television pictures.

6 Summary

The ultimate goal of our work is to increase the effectiveness of the transcoding technique applied to internet data access. We currently focus on JPEG images. Given a particular image, we want to be able to determine whether performing a transcoding operation will pay off and how aggressively it should be done. Toward that end, this paper makes the following contributions to our understanding of the use of image transcoding:

1. We have defined the notion of a "quality aware transcoding" as a transformation that explicitly trades off image information quality for reductions in object size (e.g. transmission length or storage size).
2. We have found previous work that supports the use of the JPEG Quality Factor to capture our requirement for a quantifiable measure of image information quality. This allows us to consider JPEG compression metric as a candidate for quality aware transcoding.
3. We have developed an algorithm to estimate the JPEG Quality Factor used to originally produce an image. This is necessary to make "loss of quality" a meaningful concept.
4. We have characterized a sample of typical images available on the internet with respect to their origi-

nal size and JPEG Quality. This allows us to evaluate our work in the context of a realistic workload.

5. We have developed an algorithm to predict the computational cost involved in transcoding an image. This allows us to evaluate the cost penalty in performing a transcoding operation.

6. We have developed criteria that can be used to predict whether an image will transcode efficiently, avoiding serious mispredictions. This allows us to evaluate if it will be worthwhile to transcode a particular image so as to achieve a higher information quality per byte of image storage.

We are currently investigating heuristics to identify whether a transcoding from a GIF to a JPEG image format will be efficient for a particular image.

7 Acknowledgments

This research was supported in part by a dissertation research grant from IBM. We wish to thank Ashish Gehani for many discussions of the JPEG compression algorithm. We also thank Susan Paddock for pointing us to the correct statistical analysis techniques. We also thank our shepard, Prof. Eric Brewer for his valuable comments.

References

[1] America Online Inc. Johnson grace ART image format.

[2] D. Bourgin. Color spaces FAQ. www.well.com/user/rld/vidpage/color_faq.html, May 1995.

[3] CCITT Recommendation T.81, International Telecommunication Union(ITU), Geneva. *Digital Compression and Coding of Continous-Tone Still Images - Requirements and guidelines*, Sep 1992.

[4] S. Chandra, C. S. Ellis, and A. Vahdat. Multimedia Web Services for Mobile Clients Using Quality Aware Transcoding. In *The Second ACM International Workshop on Wireless Mobile Multimedia*, Seattle, Aug 1999.

[5] CNN Interactive: All Politics. cnn.com/ALLPOLITICS/, Dec 1998.

[6] R. Floyd, B. Housel, and C. Tait. Mobile Web Access using eNetwork Web Express. *IEEE Personal Communications*, 5(5), Oct 1998.

[7] A. M. Ford. *Relations between Image Quality and Still Image Compression*. PhD thesis, University of Westminster, May 1997.

[8] A. Fox and E. A. Brewer. Reducing www latency and bandwidth requirements via real-time distillation. In *Proceedings of Fifth International World Wide Web Conference*, pages 1445–1456, Paris, France, May 1996.

[9] P. Greenspun. www.photo.net, Dec 1998.

[10] E. Hamilton. *JPEG File Interchange Format - Version 1.02*. C-Cube Microsystems, Sep 1992.

[11] R. Han, P. Bhagwat, R. LaMaire, T. Mummert, V. Perret, and J. Rubas. Dynamic adaptation in an image transcoding proxy for mobile web browsing. *IEEE Personal Communications Magazine*, 5(6):8–17, Dec 1998.

[12] Intel Quickweb. intel.com/quickweb.

[13] A. A. Jr. and C. H. Null. Image quality: a multidimensional problem. In A. B. Watson, editor, *Digital Images and Human Vision*, pages 141–148. MIT Press, Cambridge, MA, Nov 1993.

[14] D. G. Kleinbaum, L. L. Kupper, and K. E. Muller. *Applied Regression Analysis and Other Multivariable Methods*. PWS - Kent Publishing Company, second edition, 1988.

[15] T. Lane, P. Gladstone, L. Ortiz, J. Boucher, L. Crocker, J. Minguillon, G. Phillips, D. Rossi, and G. Weijers. The independent jpeg group's jpeg software release 6b. ftp.uu.net/graphics/jpeg/jpegsrc.v6b.tar.gz.

[16] M. S. Mazer, C. Brooks, J. LoVerso, L. Theran, F. Hirsch, S. Macrakis, S. Shapiro, and D. Rockwell. Distributed clients for enhanced usability, reliability, and adaptability in accessing the national information environment. Technical report, The Open Group Research Institute, Cambridge MA 02142, 1998.

[17] Media Graphics International, Arvada, CO. 70,000 Multimedia Graphics Pack, 1997.

[18] B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker. Application-aware adaptation for mobility. In *Proceedings of the 16th ACM Symposium on Operating Systems and Principles*, Saint-Malo, France, Oct 1997.

[19] A. Ortega, F. Carignano, S. Ayer, and M. Vetterli. Soft caching: Web cache management techniques for images. In *IEEE Signal Processing Society 1997 Workshop on Multimedia Signal Processing*, Princeton NJ, Jun 1997.

[20] W. B. Pennebaker and J. L. Mitchell. *JPEG - Still Image Data Compression Standard*. Van Nostrand ReinHold, NY, 1993.

[21] Spectrum Information Technologies Inc. Fastlane. spectruminfo.com.

[22] Star Wars - Episode I. starwars.com/episode-i/, Dec 1998.