AUTONOMOUS IMAGE TRANSCODING FOR THE IMPROVEMENT OF WEB CONTENT SERVER AVAILABILITY

Presented to the Graduate Council of
Texas State University-San Marcos
in Partial Fulfillment
of the Requirements

for the Degree

Master of SCIENCE

by

Michael Edmund Butterfield, B.A.A.S.

San Marcos, Texas
December 2005

COPYRIGHT

by

Michael Edmund Butterfield, B.A.A.S.

2005

To Nancy and Claire

ACKNOWLEDGEMENTS

I would like to begin by thanking my wife Nancy and my daughter Claire for their encouragement, support, and tolerance of the long hours spent away from my family to complete my studies.

I am very thankful to the chair of my thesis committee, Dr. Gregory Hall. His class lectures captivated my attention, stimulated interest in the subject material, and generated an appreciation for the need of software processes and engineering measurement. Dr. Hall's lessons were well balanced between the topic theory, and the applied demonstrations that helped cement the ideas being presented.

I am also thankful for the other members of my committee, Dr. Stephen Osella, and Dr. Anne Ngu for taking the time to review and provide feedback on the results of my research topic; and the friends, neighbors, and coworkers that volunteered to participate in the image survey.

This manuscript was submitted on November 10, 2005.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF EQUATIONS

Page

ABSTRACT

AUTONOMOUS IMAGE TRANSCODING FOR THE IMPROVEMENT OF WEB CONTENT SERVER AVAILABILITY

by

Michael Edmund Butterfield, B.A.

Texas State University-San Marcos

December 2005

SUPERVISING PROFESSOR: GREGORY HALL

Due to the exponential growth of the web and the Internet, content delivery performance is a dominant theme in Internet processes development. This thesis investigates the practices and research on the topic of web performance, and focuses on data size reduction by changing JPEG image compression levels. Compression limits were determined by a user survey on usability and acceptability of an image. The compression limits were used to evaluate website content for potential performance improvement. This information was then used to identify processes required to maintain web server availability during high server loads. A web server prototype using a JPEG filter was constructed to demonstrate the capability of automatic switching to lower data size when high loads are detected.

CHAPTER 1

INTRODUCTION

On September 11, 2001 at 8:46 a.m. EDT, American Airlines flight 11 struck the North Tower of the World Trade Center. This would be the first of many events to unfold on that fateful day. As news of the event propagated around the United States and the world, the dominant news sources on the Internet were overwhelmed, and as a result, were rendered useless.

From 9 a.m. to 10 a.m. EDT, 0% of users were able to access news sites ABCNews.com, CNN.com, and NYTimes.com. The news sites USAToday.com and NBC.com were marginally better for user accessibility at 18% and 22% respectively (Eubanks 2001). Users in the New York area turned to other communication channels, such as television and radio, only to find them disabled due to the aircraft collisions. The public Emergency Alert System (EAS) which relies on television and radio broadcast transmitters was therefore disabled. The New York area was under a wide spread information blackout.

Historical email archives of the North American Network Operators Group (NANOG), noted that the Internet did experience an increase in traffic immediately after the first collision, but this was not a factor in the unavailability of news sites (Freedman 2001). Additional observations indicate that the leading contributor to news site unavailability was not the Internet itself, but the inability of the content servers to keep up

with user requests. Bellovin (2001) acknowledges that CNN quickly responded to their congestion by switching to basic text and low graphics. Mikula (2001), discussing the current state of the Internet, observed that it was "pretty poor performance for a network orginally [*sic*] designed to faciliate [*sic*] communication in just such circumstances."

The Akamai (n.d.) website shows that there is a distinct correlation between world news events, and the peaks in Internet usage. Akamai[1], a leading provider of Internet edge servers and host to over 100 news organizations, shows that obtaining news is the third largest use of the Internet, only surpassed by searches and email.

Years later, the effects of September 11 continue to leave indelible marks on the Internet. The Akamai *Net Usage Index for News* page reports that the Internet traffic corresponding to the "Memorial Coverage for the 4 Year Anniversary of September 11, 2001" was ranked #1 among major world news events at a rate of 3,294,300 peak visitors per minute world wide (Akamai 2005).

The activities of September 11 and many other peak demands placed on the Internet highlight the need for unattended processes that are capable of maintaining web server availability on the Internet, especially in times of crisis. This thesis investigates possible ways to fulfill this need.

---

[1] Akamai was founded by Daniel M. Lewin, a graduate from Massachusetts Institute of Technology (MIT). In a somber coincidence, Daniel Lewin was among 81 passengers and 11 crewmembers that perished aboard American Airlines flight 11, as noted in an Akamai press release (Akamai 2001).

Content delivery may be improved by increasing the delivery speed of the data, or by reducing the amount of data. Text data and image data are the two primary data types for web content. This thesis investigates the characteristics of still images and their compressibility to remove unnecessary data, while preserving the information the image intends to convey. The identified compression values will then be applied to web sample images to determine performance impact. A prototype web server using a request filter will be constructed to evaluate capability of automatically switching content during periods of peak load.

CHAPTER 2

LITERATURE SURVEY

This chapter presents a review of journal articles and other reference materials used in this study. This chapter begins with a broad review of Internet performance topics, and then narrows down to topics that directly impact the study.

Problem Statement

The World Wide Web (WWW), also know as 'the web', is a collection of services that operate over the Internet (Wikipedia n.d.). The web has changed the way we live. It has become integrated into our daily lives and affects the way we get our news and entertainment. The web is virtually a requirement to conduct commerce transactions for business and personal needs on the web. The growing dependency on web content servers highlights the need to maintain server presence during periods of peak demand.

The papers reviewed for this study almost unanimously highlight the recent exponential growth and use of the web and the Internet. One consequence of this explosive growth is the demand placed upon the technology that serves the Internet. When the capacity of the web server or network is exceeded, user requests may go unfilled.

This rapid growth highlights the need for continuous performance improvements to improve the web experience of the user. One might surmise that the rapid growth of

the Internet reduces the urgency to maximize content efficiency. The rapid growth of the internet is highly correlated with the exponential increase in computing power as predicted by Moore's Law[2]. The notion that content efficiency is no longer an urgent issue is refuted by Knutsson (2003). In his article on server directed content transcoding, he states that:

> The argument has also been made that transcoding, at any location, is unnecessary, because bandwidths and client capabilities are increasing. We reject this premise.

User perceived delay is another attribute that affects perception of usability and performance. The subjectivity of this topic is readily apparent when one reviews the statements and observations of various authors. Li, et al. (2004) discusses the idea of "user perceived delay" as a dominant issue on the Internet. Wills, et al. (2001) uses the term "time-to-glass" to describe the propagation delay of the user response.

Li, et al. (2004) suggests that users experience an 'unpleasant delay' if the response exceeds 7 seconds. Curran and Duffy (2005) classifies under 5 seconds as excellent, and 5-10 seconds as 'good'. Nakano (2002) uses 15 seconds as a value for acceptable download time in an example of his "adaptive content", although this may be for illustrative purposes only.

---

[2] Gordon Moore, co-founder of Intel, predicted that the number of components increase by a factor of about 2 every year. This prediction is known as "Moore's Law" (Intel, n.d.).

The effect of user perceived delay is downplayed by Curran and Duffy (2005) when they state that "most of the major players have already established a name for themselves and delays most likely would not put off their customers".

This study will focus primarily on improvement of availability, which will likely result in positive influence on user perceived delay and website performance.

Information and Data

There are many techniques to improve Internet performance. One, for example is the efficient use of data size. For this study, the initial supposition is that authored content can be effectively and dynamically reduced in size, which directly affects the efficiency of the user response chain. The primary goal is to represent the author's original idea (the information) with a reduced amount of raw data.

This idea is inspired by the work of Edward R. Tufte (1983), author of *The Visual Display of Quantitative Information*. This book discusses the use of graphics (a mix of letters, numbers, and images) to convey information. A dominant theme in this book is the term 'Data-ink', which is used to represent the amount of information that is represented with the drops of ink used to print the graph. It discusses the differences between 'erasable' and non-erasable ink. Tufte (1983) points out that "A large share of ink on a graphic should present data-information, the ink changing as the data change. '*Data-ink*' is the non-erasable core of a graphic" Any additional ink does not contribute to the graphic, and may actually *distract* from the original information. Tufte's original ideas can be extended beyond paper and ink media, by replacing 'ink' with binary data, and removing the 'unnecessary' binary data. This concept of removing unnecessary data

is corroborated by Knutsson (2003) as he discusses the use of content transcoding which "removes ('distills') inessential or unrenderable information". Knutsson, et al. (2003) and Li, et al. (2004) both refer to the image compression as "transcoding". This term will be discussed in further detail in later sections.

Using the idea of maximizing the information to data ratio, this study explores the potential of generating a web server data response using the smallest practical data size, close to the point of origin of the information. The point of origin for reduced data size may be authoring software, a data repository, a content server, or a proxy server. If the web server is unable to generate the requested content, the bandwidth performance characteristics of the internet are irrelevant.

Survey of Internet Performance Improvement Techniques

Numerous approaches to performance improvement have been studied or proposed by various journal articles, including bundling, caching, clustering, pre-fetching, content compression of text and images, routing, web switching, geographic co-location, and proxy servers. This list is certainly not exhaustive, but touches on common areas of study. Fundamentally, the basic factors for all these topics can be distilled to the *size* of the data content and the *speed* of the content delivery chain. One factor that affects choice of performance improvement is whether the data content is *static* or *dynamic*. For purposes of this study, static content is data usable by multiple users or sessions, while dynamic content is user or session specific.

Clustering is a common technique used on most high capacity websites as noted by Cardellini and Casalicchio (2002) and Mogul, et al. (1997). Clusters consist of

multiple application servers and hosts, which may or may not be in the same physical location. Cluster servers may be used for redundancy or load balancing. Requests are routed to servers based on load distribution algorithms or round robin techniques.

Caching is the ability to serve duplicate content from a location that potentially provides the best delivery speed. Servers can be geographically located closer to the recipient, or utilize specific hardware designed to serve the specialized content type. Cardellini and Casalicchio (2002), Li, et al. (2004), and Curran and Duffy (2005) point out that caching is typically limited to static content such as Hypertext Markup Language (HTML) text and images.

Knutsson, et al. (2003) studies the possibility of caching dynamic content using a process called *server directed transcoding* (SDT). SDT extends content server capabilities by embedding directives in the Hypertext Transfer Protocol (HTTP) response. The embedded directives are used within the proxy servers to alter content.

Edge caching is the placement of servers near the point of use to minimize the number of network hops required to reach the user. Edge caching is the dominant service offered by Akamai (n.d.), a leading provider of caching on the Internet. Akamai has numerous edge cache servers geographically located around the world. Akamai claims to handle about 15% of the Internet traffic in the world, averaging 1 billion hits per day.

Pre-fetching, discussed by Curran and Duffy (2005), and mentioned by Wills, et al. (2001), is a technique where the system attempts to predict the next probable action of the user based upon typical usage patterns, user-aware processes, or other heuristic algorithms and prepares the response in advance. Dynamic content is a strong candidate

for pre-fetching, provided that the performance improvements of correctly selected content outweigh the overhead for incorrectly selected content.

Bundling, discussed by Wills, et al. (2001), is the process of encapsulating all the objects of a web page into a single file. This eliminates the redundant HTTP requests needed for each of the objects typically used by the web page, and reduces the overhead of network traffic. A possible risk of bundling is the inclusion of content that may already be cached downstream in the delivery chain.

Efficient Authoring can significantly improve web performance. Curan et al. (2005) discusses inefficient coding and authoring as a contributing factor to degraded web performance. He provides 'Page Design Guidelines' to assist authors with best practices on web content design. The central theme for this thesis is based on the same idea that not all web content is well-authored, and the content can be effectively reduced in data size.

Efficient authoring includes prudent image compression, which will be discussed in further detail in the following section. In his discussion on SDT, Knutsson, et al. (2003) points out that for SDT to be effective, designers must become responsible for directing how their content might be encoded.

Reducing the Data Size of Text and Images

Compression of text and images is studied or mentioned by Wan and Moffat (2001), Curran and Duffy (2005), Kherfi and Ziou (2004), Wills, et al. (2001), and Muller (1998). Wills, et al. (2001) states that tools such as the GZip compression tool are only effective on text content, since GZip relies on pattern matching to reduce the size of the

content. Wills, et al. (2001) also note that image content is already compressed and GZip will not effectively compress the image further. Transcoding is the conversion from one encoding scheme to another, as discussed by Nakano, et al. (2002) and Knutsson, et al. (2003). Transcoding may be changing between graphic standards of the image (GIF, PNG, TIFF, JPEG, etc.), or changing the attributes of the existing graphics standard, such as height and width dimensions, cropping, compression. This study will focus on the JPEG compression attribute for reducing the data size.

Nakano, et al. (2002) discusses the use of transcoding modules to store images in an 'adapted content cache', but does not elaborate on tools used in the transcoding methods. They also discuss the advance preparation of content to avoid the latencies that can be introduced by attempting conversion at the time of the request, a technique that will be used in the web server prototype for this study.

In his study on SDT, Knutsson, et al. (2003) discusses techniques for transcoding content through the use of ImageMagick, an open source command line tool. ImageMagick is easily implemented using languages such as Perl, or using a Java Virtual Machine (JVM) through Application Programming Interfaces (API) to the native processes. This is the tool selected for this research project.

Curran and Duffy (2005) and Wills, et al. (2001) assert that most image content is primarily static. However, dynamic information may still come in the form of images, such as charts and graphs, which will not benefit from advance cache preparation. Dynamic image content is outside the scope of this study.

It is well known that images make up a significant portion of a typical web response, so benefit from any compression of image content is quickly realized. Different

image types may be stored in different formats or encoding schemes depending on the type of image. Muller (1998) points out that "In general, line drawings and flat-color illustrations are better stored in GIF format, while photographs and complex images are more suitable for JPEG format". Charts and graphs are generally line drawings and don't compress well without losing the underlying information. He also states that JPEG compression ratios as high as 25-to-1 may be used "without a noticeable loss of image quality. This is because the human eye has the capacity to fill in missing detail, making it intelligible to the viewer." He notes that quality degradation becomes apparent at higher compression ratios.

Lane (1999) declares that for JPEG images, "Quality settings around 50 are often perfectly acceptable on the web." Lane's statement does not take into account the context of the image, which affects how much an image can be compressed. A portion of this study seeks to identify the quality level at which an image is considered by a user to be acceptable or usable. For discussion purposes, *usability* is defined as the level where the intended information can be extracted from the image, and *acceptability* is the level where the image is no longer objectionable to the user when used as content on a web page. These terms will be discussed in further detail later in this study.

Graphic formats are discussed in further detail in Chapter 3. To aid with further discussion, the differences between encoding, compression, and transcoding are summarized:

- *Encoding* is the process of converting raw images into a standard graphic encoding format, such as JPEG.

- *Compression* is the reduction of data size, and is one of many attributes in a graphic encoding format such as JPEG.
- *Transcoding* is the conversion between one graphic encoding format and another, or changing attributes within the same graphic format. Since all JPEG images are compressed to some degree, a *transcoded* JPEG image refers to an original JPEG image that has received alterations in compression level.

Proxy Servers, Routers, and Web Switches

Proxy servers are a ubiquitous part of web architecture. A proxy server may serve content from a cache, provide firewall security, or route inbound packets to specific servers based on the context of the request. Proxy servers may include *web switches* at the front end node which act as a traffic dispatcher within a distributed system. Cardellini and Casalicchio (2002) discuss two switching methods in their paper, which may be encountered on a proxy depending on the layer of information used for decision making. Content-blind web switching uses the transport layer 4 (TCP) for packet routing, and Content aware web switching uses the application layer 7 (HTTP) for packet routing. Proxy servers are integral to the research of Knutsson, et al. (2003) and clearly rely on the use of the 'content-aware' (HTTP) layer for decision making.

Routers and web switches make decisions based on the content of the data packets, such as the IP address or the HTTP information. The web server prototype constructed for this study can be compared to the content-aware (layer 7) routing that Cardellini and Casalicchio (2002) discuss, since decisions are based on the HTTP

request. The overhead that is typical of content-aware web switching is minimized, since data inspection occurs at the beginning of the request chain.

Research Objective

This thesis studies the techniques, algorithms, and attributes that affect web server availability. The information derived from the study is used to develop a process that can be used to autonomously improve web server availability during periods of peak demand.

The primary focus of improving availability is the reduction of data size while maintaining usable information to the user. The process is then applied to content from arbitrarily selected websites to examine the effect on performance and availability. This thesis further investigates a technique for detecting the occurrence of peak demand in order to trigger the techniques to ensure web server availability.

The study is broken into 5 parts:

- Part 1 examines the effect of JPEG compression on the data size of various image type samples such as portraits, maps, line art and illustrations.
- Part 2 investigates levels of image usability and acceptability by typical web users. These levels are required to determine the minimum JPEG quality levels at which an image from part 1 can be compressed.
- Part 3 applies JPEG image transcoding to website samples to adjust compression to a quality level derived from part 2. The website samples are selected from popular news and information websites to study the potential data size savings.

- Part 4 measures server performance differences between delivery of the original and transcoded website samples from part 3.
- Part 5 demonstrates a web server process that is capable of switching between original and transcoded JPEG content based on performance metrics, in order to utilize reduced image data size from part 3.

CHAPTER 3

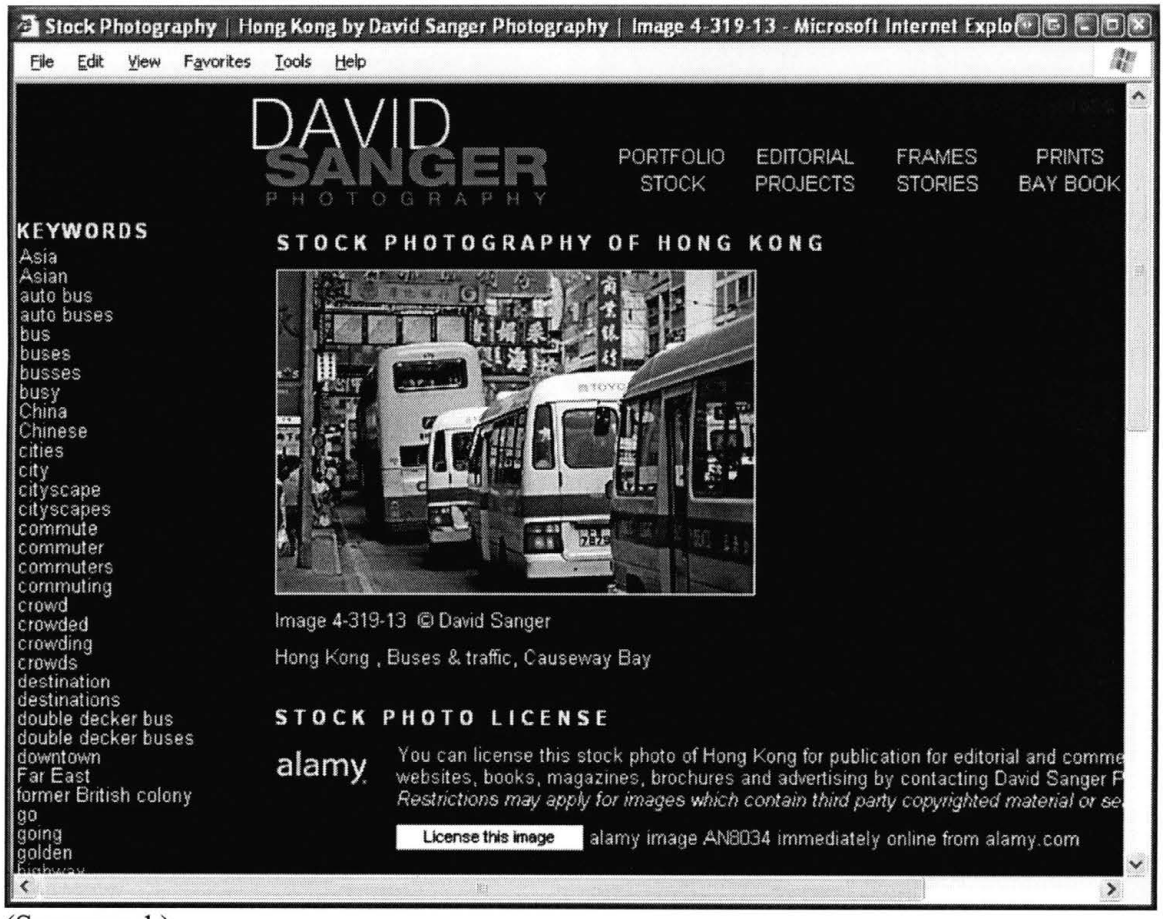
BACKGROUND INFORMATION

This chapter provides additional detail regarding the types, context, and encoding methods of images; the HTTP Request-Response cycle occurring between the browser and the web server; the behavior of a web server when a request is received and the processing of the request by the application server.

Image Taxonomy and Context

Two factors to be considered when compressing images are their taxonomy and context. The *taxonomy* indicates the type of image, such as single or group photos of people, maps, text, line art, graphs, or charts. The *context* of an image is how the image is used, such as news, advertisements, or financial. While there is a correlation between taxonomy and context, there is not a one to one relation between both domains. When information is critical, such as on a financial graph, compressed content may be detrimental since the risk of misinterpretation goes up when the image quality goes down. In other situations, 'lossy' compression levels or formats may be undesirable for the user on websites targeted at the photographic aficionado, such as www.nationalgeographic.com. Opportunities for aggressive compression exist on news and sports websites, where images complement and enhance the text content.

A good example of image type and context is found at www.davidsanger.com, where images are identified by keywords (Sanger n.d.). There can easily be 100 or more keywords to explicitly describe each image by type or context. A keyword can be selected to query and display a group of other images that have the matching keyword. This study will be limited to a few broad image type descriptions, such as portrait, group, landscape, cityscape, and line art.



(Sanger n.d.)

Figure 1: Website Sample Showing Keywords

Cardellini and Casalicchio (2002) and Kherfi, et al. (2004) point out that it is a difficult task to determine the compressibility of an image without having knowledge of

the image taxonomy or context, or using a heuristic based tool to determine the image taxonomy. For this study, some general assumptions on compressibility will be made, based on the encoding utilized by the image.

As specified in the JPEG standard, this encoding scheme is intended for use on continuous tone images, which by definition exclude high contrast images such as charts, maps, and line art. Appropriate encoding schemes should therefore be considered during the content authoring process. Continuous tone images are strong candidates for aggressive image compression, which in turn dramatically reduces the file (data) size.

High contrast images may still benefit from aggressive JPEG compression; provided the benefits of compression are weighed against the potential risks of information misinterpretation. The test environment for this research study includes images from both categories of continuous tone images and high contrast images.

Graphic Encoding Standards

A pixel is a single light point that is created by combining the output of red, green, and blue light, emitted from an RGB cathode ray tube (CRT) or light emitting diode (LED). Variations in the intensity of each primary color can generate virtually infinite combinations of color and intensity. An analog image can be digitally recreated by representing the image as a two dimensional array of closely spaced pixels, where image quality is relative to pixel density. Higher pixel densities result in images with crisp, focused images. The quantization effect of the individual pixels is reduced to an imperceptible level by the human eye, as the pixels decrease in size.

The intensity of each pixel is controlled by a digital value representing each of the three RGB colors. Typically, each color is represented by a byte (8 bits) that can generate 256 levels of intensity per color, for a combination of 16.7 million RGB colors. At 3 bytes (24 bits) per pixel, a 640 x 480 image requires 921,600 bytes (almost 1 mb, or about 2/3 of a typical 1.44 mb floppy disk) to represent a single image, without encoding. Specialized applications, such as the medical industry, may use higher color resolution, which result in proportionately larger file sizes. The un-encoded file size of an image can become sufficiently large to adversely affect storage and performance. To address this problem, there are many graphic encoding standards in use.

Image encoding is affected by the type of the image, as discussed in the previous section, a balance between preserving storage space and quality, and the usability and acceptability of the image at the endpoint. Each graphic standard has advantages and disadvantages that are described here.

The W3C website (http://www.w3.org/Graphics) provides a summary of common graphic formats in use on the web. There are many other sources of information, such as the websites hosted by the associated standards committees. Some of these websites are shown in Table 1. This table shows dominant graphic standards used on the web today. Many standards were excluded due to the limited application support, such as JPEG2000, or waning Internet usage, such as TIFF.

Table 1: Common Graphic Standards

Type	Full Description	Standard	URL
JPEG	Joint Photographic Experts Group	ISO/IEC 10918-1	www.jpeg.org
GIF	(CompuServe) Graphic Interchange Format	GIF87A GIF89A	www.w3.org/Graphics/GIF/spec-gif89a.txt
PNG	Portable Network Graphics (also PNG's Not GIF)	ISO/IEC standard 15948:2004	www.libpng.org

Additional properties of graphic standards are described in Table 2. This table only covers a few key attributes. An important consideration is the proprietary nature of each standard. PNG was designed with the intention of being patent free. JPEG is not constrained by patents as of this writing. GIF is currently protected by an IBM patent through August of 2006 (Wikipedia [1], n.d.).

Table 2: Properties of Graphic Standards

Type	Lossless	Usage	Image type	Unencoded Bits/Pixel	Proprietary
JPEG	No	Continuous tone images	True color, greyscale	Typical 24 bit (16.7 M color)	No
GIF	Yes	Line art such as Charts, Graphs	Palette	256 color (8 bit)	Yes Expires in 2006 (www.gnu.org)
PNG	Yes	GIF replacement, Continuous Tone	True color, greyscale, palette	Typical 24 bit (16.7 M color), up to 48 bit	No

Image Transcoding

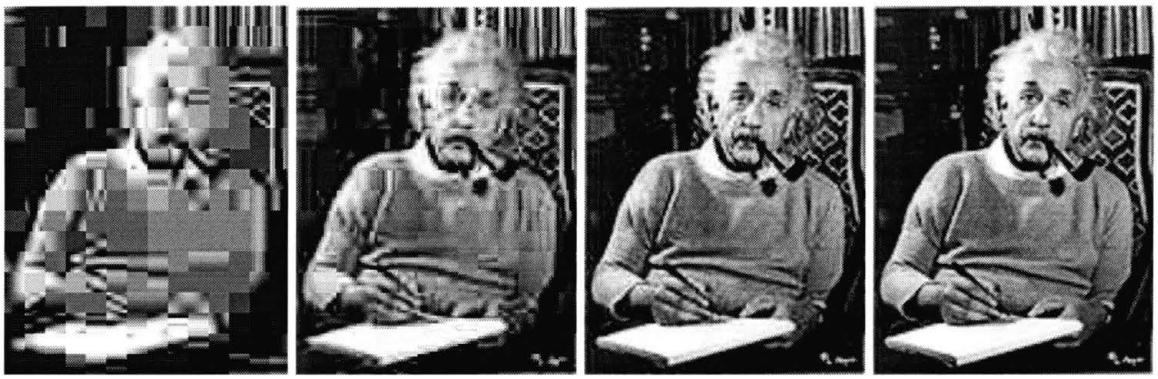
Image transcoding is modification of the image properties, such as the encoding standard, dimensional size, cropping, color resolution, or quality factor in order to match

the requirements for the image's intended use. Examples of intended use may be creating a small monochrome thumbnail for display on a wireless device, such as a cell phone, cropping background of an image to focus on a person's portrait, or decreasing the file size for improvements in storage size and transmission performance. This study focuses primarily be discussing compression by altering the JPEG quality factor (QF).

Image utilities are inconsistent in the use of values that represent JPEG compression levels. For example, Microsoft Visio erroneously uses the term *percentage* to represent quality level. Paint Shop Pro indicates compression factor from 1 to 99, with 1 representing best quality. Other utilities use verbal descriptors, such as good, better, and best. To standardize values for the remainder of this study, the quality factor value from the ImageMagick utility will be utilized. QF levels will be rated from 1 to 100, with 100 representing the best quality level.

JPEG Compression Standard

As previously mentioned, JPEG is intended for 'continuous tone' images, which exclude line art, maps, charts, or other high contrast images. JPEG is not a universal encoding standard that can be applied to all images. Improperly applied encoding is the result of poor authoring of the source information. Figure 2 shows an original JPEG image and two compression samples.



QF 1 QF 10 QF 50 Original QF95 (Aigner 1940)

Figure 2: JPEG Samples at Different Compression Levels

JPEG compression is based on the Discrete Cosine Transform (DCT), which encodes each color channel as a greyscale image. The image is divided into 8 x 8 pixel blocks for a total of 64 pixels per block (Wallace 1991). A Forward DCT (FDCT) is applied to each block to calculate a coefficient for each pixel, and the Inverse DCT (IDCT) decodes the coefficients to generate the original pixels. The FDCT and IDCT are shown in Equation 1 and Equation 2 (Wallace 1991). Sequential, progressive, hierarchical, and lossless are the 4 encoding modes available in the JPEG standard. Baseline sequential encoding will be the primary focus of this discussion, and the remaining modes are outside the scope of the discussion.

$$(1)\quad F(x,y)=\frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7}f(x,y)\cdot\cos\frac{(2x+1)u\pi}{16}\cdot\cos\frac{(2y+1)v\pi}{16}\right]$$

(Wallace 1991)

Equation 1: Forward Discrete Cosine Transform

$$(2)\quad f(x,y)=\frac{1}{4}\left[\sum_{u=0}^{7}\sum_{v=0}^{7}C(u)\cdot C(v)\cdot F(u,v)\cdot\cos\frac{(2x+1)u\pi}{16}\cdot\cos\frac{(2y+1)v\pi}{16}\right]$$

Where:

$C(u), C(v) = 1/\sqrt{2}$ for $u, v = 0$;

$C(u), C(v) = 1$ otherwise.

(Wallace 1991)

Equation 2: Inverse Discrete Cosine Transform

Wallace (1991) explains that the DCT is related to the Discrete Fourier Transform (DFT) which is better understood by "viewing the FDCT as a harmonic analyzer and the IDCT as a harmonic synthesizer." Each of the 64 points is represented as a coefficient, beginning with a DC coefficient for the first point, and then calculating an AC coefficient for the 63 remaining points that represent the spatial frequencies offset from the DC coefficient. Higher compression is achieved when the spatial frequencies have low amplitude, which reduce the amount of information that needs to be encoded. The process is reversed during decoding to change the coefficients into the original 64 points of the image.

Next, the coefficients from the FDCT are quantized, to achieve further compression. The QF determines the step size of the quantization, and is selected based on the "perceptual threshold" (Wallace 1991). Wallace (1991) states that:

> The purpose of quantization is to achieve further compression with no greater precision than is necessary to achieve the desired image quality. Stated another way, the goal of this processing step is to discard information which is not visually significant.

This is an important statement, as it is consistent with the assertion by Tufte (1983) to remove unnecessary information, and the central theme of this study. The phrase "desired image quality" from Wallace's statement is a subjective value that needs quantification, if it is to be successfully applied in the course of data size reduction.

Next, entropy encoding is used to further compress the coefficients generated by the FDCT. The process is reversed during the decoding process prior to application of the IDCT formula. Two methods of entropy encoding and decoding are used. The first utilizes Huffman encoding tables, and the other uses arithmetic encoding. An image encoded with one can be decoded with the other.

As the compression of an image increases (by lowering quality factor), the file size grows smaller. Empirical observation of a typical JPEG image is shown in Figure 3. This indicates an exponential function with a base less than 1. The first approximately 20 QF values have a steep negative slope ($a < -1$) when comparing file size to quality factor. The slope begins to level out to a negative linear slope ($-1 < a < 0$), for the remaining QF values.

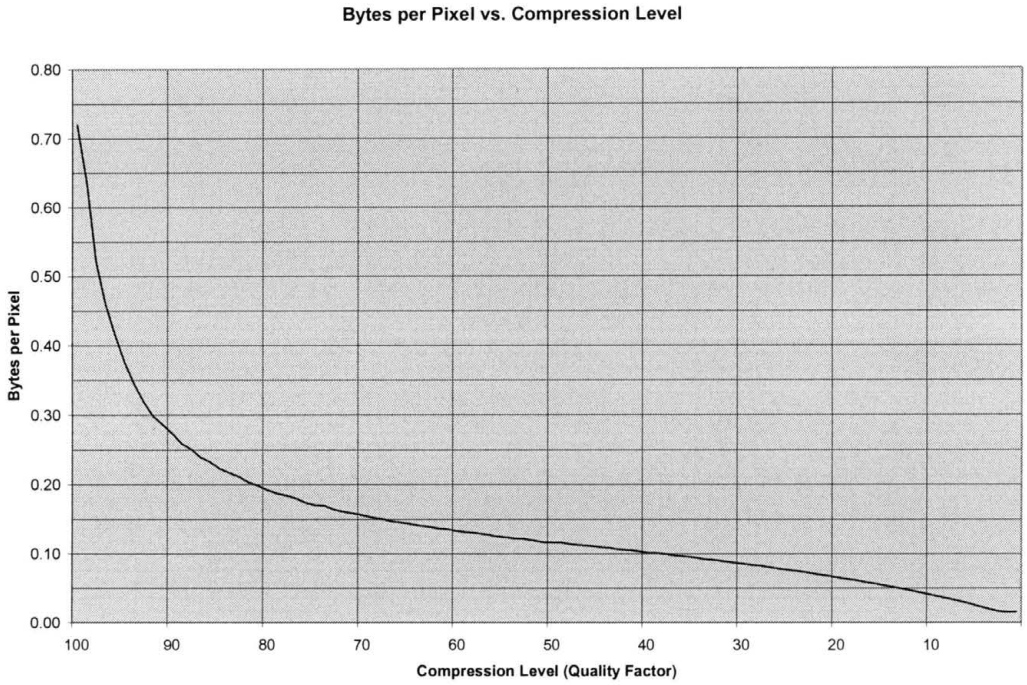


Figure 3: Typical JPEG, Bytes per Pixel vs. Compression Level

Wallace (1991) provides a guideline for quality and compression which compares pixel density with quality. This is shown in Table 3. Percentage estimates have been added to the table, based on 24 bits per pixel encoding. This guideline remains somewhat ambiguous on the usage, and supports the need for quantifiable usability values.

Table 3: JPEG Pixel Density vs. Quality

Pixel Density bytes/pixel	**Quality**	**Usage**
0.25 – 0.5 (≈ 8 – 17%)	Moderate to Good	Sufficient for some applications
0.5 – 0.75 (≈ 17 – 25%)	Good to Very Good	Sufficient for many applications
0.75 – 1.5 (≈25 – 50%)	Excellent	Sufficient from most applications
1.5–2.0 (≈ 50 – 66%)	Indistinguishable from Original	Sufficient for the most demanding applications

The 7 Layers of the OSI Model

The web server prototype will utilize content aware routing to intercept JPEG requests. To clarify previous discussions surrounding content-aware (layer 7) and content-blind (layer 4) routing and distribution, a quick review of the International Organization for Standardization (ISO) Open System Interconnection (OSI) model is warranted. The OSI model describes a series of interface layers that allow two computers to communicate across a network. The layer names and numbers, along with a few implementation samples, are shown in Table 4. The web server prototype for this study will utilize content aware (layer 7) decision making.

Table 4: OSI Model Layers

No.	Layer	Examples of Standards
7	Application	HTTP, FTP, POP3, SSH
6	Presentation	HTTP, FTP, XML, Telnet
5	Session layer	NetBIOS, Named Pipes, NFS
4	Transport	TCP, UDP, IPX/SPX
3	Network	IPv4, IPv6, IPSec
2	Data Link	Ethernet, ATM
1	Physical	Electrical characteristics: 10Base-T, 10Base2

A typical HTTP Session

The next sections review the HTTP request-response cycle with increasing detail, to clarify the proposed implementation techniques for a web server prototype. The goal is to intercept and generate an alternate response to an HTTP request.

An HTTP Session begins as the request is resolved using the beginning portion of the URL as shown in these strings:

```
http://localhost:8080/servlets-examples/servlet/HelloWorldExample
http://localhost:8080/jgadget/myImage.jpg
```

The following steps correspond to the action vectors displayed in Figure 4. Details of the web and application containers are provided in the following sections.

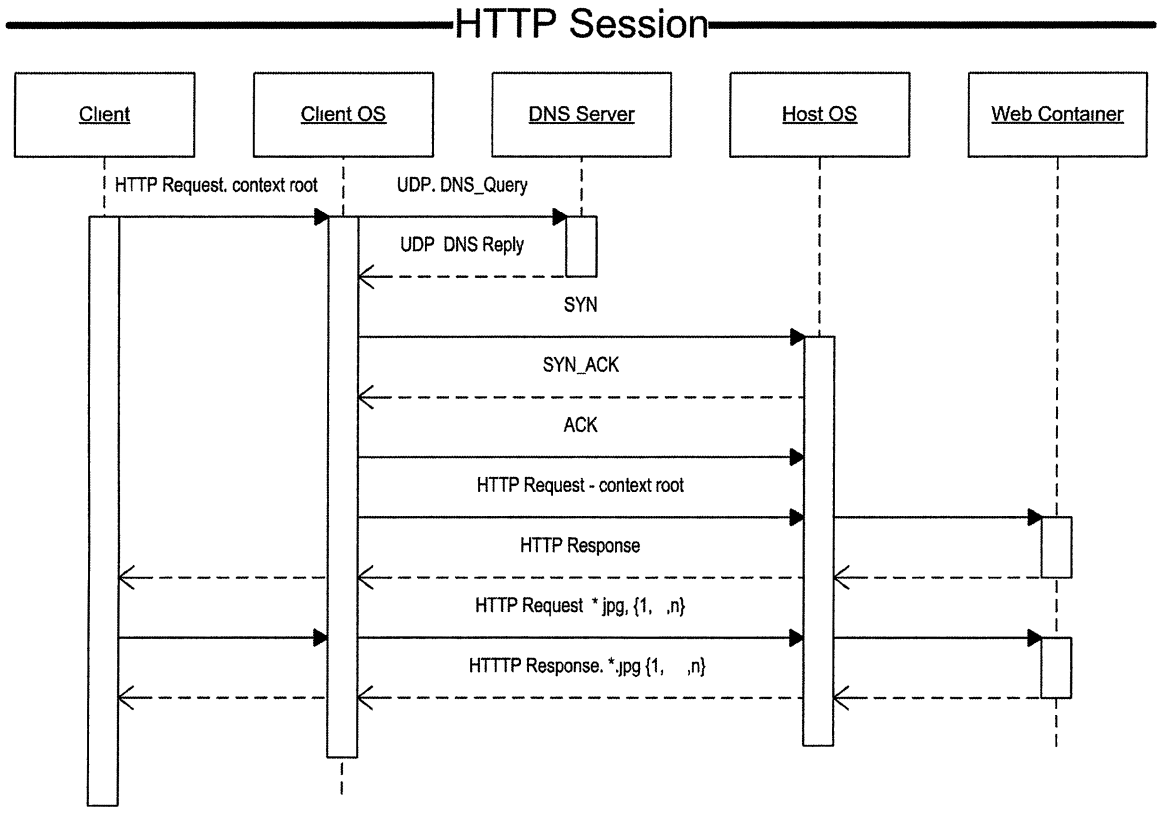


Figure 4: Sequence Diagram for HTTP Request-Response

- The HTTP browser request is initiated by a user, typically by entering a URL into a browser application, or by selecting a hyperlink in a browser or web enabled document.
- The client operating system (OS) begins establishing a TCP/IP connection, by sending the hostname to an authoritative DNS server using a DNS query.

- The DNS server returns the IP address of the hostname using a DNS Reply (http://localhost in the above examples).
- The client OS establishes a TCP/IP connection to the host OS using the IP address with a SYN, SYN-ACK, ACK sequence. The client OS is now ready to send the HTTP request to the application server
- The HTTP Request is sent to the application server, as determined by the port number (8080 in the above sample).
- The application server generates an HTTP response, which likely contains an HTML page with referenced objects.
- HTTP requests are generated for each referenced object in the HTML page, and the Application server returns the requested objects. JPEG objects are shown in the illustration, since they are the focus of this study.
- The TCP/IP connection is closed by either the client or the server.

The HTTP/1.0 standard required a new connection to be established for each HTTP request to the server. This activity created redundant and unnecessary network traffic. HTTP/1.1 addressed this issue by optionally allowing multiple requests using a single connection, thus reducing the unnecessary overhead.

Due to the processing overhead and increased data size associated with security encryption and to minimize performance impact on the server, secure connections should be reserved for transactional and sensitive information.

Web Container

This section expands the web container lane of the previous sequence diagram. When the host operating system receives a TCP/IP request, and after a session is established, the request is forwarded to the application listening on the specified port. At this point the web server processes the request and returns the appropriate response using the context root identified in the URL request. The context root follows the host and port portion of the URL as shown in the following:

```
http://localhost:8080/servlets-examples/servlet/HelloWorldExample
http://localhost:8080/jgadget/myImage.jpg
```

A sequence diagram is shown in Figure 5. The Host OS is included to help maintain context with previous illustrations.

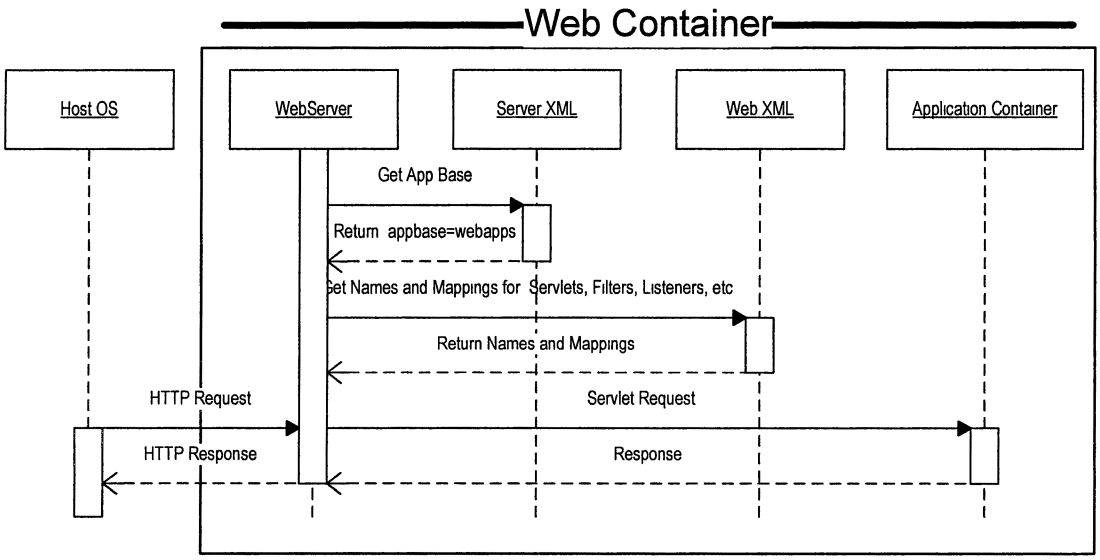


Figure 5: Sequence Diagram for Web Server Request-Response

Note that several actions occur prior to receiving an HTTP request from the OS.

- On startup of the web server, or hot deployment of an application, the server retrieves information located in the server.xml file to determine the base directory for web applications and deploys applications located in the base directory.
- For each web application, the web server identifies the mappings for servlets, filters, and listeners using the information located in the web.xml
- Using this information, HTTP requests are forwarded to the appropriate application container.
- A response is generated by the application container and is returned to the Host OS for return to the client.

Application Container

This section is of special interest, as this is where the request will be intercepted and processed by a filter. The filter is responsible for modifying the request or response based on the rules detailed in the test environment. The Application container lane from the previous sequence diagram is expanded here. The web container is included to help maintain context with previous diagram.

Once the request is received by the application container, the request is processed according to the instance mappings established in the web.xml deployment descriptor and the context portion of the URL, as shown in these sample strings:

```
http://localhost:8080/servlets-examples/servlet/HelloWorldExample
http://localhost:8080/jgadget/myImage.jpg
```

The associated actions are shown in Figure 6. Activities of interest are highlighted in bold italics.

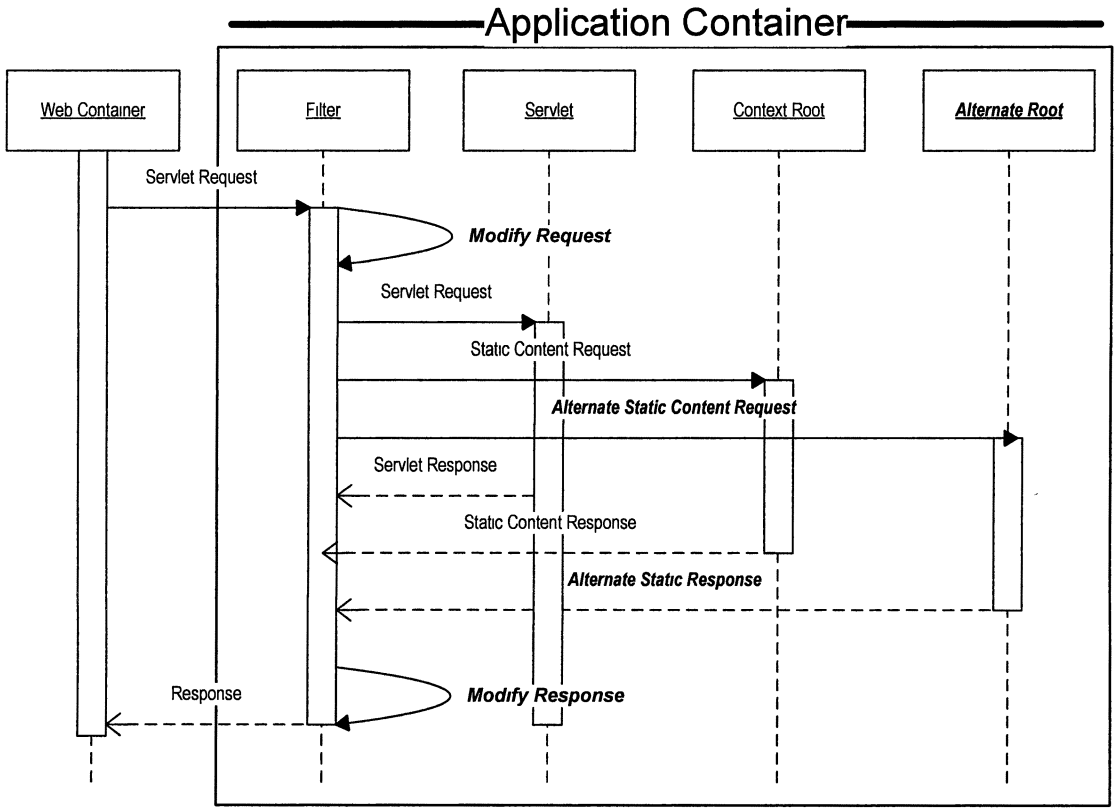


Figure 6: Sequence Diagram for Application Container

- The request is intercepted by a filter chain and processed according to the content type, such as JPEG.
- The request is *modified* if it meets the conditions of the business rules.
- The request is then routed to the appropriate resource, depending on the state of the original or modified request. The request may be forwarded to a servlet, or request may retrieve content from a specified data repository, such as the original context folder, or an *alternate* shadow folder.

- The response is returned to the filter chain.
- The response is modified if it meets the specified business rule conditions.
- The response is returned to the web container for return to the client.

For purposes of this study the test environment will be focused on processing the request to identify JPEG content, and these specific requests will be modified to return content from the alternate repository if the conditions are met. Generally, the two factors that influence the filter decisions are whether the server load exceeds an event threshold and if alternate content exists that matches the request.

There are two apparent alternatives to modifying the delivered content. The initial HTML response could be modified to replace all image references with references to the alternate content. While this may seem to be an effective solution, this approach may inadvertently result in a server load *increase*.

It is possible that the content of the original page may have been previously cached on the client browser or other points on the network, such as a proxy server. By modifying the image URL, the cache is no longer utilized, and the browser will need a fresh instance of the image. This issue impacts sites that do not change content frequently, and provide longer time to live (TTL) values for the delivered content. A possible exception to this scenario is a website where the image content changes with sufficient frequency, such as a news site, that the caching issue can effectively be ignored.

Since the primary purpose of this study is to explore the capability of reducing loads induced by data size, this study will focus on modifying JPEG image request.

CHAPTER 4

CONDUCTING THE RESEARCH

This chapter discusses the research methodology, general assumptions, image transcoding, image survey process, website sample collection, and construction of the web server prototype. Next, the functional requirements for the test environment are reviewed, with actors and use cases. An overview of the hardware and software used to implement and execute the test environment is presented.

Research Methodology

The research methodology begins by collecting images that represent various image types, such as portrait close-ups, groups, landscape, maps, and line art. These images are used as samples for a user survey to evaluate image quality. A utility will be written to automate the process of transcoding each original image into incremental compression values, and the values will be studied to examine the data size at each level.

Next, a user survey is performed using the transcoded images to determine if the images are usable and acceptable at specific compression levels. There is an important distinction between usable and acceptable. A usable image allows a reader to extract useful information from the image, but may be undesirable in appearance. An acceptable image is both usable and desirable. The results of the user survey will be used to evaluate possible data size savings on existing websites.

Several popular news and information websites were visited and a snapshot of each site was taken. Each snapshot was then used to measure the relative size differences between the original content and content that has the JPEG images transcoded to a usable threshold. This collection of images represents *popular websites* and is different from the collection of images that represent various *image types*.

Using the analyzed information for usable compression, a simulated website will be created for the purpose of measuring the response times for original and modified website image content. The web server prototype will include a transcoding utility that monitors the content folder of the website, and automatically creates a shadow directory structure that contains compressed images.

A sample web server utilizing a JPEG filter was implemented to demonstrate the potential of automatically responding to high server load and return content with a lower compression level.

Assumptions

Wherever practical, tools are selected from a collection of open source software that fall under variations of the Gnu Public License (GPL), Apache Software Foundation License (ASF), or similar public domain licenses. Open source tools enable the study of existing architecture and permit the extension or modification of the product, if required, while mitigating risk from patent or copyright infringements. In addition, cost factors associated with using proprietary software are avoided. The distributions are also readily available for download from the web.

Consideration will be given to scripts and code that are platform independent. This allows execution on multiple software platforms such as Linux, UNIX, and Windows. The proposed techniques are intended to remain within the constraints of existing standards for technologies affecting the application layers such as the HTML 4.01, HTTP/1.1, TCP/IPv4, Java Servlet 2.3, JPEG ISO/IEC 10918-1, and other related standards.

Image Type Sample Collection and Transcoding

Image samples were collected to represent different image types and uses. A conversion utility was written to automatically convert the images into to the appropriate compression levels required by the user survey. The conversion utility is included as an artifact in the Image Survey Processor section of the Appendix.

Image complexity requires that samples represent a wide spectrum of image attributes. As previously stated, the *type* of image affects the level of image compressibility. Additionally, the *usage* or context of the image also affects the level of compressibility that can be applied to an image. The 20 images selected for the user survey are shown in Table 5. For each image, only one contrast column is selected. Any of the remaining columns may be selected for each image.

Table 5: Image Types and Usage

	General Attributes (Absolute)													Usage (Relative)						
Image Description	Continuous Tone	High Contrast	Single Person/Subject	Multiple Person/Subject	Landscape/Cityscape	Photographed Text	Generated Text	Map	Line Art	Chart	Pixels X	Pixels Y	Area in^2	News, Sports	Entertainment	Financial Data	Information	Reference	Public Safety	Risk
Donald Trump	✓		✓								300	247	7 4	✓	✓					1
Double Decker Bus	✓				✓	✓					450	320	14 4					✓		1
Map of San Marcos		✓					✓	✓			1024	781	80 0				✓		✓	2
Finance Chart		✓					✓		✓	✓	328	218	7 2			✓	✓			4
Microscope Graticule		✓					✓		✓		640	638	40 8				✓			3
Bush on Boat	✓			✓							379	279	10 6	✓						1
Park illustration		✓			✓		✓		✓		700	455	31 9				✓			2
Fire Alarm Pull Station	✓		✓			✓					215	324	7 0					✓		2
Seal on Rock	✓		✓								450	320	14 4					✓		1
'Pi' Chart, Multi-line Text		✓					✓				340	400	13 6				✓			3
NY Yankee Player	✓		✓			✓					120	72	0 9	✓						1
Golden Gate	✓				✓						450	320	14 4					✓		1
Hurricane Map		✓					✓	✓		✓	516	338	17 4				✓		✓	5
Barometer Graph		✓					✓				320	200	6 4				✓		✓	2
Dilbert Cartoon		✓		✓			✓				750	544	40 8		✓					1
Alan Greenspan	✓			✓							540	351	19 0	✓				✓		1
School Illustration		✓			✓		✓	✓			1024	733	75 1				✓			2
Coke collectible	✓		✓			✓					640	480	30 7					✓		1
Horse in Meadow	✓		✓								400	262	10 5					✓		1
6, 8 and 10Point Text		✓					✓				129	253	3 3				✓			3
Count (*Mean)	10	10	6	3	4	4	10	3	3	2	486*	377*	22 3*	4	2	1	9	7	3	

Image *type* attributes may be considered *absolute* attributes as they represent physical observations of the image without considering influence from external factors. Image *usage* attributes may be considered *relative* attributes as they are primarily influenced by external factors.

Absolute image attributes may include:

- *Contrast* – Images such as photographs may have continuous tone which is a gradual transition in brightness between pixels. Maps, charts, text,

barcodes, and line art are typically high contrast where adjacent pixels have significant difference in brightness values.

- *Perspective* – Images may be further classified in categories such as single persons or subject, multiple persons or subjects, landscapes, cityscapes, maps, line art, or charts.
- *Text* – Text may be coincidentally or intentionally captured during the course of photography. Text may also be generated or added by computer applications for the explicit purpose of conveying information.
- *Size* – Dimensional size and subject size affect the compressibility of an image. Larger images and/or subjects are able to sustain higher compression.

Relative attributes may include:

- *Usage* – The intended usage of an image will determine the amount of image transcoding that can be applied. Websites that provide news, sports, and entertainment can afford aggressive image transcoding, while websites that convey information where risk is high, such as financial, weather, and public safety need to carefully consider risks of misinterpretation. Usage extends beyond human interaction. Images may be targeted at Optical Character Recognition (OCR) readers, or barcode scanners, in applications such as shipping labels or movie tickets.
- *Risk Value* – Risk represents the possible impact from image misinterpretation. An arbitrary scale of 1 to 5 was used, with 1 representing low risk and 5 representing high risk. Risk may be measured

numerous ways. For this discussion, risk will be considered potential for financial loss, property loss, or personal injury. Risk value is directly affected by most of the identified attributes.

User Survey of Image Usability and Acceptability

Image quality is the fundamental property that affects the outcome of the test environment. The values of interest are the *usable* quality level and the *acceptable* quality level of a JPEG image. These values ultimately determine the compressibility of the image.

Usability and Acceptability

There is an important distinction between usability and acceptability. An image becomes usable at the point that a user can discern the *information* that the image is intended to convey, even though the image may still be of poor or objectionable quality. An image becomes acceptable at the point the user determines that the image is *no longer* objectionable for usage on a typical web page. Acceptability is a highly subjective opinion of the user.

Usability is easier to quantify than acceptability by using questions that require the user to extract information from the image. Questions are written based on typical or expected usage of the image. The questions may include recognition of the subject material, reading text, finding directions, or determining a numeric value.

Consideration was given to reduce the user influences that may arise due to differences in educational, cultural, personal interests, or occupation affiliation. The user

was instructed not to guess at an answer, as the survey is not a knowledge test, but rather a usability test. Accordingly, a *skip* option was provided for each question.

Usable and acceptable values are represented by the QF attribute of the JPEG image. As noted in previous discussions, QF values are related by non-linear multi-variate formulas. For the purposes of this survey, it is sufficient to treat the QF values as ordinals to determine the minimal value at which an image becomes acceptable or usable. The survey document developed for the user survey is included as an artifact in the Appendix.

The test environment was controlled to reduce the number of factors affecting the outcome of the survey. The subjective image evaluation is affected by numerous environmental factors such as the display size, type (CRT, LCD), age (new, several years old), resolution (pixels per inch), quality (brightness, contrast, focus, gamma, etc.), browser size, font size, and numerous other attributes. To minimize the variance of these attributes, all user surveys were conducted on the same hardware system. Details of the hardware are shown in Table 7 of the following sections.

Image presentations were displayed in order from lowest to highest quality. The test intends to avoid starting with the highest quality, since the user may resolve the information conveyed by the image, and mentally carry information forward, distorting results from lower quality images. The usable value will be determined first, and then the acceptable value will be determined.

Usability was determined by presenting the user with the initial image and asking a question about the image. The user incrementally increased the QF factor until the question was answered. This process was repeated for all sample images. Once the

images had been evaluated for the usable value, the images were again evaluated for the acceptable value.

Compressibility

To determine if an image is compressible, it needs to exceed the quality levels of a usable image, and preferably exceed the quality levels of an acceptable image. This is expressed in three possible combinations, as clarified by Equation 3.

$$(1)\ \forall I_o\ (\exists\ (I_o \wedge QF_t)\ \ni\ QF_u \leq QF_a \leq QF_t < QF_o \rightarrow I_u \wedge I_a) \vee$$

$$(2)\ (\exists\ (I_o \wedge QF_t)\ \ni\ QF_u \leq QF_t < QF_o \leq QF_a \rightarrow I_u \wedge \neg I_a) \vee$$

$$(3)\ (\exists\ (I_o \wedge \neg QF_c)\ \ni\ QF_u \leq QF_t < QF_o < QF_a \rightarrow \neg I_a \wedge \neg I_u)$$

where:
I = image
QF = image quality factor
a = acceptable
c = compressible
t = transcoded
o = original
u = usable

Equation 3: Logic Statement for Compressible Quality Factor

- If the QF of the original image is better than the QF of the acceptable image, which is also greater or equal to the QF of the usable image, then the original image can be compressed to an acceptable level.
- If the QF of the original image is better than the QF of the usable image, then the image can be compressed to a usable level.

- If the QF of the original is equal to or less than the QF of a usable image, then it is uncompressible.

Survey Sample Size

The survey used 20 participants, a usable or acceptable percentage of 95%, a confidence level of 95% (allows for 1 rejected image at a specific QF level), and the sample population is undefined (infinite). Using

Equation 4, the confidence interval is calculated to be 9.55%.

$$ss = \frac{Z^2 \cdot (p) \cdot (1-p)}{c^2}$$

Where:

ss = sample size
z = z value
p = percentage picking a choice
c = confidence interval

Equation 4: Survey Sample Size

Web Content Sample Collection

Web content samples were collected from arbitrarily chosen sites where content changes on a regular basis, such as news, weather, sports, financial, and entertainment sites. Some of these sites are noted as major news providers by Akamai (n.d.). These samples were used to evaluate changes in file size, based on the results of the user survey.

Applying Survey Values to Web Content Samples

A QF value derived from the user survey was applied to samples of existing web content to determine if there was a statistically significant improvement in the content file size. The samples are then utilized to measure the performance differences between the original JPEG content and the transcoded JPEG content.

Web Server Prototype Using a JPEG Filter

The final phase of the project provides a working model of a web server using autonomous image transcoding. As discussed earlier, a web container intercepts the JPEG request from the client browser and presents the user with alternate content during peak server loads. An event trigger based on thread count or memory usage was implemented to demonstrate two potential techniques for indicating high server load.

To provide alternate compressed images, a utility was written to search the original web repository for images. Any images that are found are transcoded to a value determined by the user survey and stored in a new shadow repository. The shadow repository is also reviewed to remove images that are no longer present in the web repository.

The ImageManager.sh code artifact is displayed in the Image Manager section of the Appendix. The web.xml, and JPEGFilter.java code artifacts are displayed in the Tomcat Web Server JPEG Filter section of the Appendix.

Functional Requirements

This section provides use case diagrams, actors, services, and use cases for the User Survey, Image Manager, and Web Server Filter features.

User Survey

Portions of the user survey use case are manual, such as the administration of the survey, and portions utilize a computer system for viewing the image collection. The user survey use case diagram is shown in Figure 7.

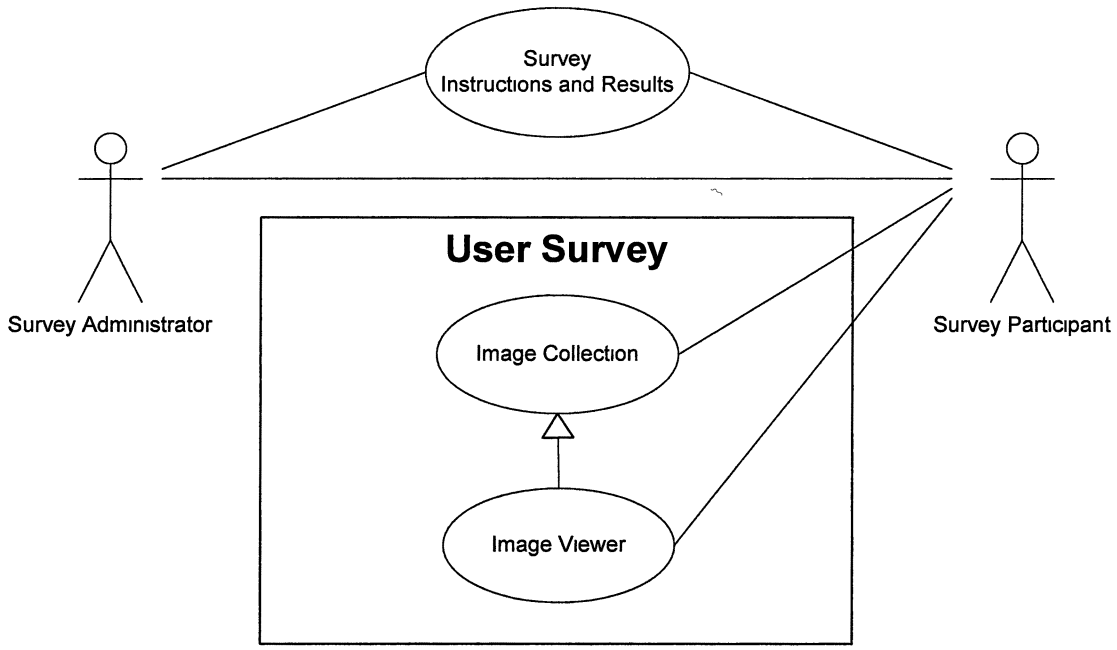


Figure 7: Use Case for User Survey

Actors

- *Survey administrator* is the person responsible for presenting the user to the participant.
- *Survey Participant* is the person inspecting the images for usability and acceptability, and recording the results of the inspection.

Additional Services

- *Image Collection* is the repository of transcoded images

- *Image Viewer* is a system utility for rendering images on the computer display.

Main Flow: Usable Survey Use Case

1. The use case begins when the administrator provides a demonstration of the image survey using a sample image.
2. The participant reads and follows instructions for part 1, the usability portion of the survey.
3. The participant reads the question and possible answers, then navigates to the image collection folder matching the question number and opens the Start image.
4. The participant navigates to images of higher quality until the image quality is sufficient to answer the question or the end is reached.
5. The user records the appropriate answer and the usable QF of the image.
6. The use case ends if there are no additional images for evaluation.
7. The use case returns to step 3.

Acceptable Survey Use Case

1. The participant reads and follows instructions for part 2, the acceptability portion of the survey.
2. The participant navigates to the folder matching the question number and opens the Start image.
3. The participant navigates to images of higher or lower quality until the participant determines the image to be at minimum acceptable quality, or the end of images is reached.

4. The user records the acceptable QF of the image.
5. The use case ends if there are no additional images for evaluation.
6. The use case returns to step 2.

Image Manager

The image manager monitors a content directory for the existence of JPEG images. The images are transcoded and stored in an alternate directory for independent use by other processes, such as a web server. The use case diagram for the Image manager is shown in Figure 8.

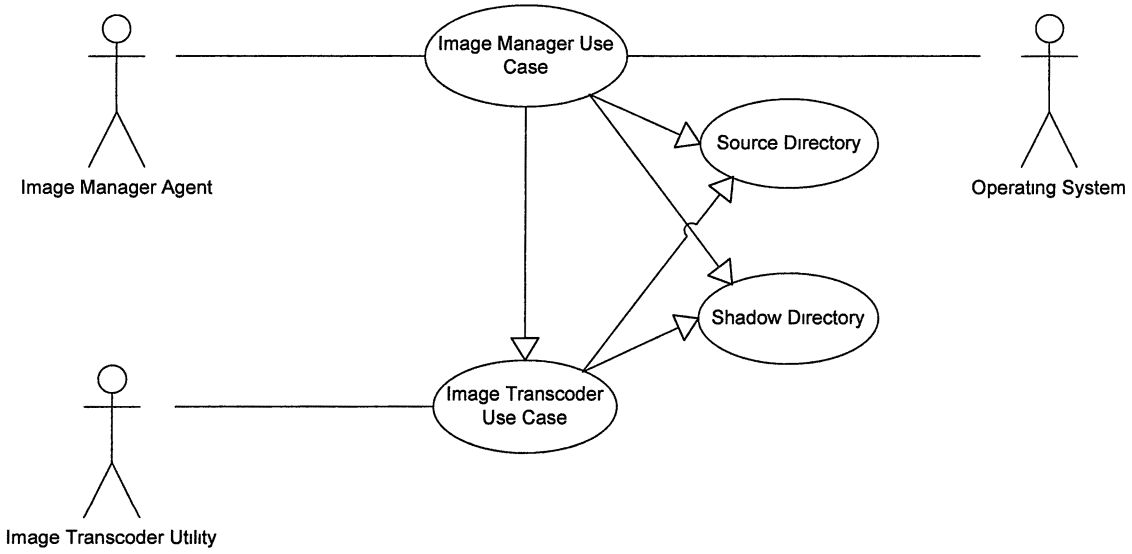


Figure 8: Use Case for Image Manager

Actors

- *Image Manager Agent* is the utility used to manage the source directory, shadow directory, and call the transcoding utility for image conversion.

- *Operating System* is the software used to control and manage the host hardware, such as the CPU.
- *Image Transcoder Utility* is the software used to compress original images to a specified level.

Additional Services

- *Source Directory* is the main content directory, such as the web context root.
- *Shadow Directory* is the alternate content directory where transcoded images are stored.

Main Flow: Image Manager Use Case

1. The use case begins when the Image Manager agent is started or re-entered, as a low priority process on the operating system.
2. The agent sleeps for a specified time period.
3. The agent recursively searches the source directory tree for JPEG images.
4. If a JPEG image is identified, the *Image Transcoding Use Case* is called.
5. The agent recursively searches shadow directory tree for JPEG images.
6. If a JPEG image is identified in the shadow directory, the source directory is inspected for a matching image.
7. If a matching image is not located, the image is deleted.
8. The use case ends when the compression agent is stopped.
9. Return to the beginning of this use case.

Alternate Flow: Image Transcoding Use Case

1. The use case begins when called from another use case.
2. The shadow directory is searched for a filename that matches the identified image.
3. If a match exists, the use case returns to the calling use case at the step following the call.
4. The image is inspected for compression level.
5. If the current compression level exceeds a threshold QF value, the image is transcoded to the threshold QF value and saved to the shadow directory tree location.
6. If the current compression level is equal to or less than a threshold QF value, the image is copied to the shadow directory tree location.
7. The use case ends and returns to the *calling use case.*

Web Server Filter

The Web Server Filter intercepts JPEG image requests to the content repository. If server load is high as determined by performance metrics, the HTTP request is altered to return alternate content to the client. The web server filter use case is shown in Figure 9.

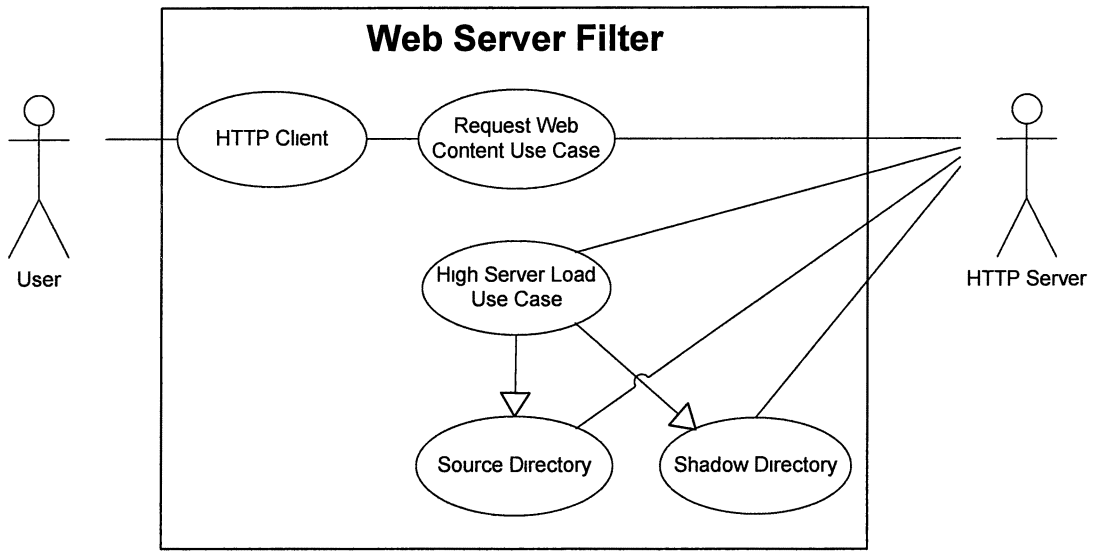


Figure 9: Use Case for Web Server Filter

Actors

- *User* is the person using the HTTP browser on a client machine.
- *HTTP Server* is a web server process running on a host machine, such as Tomcat.

Additional Services

- *HTTP Client* is a browser process running on a client machine, such as Firefox or Internet Explorer.
- *Source directory* is a file repository such as the context root of a web server.
- *Shadow directory* is a file repository that can be used by a web server to access alternate content.

Use Cases

Main Flow: Request Web Content

1. The use case begins when the user submits the URL in the browser.
2. The HTTP Client browser sends the HTTP Request to the server.
3. If the request is not a JPEG image type, the use case goes to the last step of this use case.
4. If the server load is high, as indicated by thread count or memory usage, read count or memory usage, call *High Server Load Use Case.*
5. The Server processes the original or modified HTTP request, and then sends the response to the client.
6. The HTTP Client received the HTTP response from the server.
7. The use case ends when the browser renders the response or the web server process is terminated.

Alternate Flow: High Server Load Use Case

1. The use case begins when called from another use case.
2. For a JPEG image identified in the request, the shadow directory is searched for the matching compressed image.
3. If the compressed image exists, the HTTP request is modified to point to the image in the shadow.
4. Log the HTTP request modification activities.
5. The use case ends and returns to *calling use case.*

Test Environment Implementation

This section provides a summary of hardware and software selected for the test environment. Links to software sites are provided for open source distributions. Additional information about each tool is discussed in the appropriate context as necessary.

Hardware

Table 6: Hardware for Development and Test Environment

Hardware	Description
System	Dell Dimension 9100 Desktop
Processor	Intel 830 Dual Core 3.0 GHz Processor
Memory	1 GB Dual Channel Memory
Drive	RAID-0

Table 7: Hardware for User Survey

Hardware	Description
System	Sony Vaio PCG GR390 Laptop
Processor	Intel Pentium III 1.2 GHz
Memory	512 MB
Display Type	LCD
Display Size	15 in. diagonal (SXGA+)
Display Resolution	1400 x 1050 pixels
Pixel Density	117 pixels per inch (approx.)

Software

Table 8: Test Environment Software

Type	Name	Version	URLs for Open Source Distributions
Operating System	Microsoft Windows	XP Professional, SP2	
Browser	Internet Explorer Firefox	6.0 SP2	
Browser with HTML editor	Mozilla	1.7.11	http://www.mozilla.org
Application Server	Tomcat	5.5.9	http://tomcat.apache.org
Virtual Machine Platform	Java Development Kit (JDK)	1.5.4_04	http://java.sun.com
Build Tool	Ant	1.6.5	http://ant.apache.org
Performance Measurement	Jakarta JMeter	2.1	http://jakarta.apache.org/jmeter
Image Transcoder	ImageMagick	6.2.4-1	http://imagemagick.org
Linux Shell Emulator	Cygwin	1.5.18-1	http://www.cygwin.com
Integrated Development Environment (IDE)	Eclipse	3.1	http://eclipse.org

Image Transcoder

ImageMagick is a software suite for transcoding (manipulation) of bitmap images. The distribution includes the source code and (optionally) platform specific binaries. ImageMagick can easily be interfaced with most common programming languages, such as C, Java, Perl, and many others. Interface API's (both freeware and cashware) such as JMagick are available.

UNIX shell scripts were selected for implementing the transcoder in order to demonstrate the capability of deployment on multiple operating system platforms. This image editing software is the primary focus of the "image transcoding utility" used to manage the shadow cache, as well as automate the generation of survey image samples.

Web Container with JPEG Filter

Jakarta Tomcat is recognized as a reference implementation for Java Servlet and Java Server Pages. This web server was used in the test environment to serve standard HTML pages and JPEG images. As previously noted, dynamic web content is outside the scope of this study.

In order to redirect image requests to an alternate source, a sample HTTP filter was modified and implemented as a JPEG filter on the Tomcat web server. The filter was designed to intercept JPEG requests by identifying files with a .jpg file extension. Based on the state of an event trigger, the filter may 'pass through' the original image, or return a transcoded image, provided the image exists in an alternate directory. The transcoded image content of the alternate directory is managed by the Image Manager.

Measurement

JMeter is a Java application designed to load-test functional behavior and measure performance of web applications. It was utilized to collect web page response times from original and modified content from the website sample collection.

CHAPTER 5

ANALYSIS OF RESULTS

Effect of JPEG Compression on Data Size

The 20 images selected for the user survey were transcoded in quality factor (QF) steps of 1, beginning at 1 and ending at the original QF level. The resulting data size was compared with the QF level of the image and was graphed as shown in Figure 10. About 30% of the survey images had an original compression value of greater than QF 90, and all but one image had an original compression value above QF 80.

The data size appears to have a relationship that is exponential with a base less than one, until the value reaches an approximate QF value of 80, where the relationship changes to a relatively linear relationship with a negative slope. Based on these results, the highest ratio of data savings occurs in the general range of QF100 to QF90, and then tapers off around QF80.

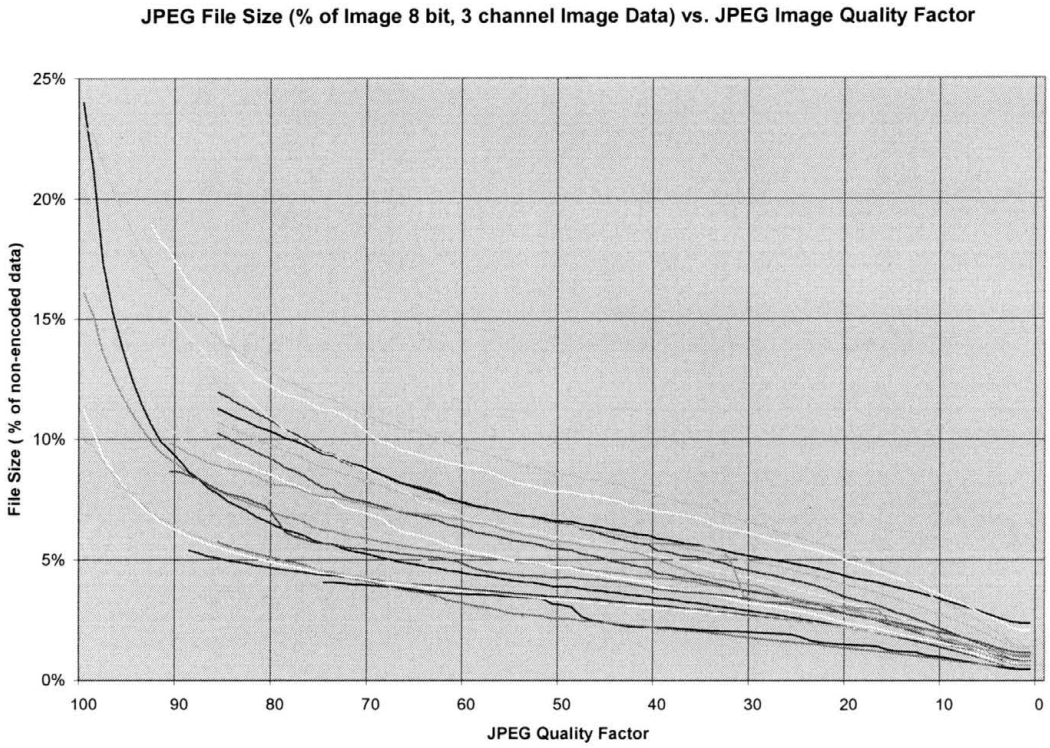


Figure 10: Effect of Quality Factor on Image File Size

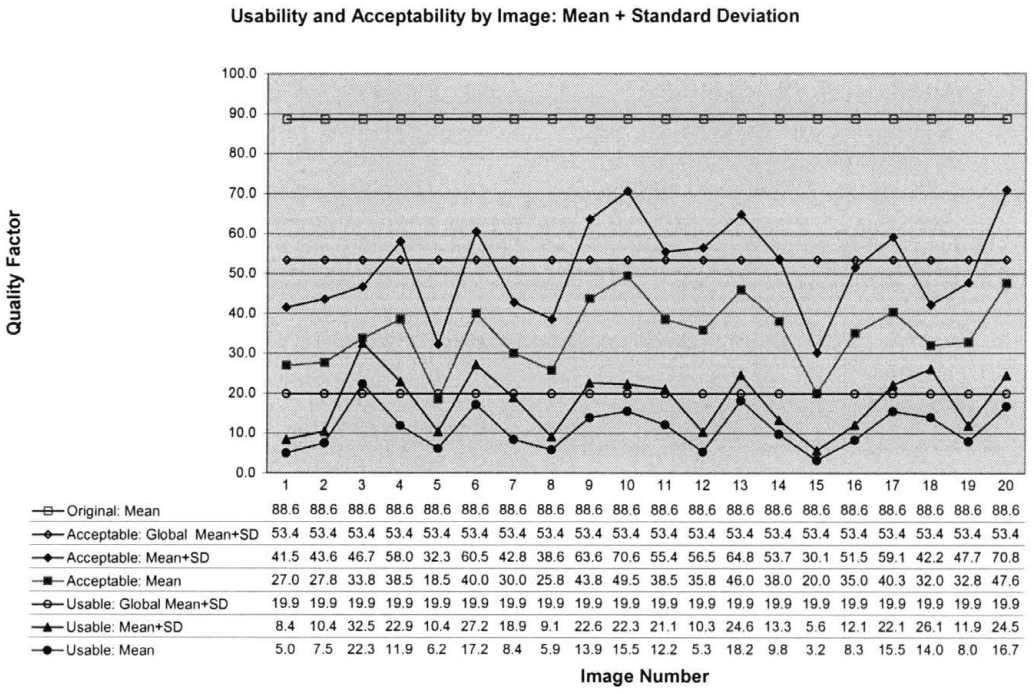
Effect of Compression on Image Usability and Acceptability

The user survey results were examined to determine the effects of image types and how image complexity affects the usability and acceptability of an image. Additionally, the results were examined to determine if a participant was particularly lenient or critical of the image quality. Observations show that in all cases acceptability was rated higher than usability, which is consistent with intuitive expectations.

Means plus Standard Deviations values were graphed. Results grouped by images are shown in Figure 11, and grouped by participant are shown in Figure 12. A wide

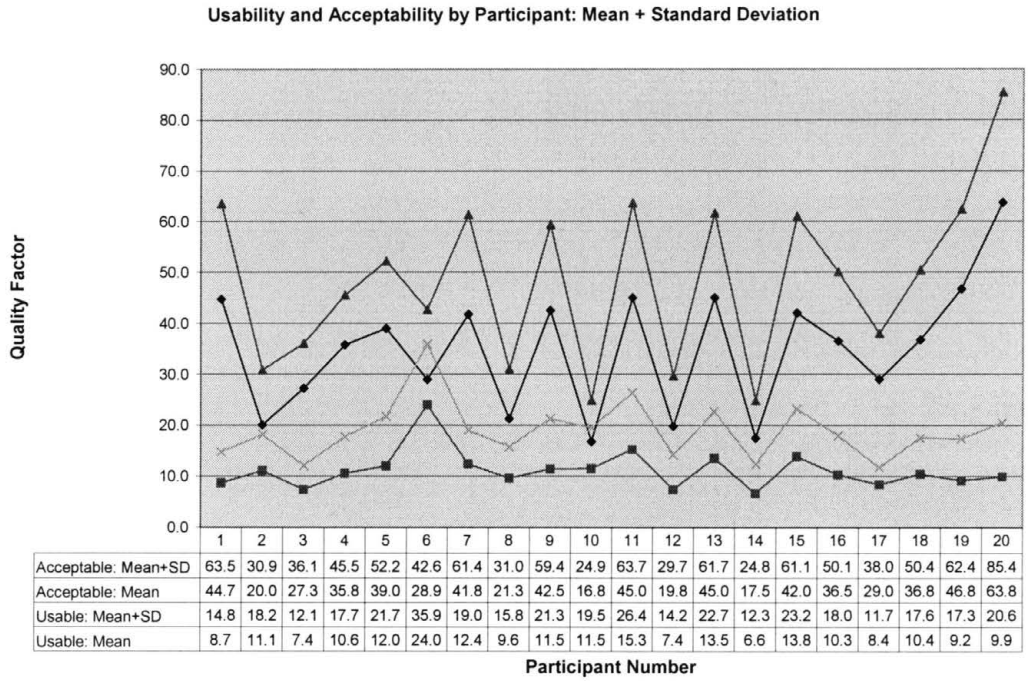
deviation is noted on the acceptability of images based on participant. The deviation for usability of images is relatively consistent across all participants.

As previously suggested by Lane (1999), a QF level of 50 appears to be a valid value that is acceptable by many participants, and usable by all participants within the boundaries of the survey questions. The survey results appear consistent with Lane's suggestions.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Original: Mean	88.6	88.6	88.6	88.6	88.6	88.6	88.6	88.6	88.6	88.6	88.6	88.6	88.6	88.6	88.6	88.6	88.6	88.6	88.6	88.6
Acceptable: Global Mean+SD	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4
Acceptable: Mean+SD	41.5	43.6	46.7	58.0	32.3	60.5	42.8	38.6	63.6	70.6	55.4	56.5	64.8	53.7	30.1	51.5	59.1	42.2	47.7	70.8
Acceptable: Mean	27.0	27.8	33.8	38.5	18.5	40.0	30.0	25.8	43.8	49.5	38.5	35.8	46.0	38.0	20.0	35.0	40.3	32.0	32.8	47.6
Usable: Global Mean+SD	19.9	19.9	19.9	19.9	19.9	19.9	19.9	19.9	19.9	19.9	19.9	19.9	19.9	19.9	19.9	19.9	19.9	19.9	19.9	19.9
Usable: Mean+SD	8.4	10.4	32.5	22.9	10.4	27.2	18.9	9.1	22.6	22.3	21.1	10.3	24.6	13.3	5.6	12.1	22.1	26.1	11.9	24.5
Usable: Mean	5.0	7.5	22.3	11.9	6.2	17.2	8.4	5.9	13.9	15.5	12.2	5.3	18.2	9.8	3.2	8.3	15.5	14.0	8.0	16.7

Figure 11: Usability and Acceptability by Image



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Acceptable: Mean+SD	63.5	30.9	36.1	45.5	52.2	42.6	61.4	31.0	59.4	24.9	63.7	29.7	61.7	24.8	61.1	50.1	38.0	50.4	62.4	85.4
Acceptable: Mean	44.7	20.0	27.3	35.8	39.0	28.9	41.8	21.3	42.5	16.8	45.0	19.8	45.0	17.5	42.0	36.5	29.0	36.8	46.8	63.8
Usable: Mean+SD	14.8	18.2	12.1	17.7	21.7	35.9	19.0	15.8	21.3	19.5	26.4	14.2	22.7	12.3	23.2	18.0	11.7	17.6	17.3	20.6
Usable: Mean	8.7	11.1	7.4	10.6	12.0	24.0	12.4	9.6	11.5	11.5	15.3	7.4	13.5	6.6	13.8	10.3	8.4	10.4	9.2	9.9

Figure 12: Usability and Acceptability by Participant

Data Size Changes on Website Samples using Image Transcoding

Data size of web content was compared in several combinations to determine the amount of content that was eligible for compression and the potential size savings that resulted specifically from JPEG image transcoding. Samples of original and transcoded images are included in the JPEG IMAGE SAMPLES section of the Appendix.

Comparison of Original Text and Image Files

Twenty websites were arbitrarily selected; and the content of each site was saved locally. The content was sorted and summed to identify the unique file types and collective size of the text and image files and is shown in Table 9.

The table includes typical statistical measures for each file type. The primary text content type was Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript (JS). Other files included text files without extensions. MSNBC was the sole user of .ASPX type files. Of the total image content population, the approximate percentages of each graphic format were JPEG at 51%, GIF at 42%, and PNG at 6%. Only two sites utilized PNG images, including Akamai and the NOAA local weather site.

Table 9: Data Size Comparisons between Web Content Text and Images

Units=Bytes	Text Files					Image Files				
Sample Site	Index HTML	CSS	Other HTM	JS	Other	GIF	JPG	PNG	Total	File Count
ABC News	47316	85804	0	14063	30024	22913	45081	0	245201	56
Akamai	72125	7581	0	22785	0	29916	0	37246	169653	115
BBC	44793	5835	0	1306	0	13885	15747	0	81566	27
CNET	160989	83401	0	15025	0	133571	19559	0	412545	82
CNN	78784	59528	0	55314	0	39938	30891	0	264455	75
Dilbert	48135	0	0	679	1897	99197	0	0	149908	51
ESPN	62436	32060	11425	42741	154	21873	48569	0	219258	45
Google	171474	0	0	0	0	6828	80525	0	258827	30
MSNBC	76492	47903	336	45901	13022	16833	60613	0	261100	33
National Geo	44876	4035	27463	32198	0	30930	35390	0	174892	29
NASDAQ	114509	9821	3897	51099	0	31764	0	0	211090	43
NYTimes	88067	5283	6512	27796	0	44895	22082	0	194635	50
NOAA, 7 day	21935	3063	0	0	0	8884	84675	54139	172696	27
NOAA, Aus-SA	34377	3102	0	0	0	21667	37419	0	96565	29
NOAA, Tropical	45038	2303	0	0	0	25226	36504	0	109071	30
People	27066	64034	4791	39569	0	8986	69289	0	213735	27
Statesman	191982	35399	0	26203	0	8458	75244	0	337286	32
USAToday	105242	2941	0	21129	0	24751	42300	0	196363	46
Yahoo	92370	0	0	0	0	9058	14170	0	115598	28
Weather Ch	71894	39231	0	92370	111	7962	34203	0	245771	34
Total	1599900	491324	54424	488178	45208	607535	752261	91385	4130215	889
Mean	79995	24566	2721	24409	2260	30377	37613	4569	206511	44
Deviation	48121	29265	6559	24787	7154	31887	26262	14328	80886	23
Min	21935	0	0	0	0	6828	0	0	81566	27
Median	72010	6708	0	21957	0	22393	35947	0	203727	34
Max	191982	85804	27463	92370	30024	133571	84675	54139	412545	115

A comparison between image data size and total data size is shown in Figure 13. Image content compared with native text shows a mean value of approximately 38%. This ratio changes significantly if GZip compression is applied to the text content, which is a common practice on web servers. Image content compared with GZip text shows a mean value of approximately 72%.

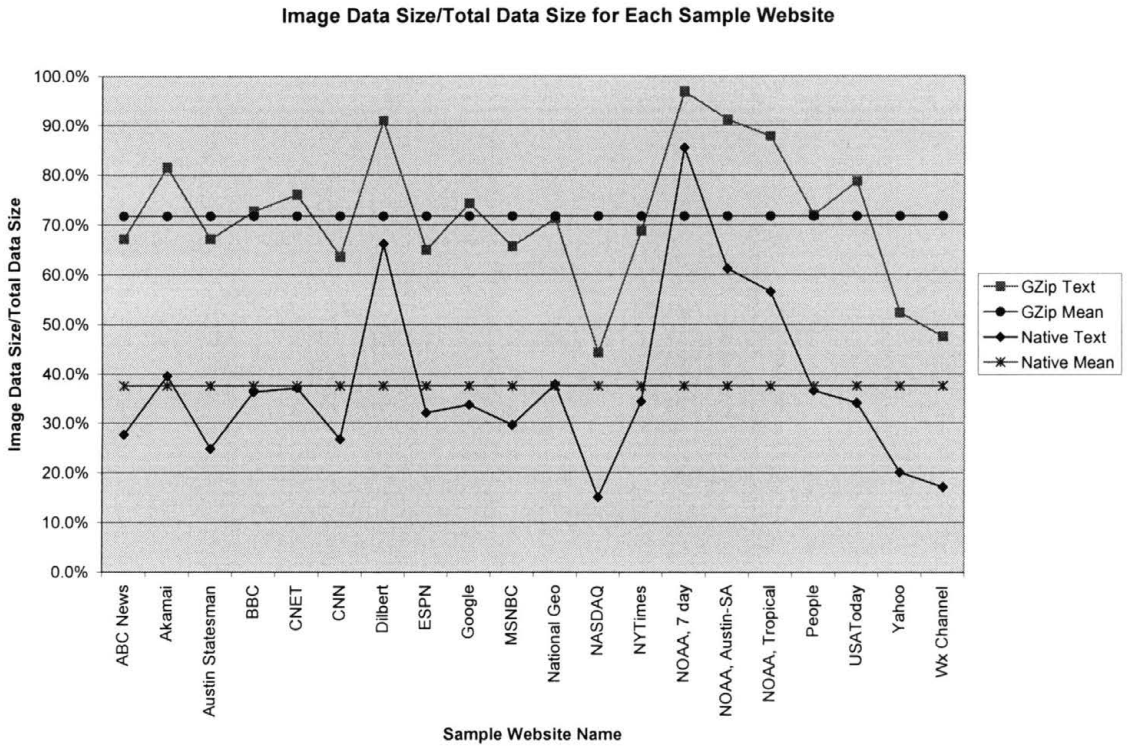


Figure 13: Comparison between Image and Total Content Data Size

Comparison of JPEG Original and Transcoded Images

Using the information derived from the user surveys as a guideline, the website JPEG sample images were transcoded to a threshold of QF50. For purposes of brevity in tables and remaining sections, the current compression level will be appended to the

JPEG acronym. For example, JPEG transcoded at a QF50 becomes JPEG50. The resulting file sizes are recorded in Table 10.

Table 10: Data Size Comparison between JPEG Original and JPEG Transcoded

Units = Bytes	A	B	C	D
Sample Site	JPEG	JPEG50	JPEG50/ JPEG	Data Size Savings
ABC News	45081	18133	40.2%	59.8%
Akamai	0			
Austin Statesman	75244	57152	76.0%	24.0%
BBC	15747	10499	66.7%	33.3%
CNET	19559	11600	59.3%	40.7%
CNN	30891	17638	57.1%	42.9%
Dilbert	0			
ESPN	48569	33518	69 0%	31.0%
Google	80525	70475	87.5%	12.5%
MSNBC	60613	48450	79.9%	20.1%
National Geographic	35390	17793	50.3%	49.7%
NASDAQ	0			
NYTimes	22082	13521	61.2%	38.8%
NOAA, Austin 7 day	84675	54549	64.4%	35.6%
NOAA, Austin-SA	37419	15634	41.8%	58.2%
NOAA, Tropical	36504	23746	65.1%	34.9%
People	69289	42866	61.9%	38.1%
USAToday	42300	27411	64.8%	35.2%
Yahoo	14170	8697	61.4%	38.6%
Weather Channel	34203	34203	100.0%	0.0%
Total	752261	505885		
Mean	37613	29758	65.1%	34.9%
Standard Deviation	26262	18795	15.0%	85.0%
Min (Max)	0	8697	40.2%	59.8%
Median	35947	23746	64.4%	35.6%
Max (Min)	84675	70475	100.0%	0.0%

The ratio between the transcoded and the original JPEG files is shown in column D. Based on the selected website samples; a savings of approximately 35% in image size is realized through transcoding. Some site samples exhibit significant potential for savings, such as ABC at approximately 60%, and NOAA Austin-SA at 58%. Other site samples are well optimized, such as Google, MSNBC, and the Weather Channel.

Comparison of JPEG and Total Content

The data size comparison between JPEG and total content was performed two ways, since it was unknown whether the text content originated in a native format or a compressed format from the server. The first subsection examines JPEG to total content size with native text, and the second subsection examines JPEG to total content size with text compressed with GZip.

Total Content Comparison using Native Text Format

To maintain reference with previous tables, Columns A and B are carried over from Table 9 to Table 11. These values are then compared with the transcoded JPEG (JPEG50) images of column C. The observed values from columns A thru C are then utilized to calculate the values in columns D thru G.

The ratio between the JPEG50 and the original JPEG images is shown in column D, which indicates that on average, JPEG50 files are approximately 65% of their original size, representing an approximate size savings of 35%.

If the overall size savings for the entire page content is to be considered, the JPEG50 content needs to be compared with the original size of all content. The original JPEG images comprise only about 21% of the total content of the sample web pages as shown in column E. This JPEG50 ratio is shown in column F. The overall savings is the difference between columns E and F are shown in column G. The net overall data size savings is approximately 8% of the total web page content.

Table 11: Data Size Comparison between JPEG Size and Total Data Size

	A	B	C	D	E	F	G
Units = Bytes	Total All Files	JPEG	JPEG50	JPEG50/ JPEG	JPEG/ Total	JPEG50/ Total	Overall Savings
ABC News	245201	45081	18133	40.2%	18.4%	7.4%	11.0%
Akamai	169653	0			0.0%		0.0%
BBC	337286	75244	57152	76.0%	22.3%	16.9%	5.4%
CNET	81566	15747	10499	66.7%	19 3%	12.9%	6.4%
CNN	412545	19559	11600	59.3%	4.7%	2.8%	1.9%
Dilbert	264455	30891	17638	57.1%	11 7%	6.7%	5 0%
ESPN	149908	0					
Google	219258	48569	33518	69 0%	22.2%	15.3%	6.9%
MSNBC	258827	80525	70475	87.5%	31.1%	27.2%	3.9%
National Geo	261100	60613	48450	79.9%	23 2%	18.6%	4.7%
NASDAQ	174892	35390	17793	50.3%	20.2%	10.2%	10.1%
NYTimes	211090	0					
NOAA, 7 day	194635	22082	13521	61.2%	11.3%	6.9%	4.4%
NOAA, Aus-SA	172696	84675	54549	64.4%	49.0%	31.6%	17.4%
NOAA, Tropical	96565	37419	15634	41.8%	38.8%	16.2%	22.6%
People	109071	36504	23746	65.1%	33.5%	21.8%	11.7%
Statesman	213735	69289	42866	61.9%	32.4%	20 1%	12.4%
USAToday	196363	42300	27411	64.8%	21.5%	14.0%	7 6%
Yahoo	115598	14170	8697	61.4%	12.3%	7.5%	4.7%
Weather Ch	245771	34203	34203	100.0%	13.9%	13.9%	
Total	4130215	752261	505885				
Mean	206511	37613	29758	65.1%	21.4%	14.7%	8.0%
Std Deviation	80886	26262	18795	15.0%	12.2%	7.7%	5.7%
Min	81566	0	8697	40.2%	0.0%	2.8%	0.0%
Median	203727	35947	23746	64.4%	20.9%	14.0%	6.4%
Max	412545	84675	70475	100.0%	49.0%	31.6%	22.6%

It is important to note that these values represent comparisons between native text files and the encoded image files. The benefits of text data in a GZip format and relative size comparisons are discussed in subsequent sections.

Total Content Comparison using GZip Text Compression

A dominant encoding technique in the HTTP/1.1 standard is the GNU Zip (GZip) format (Fielding, et al. 1999). GZip encoding is utilized to compress text information, such as ASCII or Unicode. This encoding is reported to the client in the HTTP response header. GZip is not practical for use on binary files such as images.

The relative size savings realized from JPEG transcoding may be affected if the text information is encoded with GZip, a common practice on HTTP web servers. Original text size is shown in column A of Table 12. The same files compressed with GZip are shown column B. The GZip file size as compared with the original file size is shown in Column C. An average savings of nearly 80% is realized using GZip compression.

Using GZip text encoding, a new reference point for relative size comparisons is shown in column D. While the actual amount of JPEG50 transcoding size remains unchanged, the relative ratios and their apparent savings are higher. Columns E, F, and G show the updated relations, and indicate an approximate size savings of 15%.

Calculated results (not shown in tables) indicate that on average, when GZip was utilized on text data, all original image (GIF, PNG, JPEG) content represented approximately 72% of the total web content.

Table 12: Data Size Comparison between JPEG and Total Content using GZip

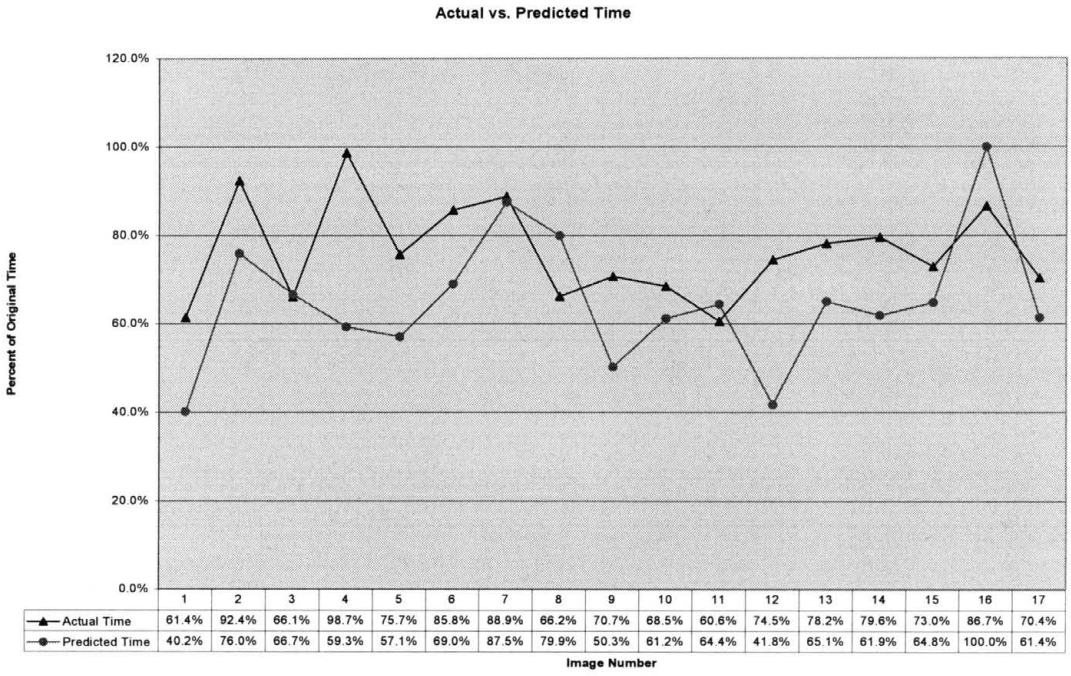
	A	B	C	D	E	F	G
Units = Bytes	Text	Text Gzip	Text Gzip/ Text	TOTAL w/ Gzip Text	JPEG/ Total GZip	JPEG50/ Total Gzip	JPEG50 + Text Gzip Savings
ABC News	177207	33358	18.8%	101352	44 5%	17 9%	26 6%
Akamai	102491	15193	14.8%	82355	0 0%	0 0%	0 0%
BBC	253584	41088	16 2%	124790	60 3%	45 8%	14 5%
CNET	51934	11149	21 5%	40781	38 6%	25 7%	12 9%
CNN	259415	48174	18 6%	201304	9 7%	5.8%	4.0%
Dilbert	193626	40612	21 0%	111441	27 7%	15 8%	11 9%
ESPN	50711	9884	19 5%	109081			
Google	148816	38064	25 6%	108506	44 8%	30 9%	13 9%
MSNBC	171474	30216	17 6%	117569	68 5%	59 9%	8 5%
National Geo	183654	40541	22 1%	117987	51 4%	41 1%	10 3%
NASDAQ	108572	26754	24 6%	93074	38 0%	19 1%	18 9%
NYTimes	179326	39892	22 2%	71656			
NOAA, 7 day	127658	30502	23.9%	97479	22 7%	13 9%	8 8%
NOAA, Aus-SA	24998	4798	19 2%	152496	55 5%	35 8%	19 8%
NOAA, Tropical	37479	5698	15 2%	64784	57 8%	24 1%	33 6%
People	47341	8529	18 0%	70259	52 0%	33 8%	18 2%
Statesman	135460	30523	22 5%	108798	63 7%	39 4%	24.3%
USAToday	129312	18094	14 0%	85145	49 7%	32 2%	17 5%
Yahoo	92370	21117	22 9%	44345	32 0%	19 6%	12.3%
Weather Ch	203606	46565	22.9%	88730	38 5%	38 5%	
Total	2679034	540751		1991932			
Mean	133952	27038	20 1%	99597	42 0%	27 7%	15 1%
Std Deviation	69682	14231	3 4%	36181	18 3%	14 9%	8 3%
Min	24998	4798	14 0%	40781	0 0%	0 0%	0.0%
Median	132386	30359	20 2%	99416	44 6%	28 3%	13 9%
Max	259415	48174	25 6%	201304	68 5%	59 9%	33 6%

Server Response Times for Original and Transcoded JPEG Images

Using the JPEG images identified in the sample websites, a performance measurement was executed using a Tomcat server across a 10 mb Linksys router. The response time for a web page containing original JPEG images was compared with the response time from a web page containing JPEG50 images. The results are shown in Table 13 and Figure 14.

Table 13: Actual vs. Predicted Time Differences for JPEG at QF50

Time Units = ms	A	B	C	D	E	F	G	H
Name	Image Count	JPEG Time	JPEG50 Time	Raw JPEG50/ JPEG Time	Adjusted JPEG Time	Adjusted JPEG Time	Actual JPEG50/ JPEG Time	Predicted JPEG50/ JPEG Time
ABC	5	143	102	70 8%	108	67	61 4%	40 2%
BBC	6	95	91	95 8%	53	49	92.4%	76 0%
CNET	1	55	39	70 4%	48	32	66 1%	66 7%
CNN	9	150	149	99 2%	87	86	98.7%	59 3%
ESPN	10	184	156	85 0%	114	86	75 7%	57 1%
GOOGLE	26	401	370	92 2%	219	188	85 8%	69 0%
MSNBC	12	238	221	92 8%	154	137	88.9%	87 5%
NATIONAL GEO	7	142	111	77 8%	93	62	66 2%	79.9%
NEW YORK TIMES	4	85	68	80 4%	57	40	70.7%	50 3%
NOAA 7 DAY	20	365	295	80 6%	225	155	68 5%	61 2%
NOAA AUSTIN SA	8	150	113	75 3%	94	57	60.6%	64 4%
NOAA HURRICANE	10	169	144	85.1%	99	74	74 5%	41 8%
PEOPLE	10	241	204	84.5%	171	134	78 2%	65 1%
STATESMAN	11	249	214	85 9%	172	137	79.6%	61 9%
USA TODAY	11	191	160	83.9%	114	83	73 0%	64 8%
WEATHER	1	98	86	87 6%	91	79	86 7%	100 0%
YAHOO	3	64	51	80 1%	43	30	70 4%	61 4%
Mean	9 0	150 3	143 8	84 0%	114 3	87 9	0 8	0 7
Std Deviation	6 4	97 5	87 7	8 1%	56 2	46 1	0 1	15 0%
Min	1 0	55 4	39 0	70.4%	43 3	30 5	0 6	40 2%
Median	9 0	150 3	143 8	84 5%	99 0	78 5	74 5%	64.4%
Max	26 0	401 1	369 9	99 2%	225 4	187 9	1.0	100 0%



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Actual Time	61.4%	92.4%	66.1%	98.7%	75.7%	85.8%	88.9%	66.2%	70.7%	68.5%	60.6%	74.5%	78.2%	79.6%	73.0%	86.7%	70.4%
Predicted Time	40.2%	76.0%	66.7%	59.3%	57.1%	69.0%	87.5%	79.9%	50.3%	61.2%	64.4%	41.8%	65.1%	61.9%	64.8%	100.0%	61.4%

Figure 14: Actual vs. Predicted Time Differences for JPEG at QF50

Columns A and B show the average response time after 30 requests for each page. Column C shows the relative time differences between columns A and B. Columns D and E indicated an adjusted value intended to compensate for propagation delay common to each request object. This adjusted value is determined by performing a *calibration request.* Multiple HTTP requests are submitted for single pixel GIF images to determine the average propagation delay for each object response. The calibration value used in this test was 7 ms per web object.

Columns F and G display the test results and the predicted values of a JPEG object collection. These columns are graphed in Figure 14. While it is apparent that there are savings to be realized through the use of JPEG transcoding, further trials would be required to identify the observed anomalies, where the actual value is less than the

predicted value. The Standard Deviation value in column D indicates a wide variance in response times, even when adjusting for propagation delay. An overall performance time savings of about 25% is noted, but this value is not weighted by the size of requests on each website.

Web Server Prototype Using a JPEG Filter

A web server prototype was implemented using an image manager process, and a web server container using a JPEG filter. The filter is designed to respond to event triggers based on memory usage or busy threads.

If an event trigger exceeds a specified value, the request is rerouted to alternate image content. Console output is displayed in following sections for 'default', 'event trigger by busy thread count', and 'event trigger by memory use' behaviors. This output demonstrates the capability of altering image requests based on performance parameters.

Console Output, Default

```
========== HTML Request ============================
ServletPath: /index.html

MEMORY Values
--------------
Free Memory.             2376992
Max Memory:              66650112
Total Memory.            6791168
Used Memory:         --> 4414176

THREAD Values
--------------
Current Thread Count.    25
Minimum Spare Threads:   25
Maximum Spare Threads:   75
Maximum Threads:         150
Threads Busy:        --> 2
```

```
---------- JPEG Request ----------
ServletPath. /image1.jpg

MEMORY Values
--------------
Free Memory              2454288
Max Memory.              66650112
Total Memory·            6791168
Used Memory:         --> 4336880

THREAD Values
--------------
Current Thread Count:    25
Minimum Spare Threads    25
Maximum Spare Threads    75
Maximum Threads:         150
Threads Busy:        --> 2

JPEG Request not modified
```

Console Output, Event Trigger by Busy Thread Count

```
========== HTML Request ==============================
ServletPath: /index html

MEMORY Values
--------------
Free Memory.             2219672
Max Memory·              66650112
Total Memory:            6791168
Used Memory.         --> 4571496

THREAD Values
--------------
Current Thread Count·    25
Minimum Spare Threads:   25
Maximum Spare Threads:   75
Maximum Threads:         150
Threads Busy.        --> 5

---------- JPEG Request ----------
ServletPath: /image1.jpg

MEMORY Values
--------------
Free Memory:             2190936
Max Memory:              66650112
Total Memory:            6791168
Used Memory:         --> 4600232

THREAD Values
--------------
Current Thread Count:    25
Minimum Spare Threads:   25
Maximum Spare Threads:   75
Maximum Threads:         150
Threads Busy.        --> 5

Busy threads exceed event trigger level of 4

JPEG Request modified to: /shadow/image1 jpg
```

Console Output, Event Trigger by Memory Use

```
========== HTML Request ============================
ServletPath· /index html

MEMORY Values
--------------
Free Memory              1501416
Max Memory.              66650112
Total Memory:            6791168
Used Memory:         --> 5289752

THREAD Values
--------------
Current Thread Count:    25
Minimum Spare Threads.   25
Maximum Spare Threads·   75
Maximum Threads.         150
Threads Busy:        --> 7


---------- JPEG Request ----------
ServletPath: /image1 jpg

MEMORY Values
--------------
Free Memory·             1477696
Max Memory.              66650112
Total Memory             6791168
Used Memory:         --> 5313472


THREAD Values
--------------
Current Thread Count:    25
Minimum Spare Threads:   25
Maximum Spare Threads:   75
Maximum Threads:         150
Threads Busy         --> 7

Used memory exceeds event trigger level of 5000000

JPEG Request modified to. /shadow/image1.jpg
```

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH

The problem that initiated this study is web content server unavailability during peak user demands, particularly during major events with public safety impact. To address this problem, this thesis studied possible performance improvement by reducing the web content data size, while conveying the intended information to the end user.

In particular, the study focused on the reduction of JPEG content data size through the use of image transcoding. The reduced data size was then used to demonstrate the capability of using unattended transcoding processes and content filters to address the problem of maintaining web server availability during peak load periods.

Conclusions

This study validates various compression values suggested by other studies or authors. Generally, a JPEG quality factor value of 50 was determined to be a value that is usable by all participants and acceptable to many, even if JPEG encoding is used on image types that are not ideal candidates for the JPEG graphic format. The survey did not consider risk factors associated with misinformation due to low compression values.

Website samples were collected from popular news and information sites. The collected information confirms that static content continues to represent a significant portion of web content. The QF value of 50 was then applied to website samples and

demonstrates a JPEG data size improvement of about 35% and a total web content size improvement of about 14%, assuming the website text content is GZip encoded. This data size savings strongly indicates that web content is not authored at an optimum level. This data indicates potential for improvement in web server availability by utilizing alternate content with reduced data size.

Performance results on the response times of original and transcoded website JPEG content indicate a time savings based on reduction of data size of about 25%. This is a preliminary value, as this value is not weighted by the relative JPEG data size for each website. Further validation is also needed to identify why actual time savings would sometimes exceed predicted time savings, even after using a response time sample size of 30.

Collectively, the study results indicate that a properly implemented image transcoding system will likely yield performance improvements. Based on the observed and calculated results, some websites may benefit greatly with image transcoding, where others may not recognize additional value, if they contain uncompressible content or are well authored from the beginning.

Constraints and Issues

The success of data size reduction is clearly affected by numerous factors. Some images cannot easily be compressed or they may already be optimized. Latency associated with the network delivery affects all content files regardless of size and the topology of the computing environment affects the success of performance improvements.

The original intent of this study was to generate lower content during peak load periods, but there may be benefit in leaving an image transcoder to improve server efficiency at all times, if efficient content authoring is not enforced.

Servers with well managed image caches that do not change content frequently will not benefit as much as a site that has frequent content change. A concern of using image transcoding to reduce the quality of images is that the low resolution content may be cached at one or more points along the HTTP response chain. Web content providers need to understand how this affects usability and acceptability, before a transcoding system is implemented. This concern may not be an issue with sites that change content frequently.

A controlled test environment is necessary to reduce the random noise that can affect the performance characteristics under inspection. Any unrelated network traffic or CPU activity may induce undesirable delay variations into the traffic under measurement which can distort the measurements.

Another concern is the balance between commercial motivations and public safety for the purpose of maintaining availability. If web content is scaled down, revenue generating content such as banner ads are reduced or eliminated, and interrupts the revenue stream of most content providers. This is no longer an issue, if no one is able to retrieve any content due to server overloads.

Future Research

There are numerous topic branches to this study that could be explored, and still remain within the general umbrella of improvement to web server availability and

performance. The following ideas highlight just a few of the areas of potential or ongoing research.

Image related research

Usability studies should be the cornerstone of any image research. Any image manipulation technique requires an understanding of the usability and acceptability of the delivered image. The process of performing a user survey can become a significant undertaking in itself, since image interpretation is very subjective based on image context and taxonomy. Usability also extends to non-human applications.

Many image attributes can affect the information perception of the user. Understanding which attributes are important and which can be effectively ignored can become an ongoing and lengthy process.

Other areas of possible image research include heuristic techniques for image identification and exploration of alternate encoding techniques, such as conversion between bitmap images and vector graphics.

Other performance considerations

Efficient content authoring at the *point of origin* is an area where significant performance improvements are possible. The overhead associated with downstream content manipulation might be avoided altogether if content is authored at maximum efficiency. If a system is designed to serve alternate content at peak server loads, the alternate content might be generated at the same time as the original content is generated.

This same idea may apply if the content is being authored for multiple hardware devices such as phones, pocket PC's, personal digital assistants (PDA), and such.

Efficient authoring needs to include HTML tags that utilize image dimensions to improve the user experience. This important approach is intended to keep the text from jumping on the screen as images render while the user attempts to read the content. The same HTML tags need to include alternate text that is displayed if the image is not available.

Event triggers used within automated processes to maintain web server availability offer significant opportunity for further research. The performance metrics for thread count and memory usage are just a small sampling of the many ways to measure server availability. Additional areas of potential study include network traffic awareness and client state awareness. For public service, a tiered or scalable event trigger system may be considered to keep the servers available at all times, possibly reducing content to basic text.

Public Safety

On November 3, 2005, the Federal Communication Commission (FCC) initiated an action to amend the Emergency Alert System (EAS) rules. This action is summarized in FCC (2005) report 05-191, which is the first step in "adopting rules that expand the reach of the EAS, as currently constituted, to cover digital communications technologies". The report specifically mentions digital communication technologies such as digital television, and direct broadcast satellite (DBS) as currently unregulated technology channels in need of further regulation.

Public comment is being sought before the measure goes to legislative rulemaking committees. As an example, public comment is solicited to identify "the type of system architecture and common protocols that would be required in such a system." The enacted legislation requires service compliance with the FCC rule changes by December 31, 2006, except for DBS which must comply by May 31, 2007.

B. Whiteaker (personal communication, November 10, 2005) notes that the Internet is not currently part of the EAS notification model. The FCC website (n.d.) also states that the agency does not regulate the Internet or the Internet service providers. However, the FCC (2005) report notes the possible influence of the Internet on the proposed EAS rule changes as it states "We also seek comment on other distribution models. For example, given its inherent robustness, we believe the Internet should serve an important role in distribution of alerts and warnings."

The general perception within the broadcast industry is that the current EAS system is in a state of ineffectiveness, and news and information channels remain the primary source of emergency information. Wenglar (2001) offers this commentary:

> Do you know what I think the modern EAS system really is? It's journalism. It is CNN, Fox News, ABC, CBS, NBC, NPR, TSN, local radio and television news teams, even pagers, cellphones and many more players. I believe that this is the true EAS system in America today.

Within the last few years, the presence of journalism has become one of the dominant sources of traffic on the Internet, shadowed only by searches and emails (Akamai n.d.).

The information blackout that immediately followed the first aircraft collision on September 11 was a catalyst for this research study. In today's technology, an information blackout has potential to be as disruptive and damaging as an electrical grid blackout. The FCC proposal should be viewed as a wake up call to the content providers that supply news and information.

It is apparent that the news and information channels failed during September 11 due to server overloads, and action is needed to avoid recurrence. An organized research effort between content providers may be the best solution to study techniques to maintain end point presence during emergency periods. A potential name for such a consortium might be the "Foghorn Project," a name that accurately represents the intent of such a collaborative project. The details of how to maintain Internet presence and availability should be made by the content providers themselves, before the decisions are made for them by regulatory agencies.

APPENDIX

JPEG IMAGE SAMPLES

To demonstrate effects of JPEG compression, samples were selected to represent portrait, landscape, line art, and map art categories and transcoded to incremental QF values. User survey samples are also provided as a reference.

The image compression samples are displayed in increments of 1 for the range of QF 1 to 9, and in increments of 10 for the range of QF 10 to original. The images are scaled to an approximate resolution of 100 pixels per inch in an attempt to simulate the resolution of a typical LCD display. Image compression samples are shown in Figure 15 thru Figure 21. The line art and map images were cropped from images used for survey questions number 3 and 13.

The user survey images are provided at the original compression level. These images are scaled to an approximate resolution of 200 pixels per inch. This scale is about 50% of the size displayed on a typical computer display. User survey images are shown in Figure 22 thru Figure 25.

(Aigner 1940)

Figure 15: JPEG Portrait Images at 100% scale

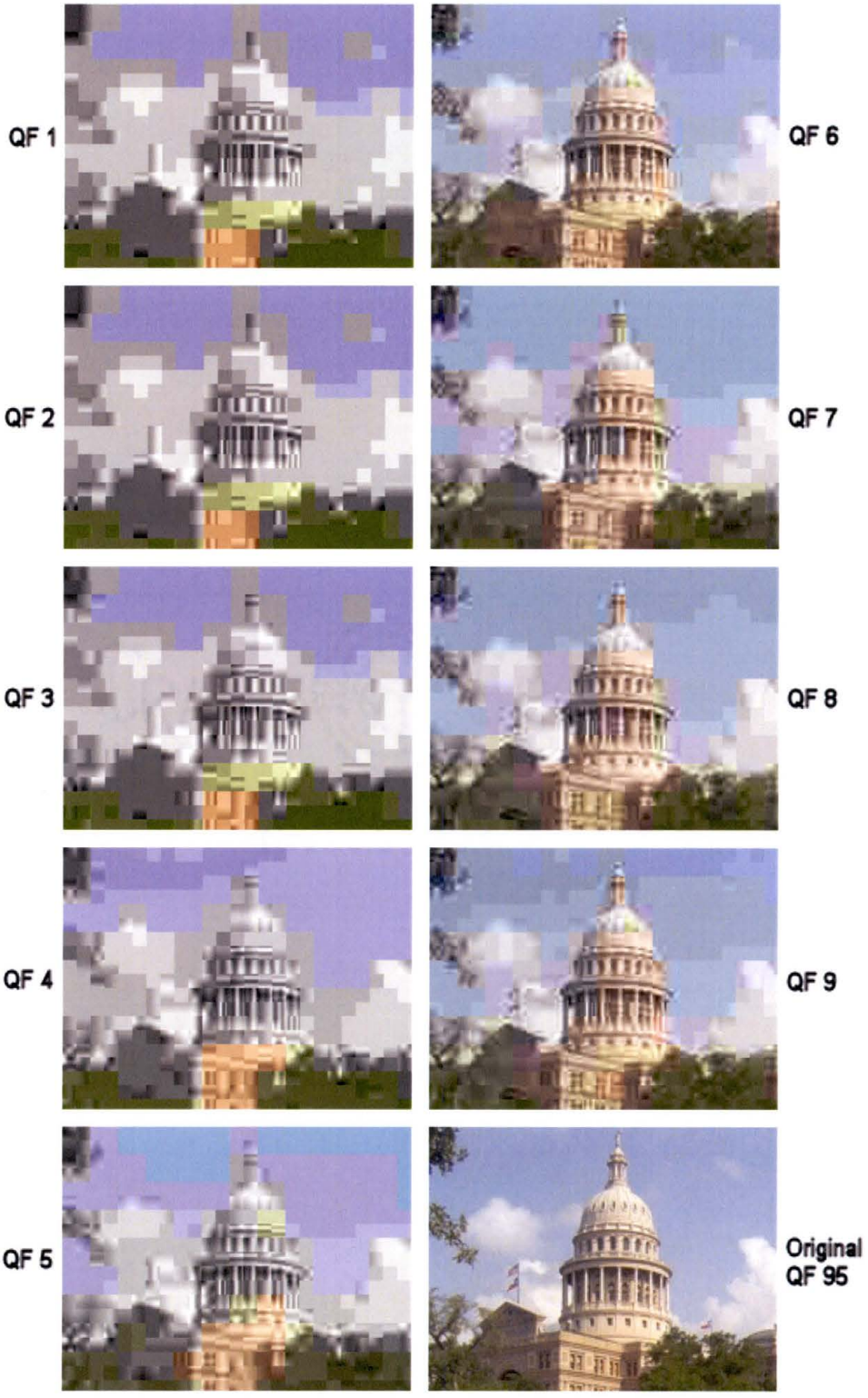


Figure 16: JPEG Landscape Images (1 of 2) at 100% scale

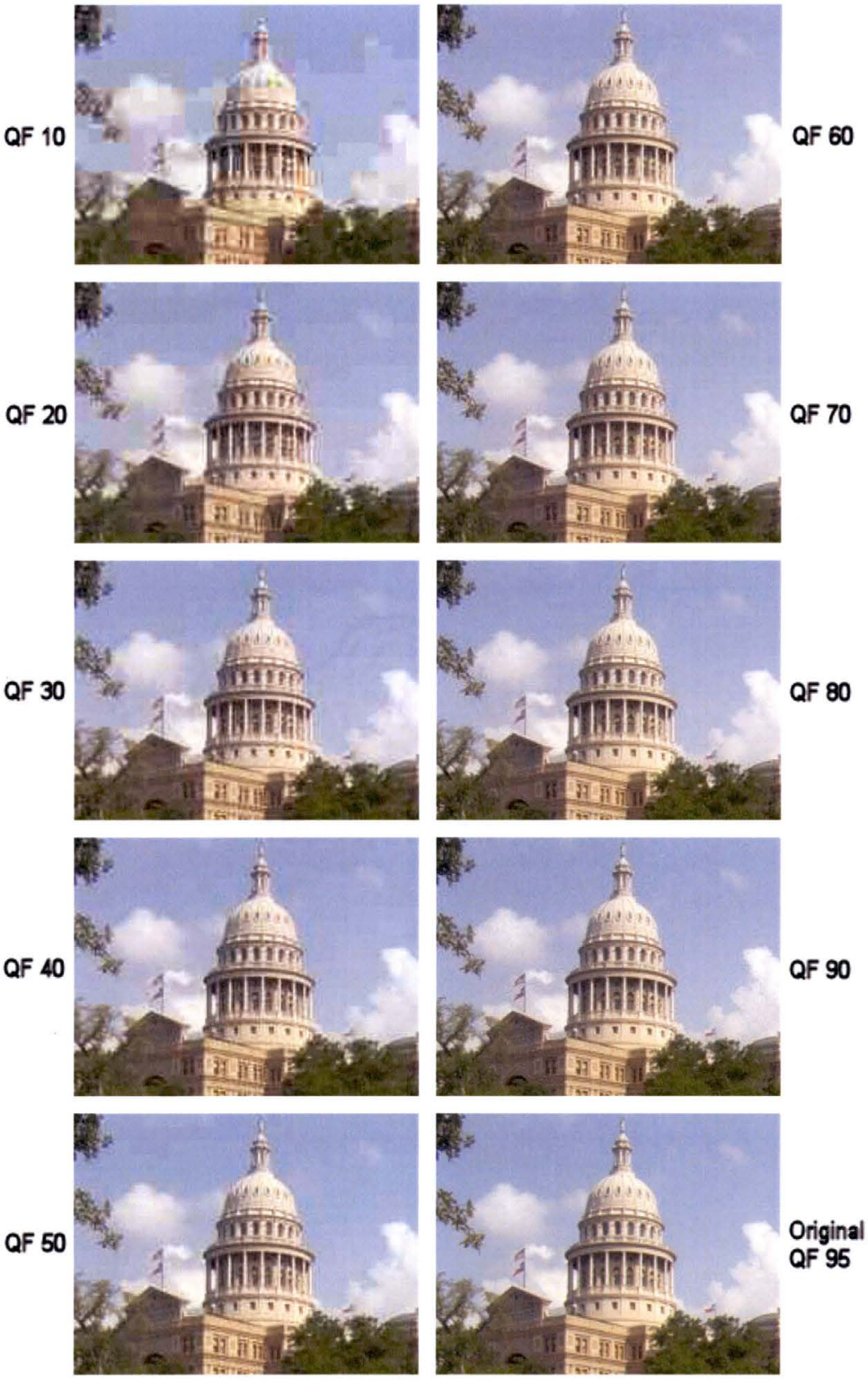
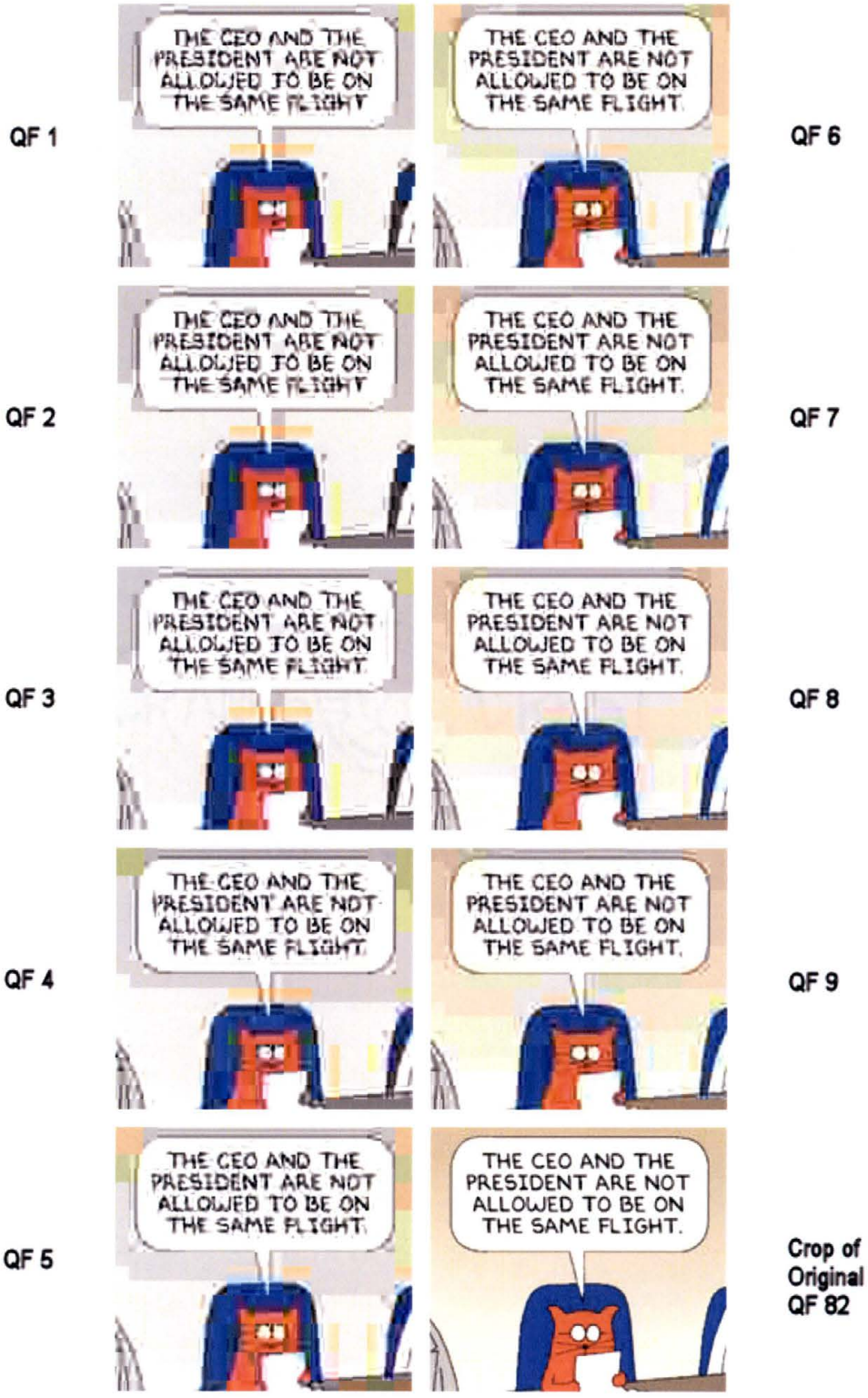
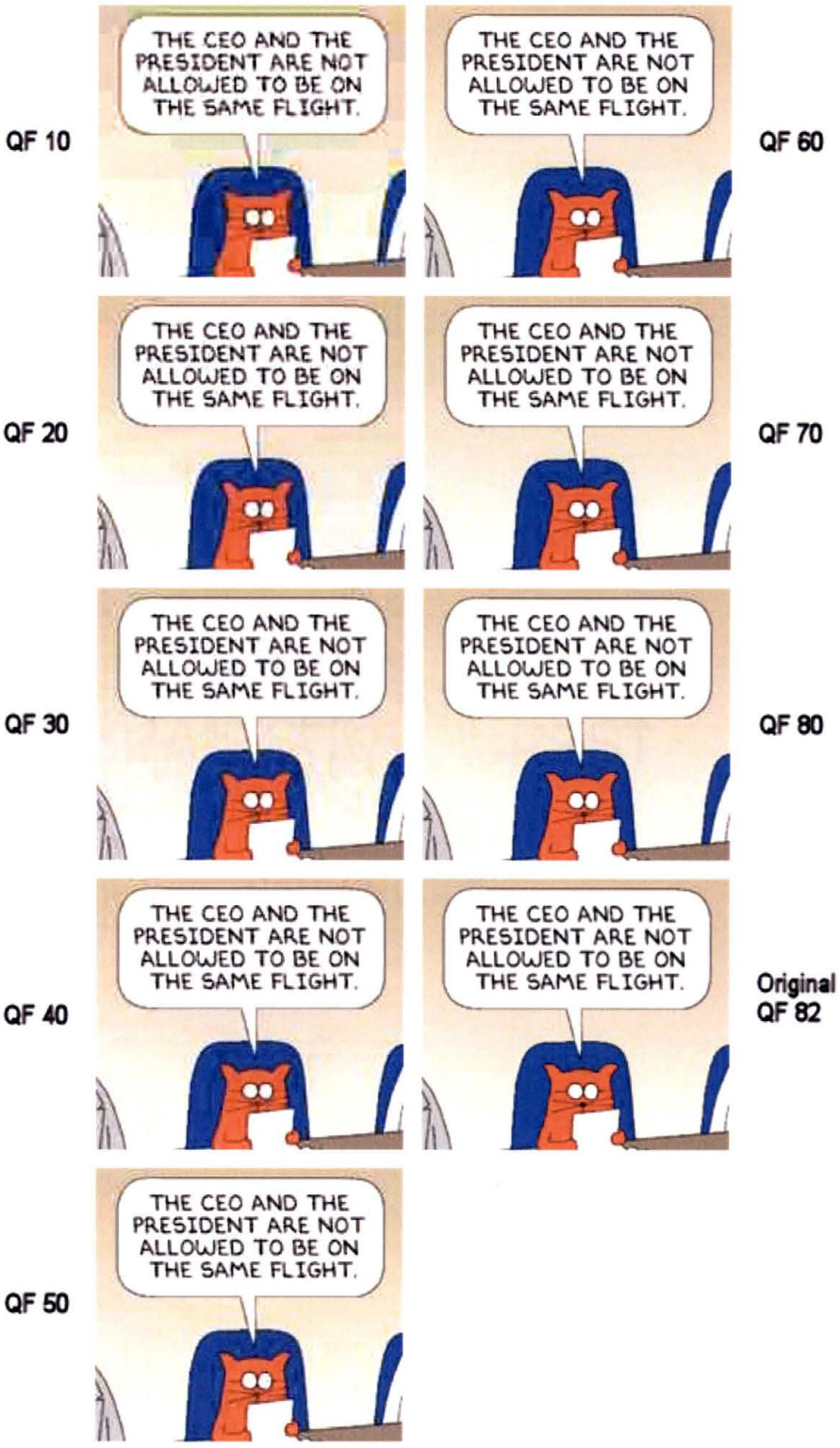


Figure 17: JPEG Landscape Images (2 of 2) at 100% scale



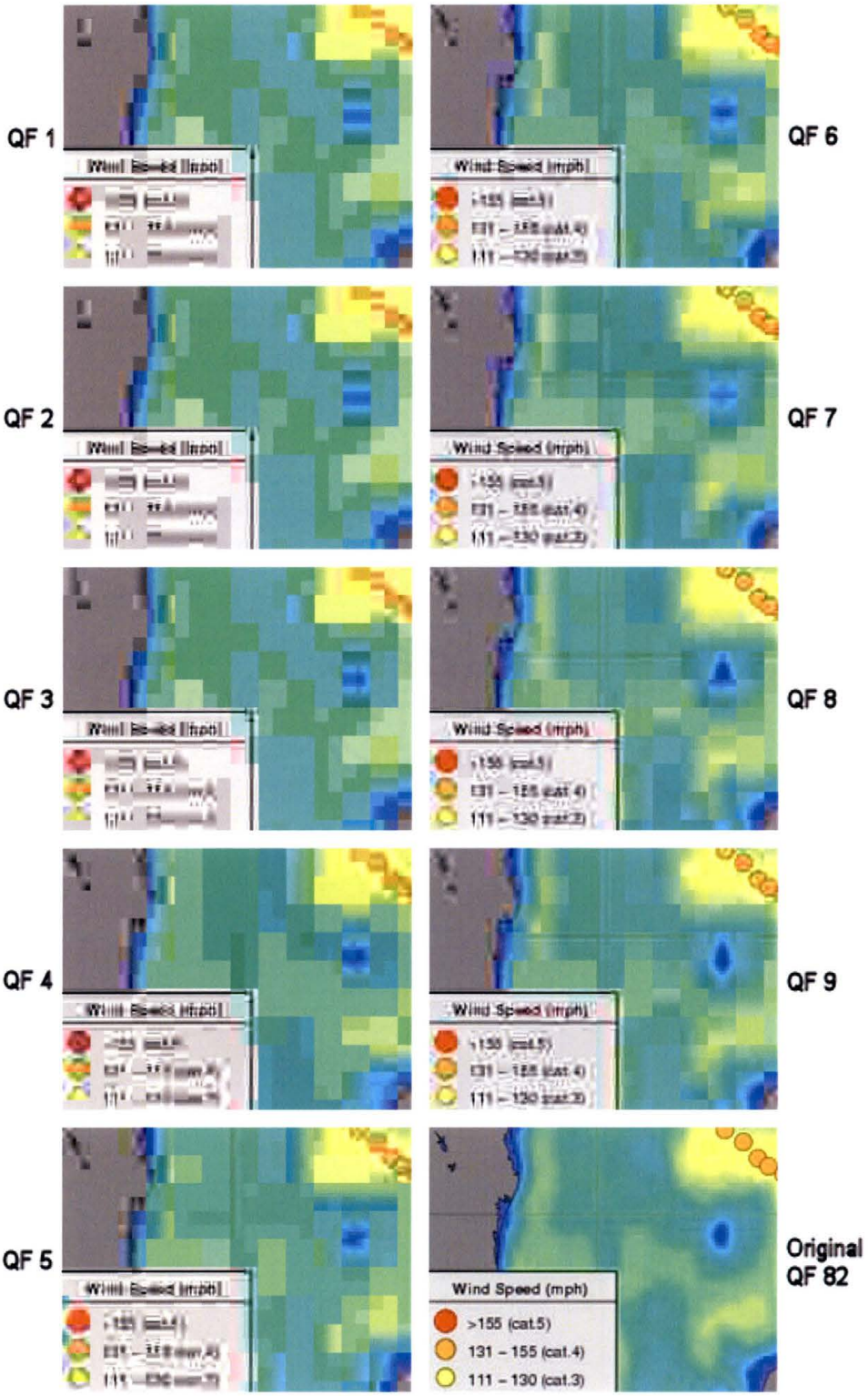
(Adams 2005)

Figure 18: JPEG Line Art Images, Cropped (1 of 2) at 100% scale



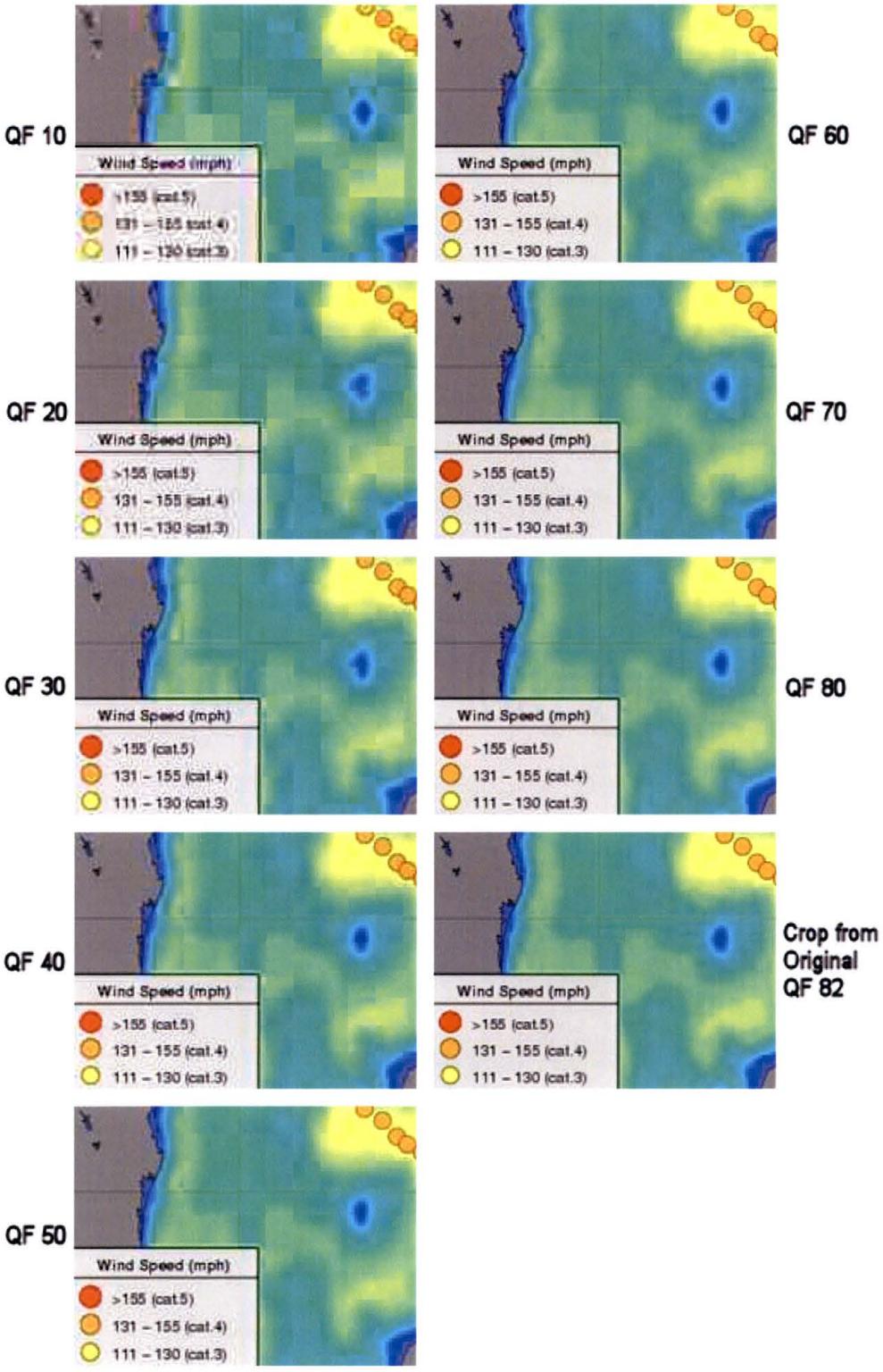
(Adams 2005)

Figure 19: JPEG Line Art Images, Cropped (2 of 2) at 100% scale



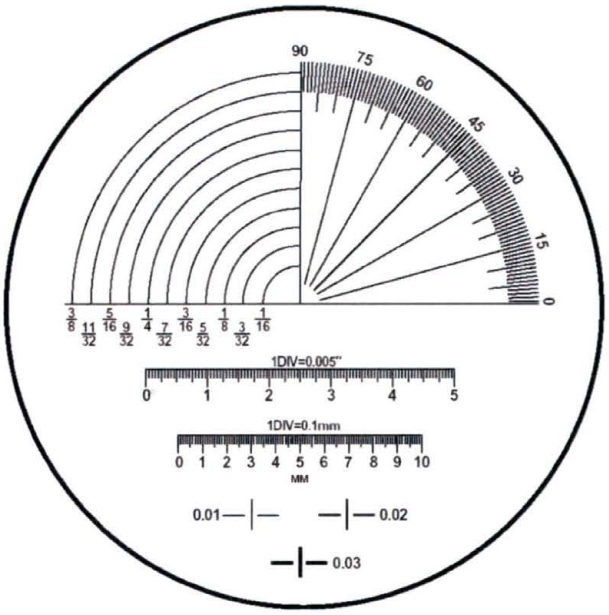
(NOAA 2005)

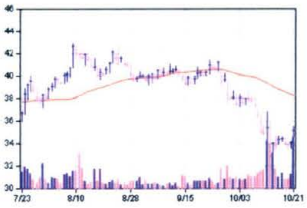
Figure 20: JPEG Map Images, Cropped (1 of 2) at 100% scale



(NOAA 2005)

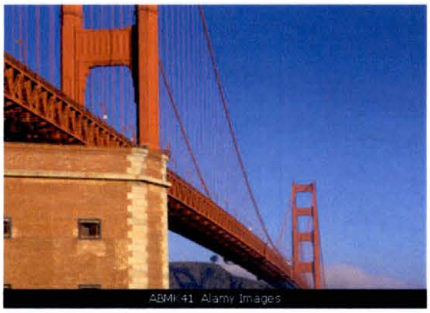
Figure 21: JPEG Map Images, Cropped (2 of 2) at 100% scale

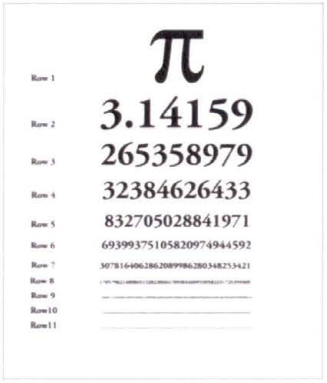


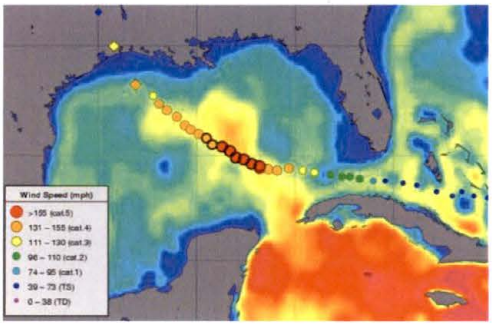


Top to bottom, Left (1, 2, 4): (Statesman.com 2005), (Alamy 2005), (Investors Business Daily 2005).
Right (5, 6, 7): (Bausch & Lomb n.d.), (NewsCom.com 2005), (EDAW n.d.)

Figure 22: User Survey Images 1, 2, 4, 5, 6, and 7 at 50% scale

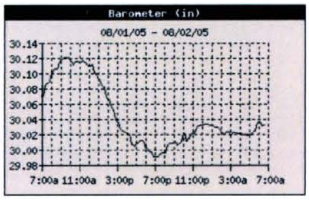


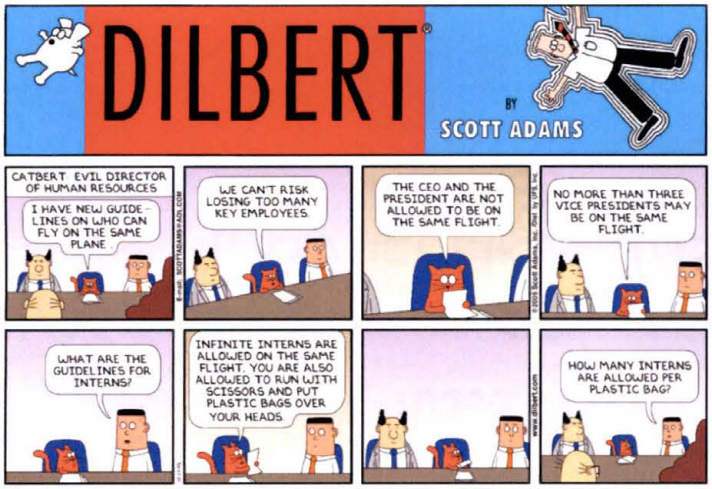


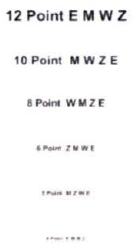


Top to bottom, Left (8, 9, 10): (Sanger 2005), (Alamy 2005), (Instructional Images 2005). **Right (11, 12, 13):** (ESPN 2005), (Alamy 2005), (NWS 2005)

Figure 23: User Survey Images 8, 9, 10, 11, 12, and 13 at 50% scale







Top to bottom, Left (14, 15, 16): (self), (Adams 2005), (NewsCom 2005). **Right (18, 19, 20):** (self), (Carey 2005), (self)

Figure 24: User Survey images 14, 15, 16, 18, 19, and 20 at 50% Scale

Top to bottom (3, 17): (MapQuest.com 2005), (Poway 2005)

Figure 25: User Survey Images 3 and 17 at 50% scale

USER SURVEY SAMPLE

This section presents the actual user survey utilized to collect usability and acceptability metrics from the user.

Name _____ Date _____

PLEASE READ INSTRUCTIONS THOROUGHLY BEFORE PROCEEDING:

Images (pictures, photos, etc.) on the internet can be significantly compressed to save file storage space. Smaller image files increase internet delivery speed. However, the consequence of compression is that the quality of the image is reduced. This survey investigates the balance between the smaller file sizes versus the lower image quality. You will be viewing a series of images ranging from the lowest quality, to the original quality and evaluating them for usability and acceptability.

There are two objectives of this survey:

1. The first objective is to determine the *minimum* level of quality required for image *usability* on the internet. An image is usable if the desired information can be obtained from the image. This survey is intended to test the information conveyed by an image, not your knowledge, so it is OKAY to skip questions.

2. The second objective is to determine the *minimum* level of quality required for image to be considered *acceptable* for use on the web, such as on a news site. For purposes of this survey, 'acceptable' is somewhere in the middle between 'poor' and 'excellent'. There are NO right or wrong answers, just your personal preferences.

NOTE: All image files have the following naming format of:

NN_QF.jpg

where **NN** is the *image number* and **QF** is the *quality factor*

Part 1: Usability

- Read the ENTIRE QUESTION and ALL ANSWERS (if provided) before proceeding.
- Under the C:\ImageSurvey\Part1-Usable folder, open the folder matching the question number, and select _START filename. (this should open with Windows Picture and Fax Viewer)
- Using the right-arrow cursor key, steadily increase the image quality until the question can be answered with reasonable confidence.
- Increase the level as fast as comfortable, but spend *no more* than 5 seconds studying any *single* image before making a decision to move forward or accept the image.
- Do NOT go back to previous images. (Once you are familiar with the material at a higher quality, the lower quality images may be misinterpreted.)
- Once the question can be answered with confidence, record your answer, and enter the QF value of the image on the Usability blank. (The acceptability blank will be used in part 2)
- DO NOT GUESS. If you are unfamiliar with the subject material, or reach the end of the images select SKIP.
- Repeat above steps for remaining questions.

Part 2: Acceptability

- Under the C:\ImageSurvey\Part2-Acceptable folder, open the folder matching the question number, and select the _START filename.
- Using the cursor keys, increase or decrease image quality until the image reaches the *minimum* quality that you consider to be acceptable. Spend no more than 10-15 seconds before making a decision.
- Record the QF value of the image the Acceptable blank.
- If you do not find ANY of the images to be minimally acceptable, write in the word NONE in the acceptability blank.
- Repeat above steps for remaining questions

01 Who is this person?

Name_____ ❍ Skip

Usable QF _____ Acceptable QF_____

02 Where was this picture taken?

❍ Hong Kong ❍ England ❍ Mexico ❍ Skip

Usable QF _____ Acceptable QF_____

03 If you begin at the star and travel Northeast on University Drive, what is the first street where you can turn LEFT?

Street Name _____ ❍ Skip

Usable QF _____ Acceptable QF_____

04 What is the Y value on 8/28 (rounded to nearest line)?

Value _____ ❍ Skip

Usable QF _____ Acceptable QF_____

05 What is the value of the fraction nearest the center of the image?

Fraction _____/_____ ❍ Skip

Usable QF _____ Acceptable QF_____

06 Who is the person waving?

❍ George Clooney ❍ George Bush ❍ Skip

Usable QF _____ Acceptable QF_____

07 Which best describes the people in the image?

❍ Construction Workers ❍ Park Visitors ❍ Skip

Usable QF _____ Acceptable QF_____

08 What is this object?

Object _____ ❍ Skip

Usable QF _____ Acceptable QF_____

09 What kind of animal is this?

❍ Walrus ❍ Seal ❍ Manatee ❍ Skip

Usable QF _____ Acceptable QF_____

10 What are the last 3 numbers on Row 7?

Value _____ ❍ Skip

Usable QF _____ Acceptable QF _____

11 This person is affiliated with which sports team?

Team_____ ❍ Skip

Usable QF _____ Acceptable QF_____

12 Which bridge is this?

Name _____ ❍ Skip

Usable QF _____ Acceptable QF_____

13 What is the wind speed range of the orange dot on the legend?

Range _____ - _____ ❍ Skip

Usable QF _____ Acceptable QF_____

14 About what time does the barometric pressure cross the 30.04 line?

❍ 8 AM ❍ 3 PM ❍ 4 AM ❍ Skip

Usable QF _____ Acceptable QF_____

15 Which employee is not allowed to be on the same plane as the CEO?

Employee _____ ❍ Skip

Usable QF _____ Acceptable QF_____

16 Who is the person on the left side of the photo?

❍ Bill Clinton ❍Alan Greenspan ❍Dick Cheney ❍ Skip

Usable QF _____ Acceptable QF_____

17 Where are the basketball courts are located?

❍Right Edge ❍Top Edge ❍Left Edge ❍ Skip

Usable QF _____ Acceptable QF_____

18 This collectible represents what product brand?

Brand_____ ❍ Skip

Usable QF _____ Acceptable QF_____

19 What kind of animal is this?

❍Donkey ❍Horse ❍Mule ❍ Skip

Usable QF _____ Acceptable QF_____

20 What is the text on the 6 point line?

Text_____ ❍ Skip

Usable QF _____ Acceptable QF_____

CODE SAMPLES

Image Survey Processor

```
################################################################################
#
# Filename: image_survey_processor sh
# Author: Michael Butterfield
# Date create. 2005 10.06
# Date modified. (new)
#
# Description:
#  This script is used to identify the current quality level of JPEG images,
#  and create a series of copies with incremental changes of quality level
#
################################################################################
#set -x

echo
echo "************************************************"
echo " BEGIN Image Survey Processor"
date
echo "************************************************"

echo,echo ==========================================================================
echo Setting and Checking prerequisites

# Directories
imageHome="c./ImageTemp"
imagickHome="C:/Program Files/ImageMagick-6.2.4-Q16"

# Program values
qualityIncrement=5
imageMax=1024

echo ------------------------------------------------------------
echo Check for content directory and Image Magick directory.
#Exit with error if either does not exist

if [ ! -e "$imageHome" ]
then
   echo Directory for $imageHome does not exist.
   exit 1
fi

if [ ! -e "$imagickHome" ]
then
   echo Directory for $imagickHome does not exist.
   exit 1
fi

echo Directory check. Passed


echo;echo ==========================================================================
echo List JPEG images in folder, and use the information to create subfolders
echo for resized and compressed images
echo ------------------------------------------------------------
```

```
fileList="$(ls "$imageHome"|grep  jpg)"
echo $fileList

for imageFileNameExt in ${fileList}
do
   echo Processing image \""$imageFileNameExt"\" using quality factor \
      of "$qualityIncrement"

   # Make directory for each file to hold set of resized or compressed images
   imageFileName=$(echo "${imageFileNameExt}"|sed 's/.jpg//')
   rm -R "${imageHome}/${imageFileName}"
   mkdir "${imageHome}/${imageFileName}"

   # Get the image quality factor
   qualityText=$("$imagickHome/identify exe"  \
     -verbose "${imageHome}"/"${imageFileNameExt}"|grep Quality)
   qualityDefaultValue=$(echo $qualityText|cut -c10-12)

   # Get the image geometry WxH
   geometry=$("$imagickHome/identify exe" \
     -verbose "$imageHome"/"$imageFileNameExt"|grep Geometry)
   geometryXValue=$(echo "$geometry"|cut -dx -f1|cut -d  -f2|cut -c2-6)
   geometryYValue=$(echo "$geometry"|cut -dx -f2)

   # find the largest dimension, and resize if necessary
   if [ "$geometryXValue" -gt "$imageMax" -o \
        "$geometryYValue" -gt "$imageMax" ]
   then
      if [ $geometryXValue -ge $geometryYValue ]
      then
         # A null will not modify the resize argument, which defaults to the X value
         dimYmodifier=""
      else
         # In this context, x means "X x Y", to build image resize command
         dimYmodifier="x"
      fi

      # Resize the image to match the max size specified
      "${imagickHome}/convert.exe" "${imageHome}/${imageFileNameExt}" \
        -resize "$dimY$imageMax"  \
        "$imageHome/$imageFileName/$imageFileName_resize$dimY$imageMax.jpg"

      # Use the resized image for the remainder of the code block
      imageFileNameExt="$imageFileName/$imageFileName_resize$dimY$imageMax.jpg"

   fi


   # Round quality target value to nearest multiple of quality increment value
   qualityTargetValue=$((qualityDefaultValue - \
     (qualityDefaultValue % qualityIncrement)))

   while [ "$qualityTargetValue" -gt 0 ]
   do
      # Syntax. convert <source_image> -quality=<value> <target_image>
      "$imagickHome/convert.exe" \
        "$imageHome"/"$imageFileNameExt" -quality "$qualityTargetValue" \
        "$imageHome"/"$imageFileName"/"$imageFileName"_$qualityTargetValue.jpg

      qualityTargetValue=$(( qualityTargetValue - qualityIncrement ))
   done
done

echo
echo "************************************************"
echo " END Image Survey Processor"
date
echo "************************************************"
# End of file
```

Image Manager

```
################################################################################
#
# Filename· image_manager.sh
# Author: Michael Butterfield
# Date create: 2005 11.04
# Date modified  (new)
#
# Description
#  This script monitors a content directory tree for JPEG images. For each image,
#  a matching shadow directory tree is checked for a matching image  If the image
#  does not exist, the image is transcoded, then placed in the shadow directory
#  tree.
#
#  The shadow directory tree is also monitored for images that do not exist in
#  the content directory tree. For each image, if the image does not exist in the
#  content directory tree, it is removed from the shadow directory tree.
#
################################################################################
#set -x

echo
echo "************************************************"
echo "BEGIN Image Manager";date
echo "************************************************"

echo,echo ========================================================================
echo Setting and Checking prerequisites

# Directories
imagickHome="C./Program Files/ImageMagick-6.2.4-Q16"
#contentHome=${deployHome}/myContextRoot
#shadowHome=${deployHome}/myContextRoot_shadow
deployHome=c:/jakarta-tomcat-5.5 9/webapps
contentHome="${deployHome}/jgadget/sampleSites"
shadowHome="${deployHome}/jgadget/sampleSites_shadow"

# Program control values
scanTimer=10
deleteTimer=120

# Set to 0 to copy original (filters JPEGS for original size reference)
imageQuality=50

# Tell the use what is happening
echo imagickHome is "$imagickHome"
echo deployHome is "$deployHome"
echo contentHome is "$contentHome"
echo shadowHome is "$shadowHome"


echo;echo ------------------------------------------------------------------
echo Check for content directory and Image Magick directory.
echo
#Exit with error if either does not exist


if [ ! -e "$imagickHome" ]
then
   echo Directory for $imagickHome does not exist.
   exit 1
fi

if [ ! -e "$contentHome" ]
```

```
then

   echo Directory for $contentHome does not exist.
   exit 1
fi

echo Directory check  Passed


echo,echo ------------------------------------------------------------------
echo See if shadow directory exists. If not, make directory,echo

# Check to see if files exist in shadow directory  If not, process the file
# and store as image or link (depending on if image is compressible, based
# on quality watermark)

if [ -d $shadowHome ]
then
   echo Shadow Directory already exists· "$shadowHome"
else
      mkdir $shadowHome
      echo Shadow Directory created: "$shadowHome"
fi


#loop forever, Ctrl-C to stop program
while true
do

   echo;echo ------------------------------------------------------------------
   echo START of Primary timed loop
   date +%r
   echo

   # This doesn't work. It gets word tokens, not entire line (filename):
   # originalFileList="$(find $contentHome -name "*.jpg")"
   # for originalPathFile in ${originalFileList}

   find $contentHome -name "*.jpg">filelist1.txt

   while read originalPathFile
   do

      #echo "$originalPathFile"

      #Trim off the contentHome
      trimmedPathFile=$(echo "$originalPathFile"|sed s^"$contentHome"^""^)

      #Splice on the shadowHome
      shadowPathFile="$shadowHome""$trimmedPathFile"


      if [ -f "$shadowPathFile" ]
      then
        #echo File "${shadowPathFile}" already exists
        continue
      fi

      #File does not exist in shadow. Check if directory exists.
      #Chop off the file name
      trimmedPath=$(echo "$trimmedPathFile"|sed -e 's/\/[^/]*$//')

      #Splice onto the shadowHome
       shadowPath="$shadowHome""$trimmedPath"

      #Now you have the absolute path
      if [ ! -d "$shadowPath" ]
      then
         echo;echo MAKE DIRECTORY  "$shadowPath"
```

```
         mkdir -p "$shadowPath"
      fi

      #Finally, create file or link

      if [ ! "$imageQuality" -eq 0 ]
      then
        echo TRANSCODE FILE· "$shadowPathFile"
        "$imagickHome"/convert "$originalPathFile" \
          -quality "$imageQuality"  "$shadowPathFile"
      else
        echo "COPY (QF is 0):" "$shadowPathFile"
        cp "$originalPathFile" "$shadowPathFile"
      fi

   done < filelist1.txt
   rm filelist1 txt

   find $shadowHome -name "*.jpg" > filelist2.txt

   while read shadowPathFile
   do

      #Check if file already exists. Continue loop, if it does

      #Trim off the shadowHome
      trimmedPathFile=$(echo "$shadowPathFile"|sed s^"$shadowHome"^""^)

      #Concatenate the shadowHome
      originalPathFile="$contentHome""$trimmedPathFile"
      if [ ! -f "$originalPathFile" ]
      then
         echo REMOVE FILE   : "$shadowPathFile"
         rm "$shadowPathFile"
      fi
   done < filelist2.txt
   rm filelist2.txt


   #-----------------------------------------------------------------
   # Future work: prune empty shadow directories

   sleep "$scanTimer"

done


echo
echo "**************************************************"
echo " END Image Manager"
echo "**************************************************"
# End of file
```

Tomcat Web Server JPEG Filter

web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
  Copyright 2004 The Apache Software Foundation

  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>

    <display-name>JGadget Samples</display-name>
    <description>
      JGadget Samples
    </description>

    <!-- Define servlet-mapped and path-mapped example filters -->
    <filter>
        <filter-name>Servlet Mapped Filter</filter-name>
        <filter-class>filters.ExampleFilter</filter-class>
        <init-param>
            <param-name>attribute</param-name>
            <param-value>filters.ExampleFilter.SERVLET_MAPPED</param-value>
        </init-param>
    </filter>

    <filter>
        <filter-name>Path Mapped Filter</filter-name>
        <filter-class>filters.ExampleFilter</filter-class>
        <init-param>
            <param-name>attribute</param-name>
            <param-value>filters.ExampleFilter.PATH_MAPPED</param-value>
        </init-param>
    </filter>

    <filter>
        <filter-name>JPEG Filter</filter-name>
        <filter-class>filters JPEGFilter</filter-class>
        <init-param>
            <param-name>attribute</param-name>
            <param-value>filters.JPEGFilter.PATH_MAPPED</param-value>
        </init-param>
    </filter>

    <filter>
        <filter-name>Request Dumper Filter</filter-name>
        <filter-class>filters RequestDumperFilter</filter-class>
    </filter>
```

```
<!   Example filter to set character encoding on each request   >
<filter>
    <filter-name>Set Character Encoding</filter-name>
    <filter-class>filters.SetCharacterEncodingFilter</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>EUC_JP</param-value>
    </init-param>
</filter>


<!-- Define filter mappings for the defined filters -->
<filter-mapping>
    <filter-name>Servlet Mapped Filter</filter-name>
   <servlet-name>invoker</servlet-name>
</filter-mapping>

<filter-mapping>
    <filter-name>Path Mapped Filter</filter-name>
   <url-pattern>/servlet/*</url-pattern>
</filter-mapping>

<filter-mapping>
    <filter-name>JPEG Filter</filter-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>


<!-- Define example application events listeners -->
<listener>
    <listener-class>listeners ContextListener</listener-class>
</listener>
<listener>
    <listener-class>listeners.SessionListener</listener-class>
</listener>

<!-- Define servlets that are included in the example application -->

<servlet>
    <servlet-name>JGadgetExample</servlet-name>
    <servlet-class>JGadgetExample</servlet-class>
</servlet>


<servlet-mapping>
    <servlet-name>JGadgetExample</servlet-name>
    <url-pattern>/servlet/JGadgetExample</url-pattern>
</servlet-mapping>


<security-constraint>
  <display-name>Example Security Constraint</display-name>
  <web-resource-collection>
     <web-resource-name>Protected Area</web-resource-name>
     <!-- Define the context-relative URL(s) to be protected -->
     <url-pattern>/jsp/security/protected/*</url-pattern>
     <!-- If you list http methods, only those methods are protected -->
     <http-method>DELETE</http-method>
     <http-method>GET</http-method>
     <http-method>POST</http-method>
     <http-method>PUT</http-method>
  </web-resource-collection>
  <auth-constraint>
     <!-- Anyone with one of the listed roles may access this area -->
     <role-name>tomcat</role-name>
     <role-name>role1</role-name>
  </auth-constraint>
</security-constraint>
```

```
    <!-- Default login configuration uses form-based authentication -->
    <login-config>
      <auth-method>FORM</auth-method>
      <realm-name>Example Form-Based Authentication Area</realm-name>
      <form-login-config>
        <form-login-page>/jsp/security/protected/login.jsp</form-login-page>
        <form-error-page>/jsp/security/protected/error.jsp</form-error-page>
      </form-login-config>
    </login-config>

    <!-- Security roles referenced by this web application -->
    <security-role>
      <role-name>role1</role-name>
    </security-role>
    <security-role>
      <role-name>tomcat</role-name>
    </security-role>

</web-app>
```

JPEGFilter.java

```
/*
* Copyright 2004 The Apache Software Foundation
*
* Licensed under the Apache License, Version 2.0 (the "License"),
* you may not use this file except in compliance with the License
* You may obtain a copy of the License at
*
*     http //www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/

package filters;

import java io.IOException;
import javax.servlet Filter;
import javax.servlet.FilterChain;
import javax servlet FilterConfig;
import javax.servlet.ServletException,
import javax.servlet.ServletRequest;
import javax servlet.ServletResponse;

//-----------------------------------------------------------------Added by Mike
import javax.servlet.RequestDispatcher;
import javax.servlet.http.HttpServletRequest;
import javax.servlet http HttpSession;
import javax.management.MBeanServer;
import javax.management.ObjectName;
import org.apache.commons.modeler.Registry;


/**
 * This filter intercepts all content requests and processes requests which
 * include JPEG images·
 * <ul>
 * <li>Identifies current thread and memory usage</li>
 * <li>Intercepts requests with a  jpg suffix</li>
 * <li>If a thread count or memory value is exceeded, the request is diverted
 * to an alternate file repository</li>
```

```
 * </ul>
 * <ul>
 * <li>Prototype assumtions (should be changed for actual use) </li>
 *  <li>* Compressed image already exists in shadow directory</li>
 *  <li>*  Values are hardcoded</li>
 *  <li>*  Only ".jpg" file extensions are captured, others ignored</li>
 * </ul>
 *
 * @author Michael Butterfield
 * @version $Revision: 0.1 $ $Date: 2005.11.21 $
 */

public final class JPEGFilter implements Filter {

    // ----------------------------------------------------- Instance Variables

    /**
     * The request attribute name under which we store a reference to ourself.
     */
    private String attribute = null;

    /**
     * The filter configuration object we are associated with   If this value
     * is null, this filter instance is not currently configured.
     */
    private FilterConfig filterConfig = null;

    //-------------------------------------------------------------Added by Mike
    //private Object memoryUsage = null,
        private String modifiedServletPath = null,
        private RequestDispatcher jpegRequestDispatcher = null;

        private boolean alternateJPEGContent = false,

        private int threadsBusy = 0;
        private int threadCount = 0;

        private long freeMemory = 0,
        private long maxMemory = 0;
        private long totalMemory = 0;
        private long usedMemory = 0;
        private long processors = 0;



        //------------------------------ Hardcoded Values (for demo purposes only)
        private String threadPoolObject =
                "Catalina.type=ThreadPool,name=http-8080";
        private String shadowDirectory = "/shadow";
        private int eventTriggerThread = 10;
        private long eventTriggerMemory = 5000000,



    // --------------------------------------------------------- Public Methods

    /**
     * Take this filter out of service.
     */
    public void destroy() {
        this.attribute = null;
        this.filterConfig = null;
    }

    public void doFilter(ServletRequest request, ServletResponse response,
                         FilterChain chain)
        throws IOException, ServletException {
```

```
//Write to filter log
    filterConfig getServletContext() log("JPEGFilter activated"),

    //------------------------------------------- RESET parameters each call
    jpegRequestDispatcher = null;
    modifiedServletPath = null;
    alternateJPEGContent = false,

    //Cast a request to an HTTPServlet request
    HttpServletRequest hr = (HttpServletRequest) request;
    HttpSession session = hr.getSession();
    String servletPath = hr.getServletPath();       //returns /image1.jpg

    // Draw a line on the console when a new request arrives
        if(servletPath.indexOf(".htm") > 0)
                System.out println(
                  "\n========== HTML Request ============================"),

        if(servletPath.indexOf(".jpg") > 0)
                System out println(
                  "\n---------- JPEG Request ----------"),

    System.out.println("ServletPath: " + servletPath),

    // Store ourselves as a request attribute (if requested)
    if (attribute != null)
        request.setAttribute(attribute, this);

            //-------------------------------------------Examine Performance Metrics
    //MEMORY values
    maxMemory = Runtime.getRuntime().maxMemory(),
            totalMemory = Runtime.getRuntime().totalMemory();
    freeMemory = Runtime getRuntime().freeMemory();
    processors = Runtime.getRuntime().availableProcessors();
    usedMemory = totalMemory - freeMemory;

    System.out.println("\nMEMORY Values"),
    System.out.println(  "--------------"),
    System.out.println("Free Memory:              " + freeMemory);
    System.out.println("Max Memory:               " + maxMemory);
    System.out.println("Total Memory:             " + totalMemory),
    System.out.println("Used Memory:         --> " + usedMemory);

    //System.out.println("Processors:   " + processors);

    //THREAD attributes
    MBeanServer mBeanServer =
                Registry.getRegistry(null, null) getMBeanServer(),

        try {
                ObjectName objectName =
                        new ObjectName(threadPoolObject);

                Integer iThreadCount = (Integer)
                mBeanServer.getAttribute(objectName, "currentThreadCount");
                threadCount = iThreadCount.intValue(),

                Integer iThreadsBusy = (Integer)
                mBeanServer.getAttribute(objectName, "currentThreadsBusy");
                threadsBusy = iThreadsBusy.intValue(),



            System.out.println("\nTHREAD Values ");
            System.out.println(  "--------------");

            System.out.println("Current Thread Count·    "
                + mBeanServer.getAttribute(objectName, "currentThreadCount"));
```

```
        System out.println("Minimum Spare Threads·    "
            + mBeanServer getAttribute(objectName, "minSpareThreads")),
        System.out.println("Maximum Spare Threads·    "
            + mBeanServer.getAttribute(objectName, "maxSpareThreads"));
         System.out.println("Maximum Threads:          "
            + mBeanServer.getAttribute(objectName, "maxThreads"));
          System out println("Threads Busy:         --> "
            + mBeanServer.getAttribute(objectName, "currentThreadsBusy"));


    } catch (Exception e) {
    e printStackTrace(),
}


    //Process JPEG images
    if(servletPath indexOf(" jpg") > 0)
    {
            //Check memory usage
            if(usedMemory > eventTriggerMemory)
            {
            System out.println("\nUsed memory exceeds event trigger level of "
                                + eventTriggerMemory),
                    alternateJPEGContent = true;
            }

            //Check threads busy
            if(threadsBusy >= eventTriggerThread)
            {
                    System.out.println(
                            "\nBusy threads exceed event trigger level of "
                            + eventTriggerThread);
                    alternateJPEGContent = true;
            }

            //Alter request if memory usage or thread busy thresholds exceeded
            if( alternateJPEGContent )
            {

                    modifiedServletPath =
                    servletPath replace(
                            servletPath,shadowDirectory + servletPath),

                    System.out.println(
                            "\nJPEG Request modified to: "
                            + modifiedServletPath);
            }
            else
            {
                    System out.println("\nJPEG Request not modified");
            }

            jpegRequestDispatcher =
                    hr.getRequestDispatcher(modifiedServletPath);


            if (jpegRequestDispatcher != null)
                    jpegRequestDispatcher.forward(request, response);
    }


    // Time and log the subsequent processing
    long startTime = System.currentTimeMillis();

    chain.doFilter(request, response);

    long stopTime = System.currentTimeMillis();

}
```

```
    /**
     * Place this filter into service
     *
     * @param filterConfig The filter configuration object
     */
    public void init(FilterConfig filterConfig) throws ServletException {

        this.filterConfig = filterConfig,
         this attribute = filterConfig getInitParameter("attribute"),

    }


    /**
     * Return a String representation of this object.
     */
    public String toString() {

        if (filterConfig == null)
            return ("InvokerFilter()");
        StringBuffer sb = new StringBuffer("InvokerFilter(");
        sb.append(filterConfig);
        sb append(")");
        return (sb.toString());

    }
}
```

BIBLIOGRAPHY

Adams, Scott. (2005, October 23) *Dilbert*. United Feature Syndicate. Retrieved 10/30/2005 from http://www.dilbert.com

Aigner, L. (1940). Albert Einstein: Physicist and Pacifist. *Portraits of Character from the National Portrait Gallery.* Smithsonian, NPG.79.251 Retrieved 11/20/2005 from http://www.npg.si.edu/educate2/ein2.htm

Akamai. (n.d.) Why is this information important? *Frequently Asked Questions.* Retrieved 10/29/2005 from http://www.akamai.com/en/html/industry/net_usage/faq.html

Akamai. (2001, September) Akamai Technologies Mourns the Loss of Co-Founder and CTO Daniel Lewin. *Press Release.* Retrieved 10/29/2005 from http://www.akamai.com/en/html/about/press/press292.html

Associated Press (AP), Franklin, Frank (n.d.). Trump earns $25,000 per minute for talk. *Austin American Statesman.* Retrieved 10/25/2005 from http://www.statesman.com

Bausch & Lomb (n.d.) Hastings (Bausch & Lomb) general purpose scale 81-34-36. *Bausch & Lomb.* Similar image available at http://www.fairfieldny.com

Bellovin, S. (2001, September 11). cnn.com cuts graphics. *Searchable NANOG Mail Archives.* Retrieved 10/29/2005 from http://www.cctec.com/maillists/nanog/historical/0109/maillist.html

Bodoff, S., Armstrong, E., Ball, J., Carson, D., Evans, I., Green, D., et al. (2004) *The J2EE Tutorial, Second Edition.* Addison-Wesley. ISBN 0-321-24575-X.

Caray, A., Caray, S. (n.d.) Quarter horse running. *Horses.* Retrieved 10/21/2005 from http://www.alanandsandycarey.com/Domestic/Horses

Cardellini, V., Casalicchio, E. (2002). The State of the Art in Locally Distributed Web-Server Systems. *ACM Computing Surveys*, Vol. 32, No. 2, June 2002, pp. 263-311.

CompuServe, Inc. (1990). *Graphics Interchange Format (sm) Version 89a.* Retrieved 10/29/2005 from http://www.dcs.ed.ac.uk/home/mxr/gfx/2d/GIF89a.txt

Curran, K., Duffy, C. (2005). Understanding and reducing web delays. *International Journal of Network Management* 2005, 15: pp. 89-102.

Donelan, S. (2001, October). September 11: What Worked, What Didn't. *North American Network Operators Group (NANOG) Meeting Index.* Retrieved 10/29/2005 from http://www.nanog.org/mtg-0110/donelan.html

EDAW (n.d.) Costal Maine botanic garden master plan. *Landscape Architecture.* Retrieved on 10/30/2005 from http://www.sketchup.com/?sid=24

ESPN (2005). Joe Torre. *ESPN Home Page.* Retrieved on 10/16/2005 from http://espn.go.com

Eubanks, M. (2001, October). Multicast after 9/11/2001. *North American Network Operators Group (NANOG) Meeting Index.* Retrieved 10/29/2005 from http://www.nanog.org/mtg-0110/eubanks.html

Federal Communications Commission (2005, November 10). First report and order and further notice of proposed rule making. *In the matter of review of the emergency alert system.* Document ID FCC 05-191. Retrieved 11/10/2005 from http://www.fcc.gov

Federal Communications Commission (n.d.). Internet. *Consumer & Governmental Affairs Bureau.* Retrieved 11/12/2005 from http://www.fcc.gov/cgb/internet.html

Fielding, R., Irvine, U.C., Gettys, J., Compaq/W3C, Mogul, J., Compaq, Frystyk, H. W3C/MIT, et al. (1999, June). *Hypertext Transfer Protocol – HTTP/1.1.* Retrieved 10/29/2005 from http://www.w3.org/Protocols/rfc2616/rfc2616.html

Freedman, A. (2001, September 11). Horrible world trade center crash, traffic effect. *Searchable NANOG Mail Archives.* Retrieved 10/29/2005 from http://www.cctec.com/maillists/nanog/historical/0109/maillist.html

Instructional Images (n.d.). Pi eye chart, item 42204. *Instructional Images.* Retrieved on 11/02/2005 from http://catalog.instructionalimages.com

Intel. (n.d.). Moore's Law. *Intel museum.* Retrieved 11/17/2005 from http://www.intel.com/museum/archives/history_docs/mooreslaw.htm

Investors Business Daily. (2005, 10/21). Advanced medical optics (symbol: EYE) chart. *Investors Business Daily.* Retrieved on 10/21/2005 from http://investors.com

Jakarta. (n.d.). *Apache JMeter*. Retrieved 11/15/2005 from http://jakarta.apache.org/jmeter/index.html

Kherfi, M.L, Ziou, D. (2004). Image retrieval from the World Wide Web: issues, techniques, and systems. *ACM Computing Surveys*, Vol. 36. No. 1, March 2004, pp. 35-67

Knutsson, B., Lu, H., Mogul, J., Hopkins, B. (2003). Architecture and performance of server-directed transcoding. *ACM Transactions on Internet Technology*, Vol. 2, No. 4, November 2003, pp 392-424.

Kochan, S, Wood, P. (2005) *UNIX Shell Programming, Third Edition*. Indianapolis, IN: Sams Publishing.

Lane, T. (1999, March 29). *JPEG image compression FAQ*. Retrieved 10/29/2005 from http://www.faqs.org/faqs/jpeg-faq.

Li, W., Hsiuing, W., Po, O., Hino, K., Candan, K., Agrawal, D. (2004, May 17-22). Challenges and practices in deploying web acceleration solutions for distributed enterprise systems. *WWW204*, May 17-22, 2004 New York, New York, USA. ACM 1-58113-844-X/04/2005.

Mapquest (2005). Maps. *Mapquest*. Get Map for 500 University Drive, San Marcos, Tx. Retrieved 10/2/2005 from www.mapquest.com

Mikula, D. (2001, September 11). Re: Horrible world trade center crash, traffic effect. *Searchable NANOG Mail Archives*. Retrieved 10/29/2005 from http://www.cctec.com/maillists/nanog/historical/0109/maillist.html

Mogul, J. C., Douglis, F., Feldmann, A., Krishnamurthy, B. (1997). Potential benefits of delta encoding and data compression for HTTP. *SIGCOMM '97* Cannes, France ACM 0-89791-905-X/97/0009.

Moore, G. (1965, April 19). Cramming more components onto integrated circuits. *Electronics*. Volume 38, Number 8, April 19, 1965.

Muller, N. (1998). Improving and managing multimedia performance over TCP/IP nets. *International Journal of Network Management*. John Wiley & Sons, 8, pp 356-367.

Nakano, T., Harumoto, K, Shimojo, S., Nishio, S. (2002). User adaptive content delivery mechanism on the World Wide Web. *ACM*. SAC 2002, Madrid, Spain ACM 1-58113-445-2/02/03, pp. 1140-1146.

National Weather Service (NWS) (2005, 09/23). TAFB Atlantic Forecasts and Analyses: Sea Surface Temperature Analyses. *National Hurricane Center.* Retrieved 9/23/2005 from http://www.nhc.noaa.gov/tafb-atl.shtml

Poway Unified School District (n.d.). Landscape concept plan-overall site. *Building for Success Program.* Retrieved 10/21/2005 from http://powayusd.sdcoe.k12.ca.us/Bond/OurSchools_html/westwood_es.htm

Sanger, David (n.d.) Travel stock photography & assignments worldwide. *David Sanger Photography.* Retrieved 11/03/2005 from http://www.davidsanger.com

Sun. (2004). JSR-000154 Java servlet 2.4 specification (final release). *Sun Developer Network.* Retrieved 10/29/2005 from http://java.sun.com/products/servlet/download.html

Sun. (2005). The essentials of filters. *Sun Developer Network.* Retrieved 10/29/2005 from http://java.sun.com/products/servlet/Filters.html.

Tufte, E. (1983). *Visual Display of Quantitative Information.* Cheshire, CT: Graphics Press.

W3C. (1999, December). *HTML 4.01 Specification.* Retrieved 10/29/2005 from http://www.w3.org/TR/html401/html40.pdf.gz

Wallace, G.K. (1991, April). The JPEG still picture compression algorithm. *Communications of the ACM.* April 1991/Vol.34, No. 4.

Wan, R., Moffat, A. (2001). Effective compression for the web: exploiting document linkages. *IEEE* 1530-09 19/01 pp. 68-75.

Wenglar, Mike. (2001, October). The EAS: is it worth it? *Chapter 79 Horizons.* Retrieved 11/10/2005 from http://www.broadcast.net/~sbe79/oct01sbe.pdf

Wills, C., Mikhailov, M., Shang, H. (2001, May 1-5). N for the price of 1: Bundling web objects for more efficient content delivery. *WWW10.* May 1-5, 2001 Hong Kong, ACM 1-58113-348-0/01/0005.

Wikipedia. (n.d.). JPEG: potential patent issues. *Wikipedia, the free Encyclopedia.* Retrieved 10/29/2005 from http://en.wikipedia.org/wiki/Jpeg#Potential_patent_issues.

Wikipedia. (n.d.). World Wide Web. *Wikipedia, the free Encyclopedia.* Retrieved 10/29/2005 from http://en.wikipedia.org/wiki/World_wide_web

VITA

Michael Edmund Butterfield was born in Houston, Texas, on January 10, 1958, the son of Robert Albert Butterfield and Geneva Louise Butterfield. He entered San Antonio College to begin general undergraduate studies in 1976. He later entered Texas State University-San Marcos (formerly Southwest Texas State University) in 1981 and received a Bachelor of Arts degree in Industrial Technology in 1983.

Michael re-entered the Graduate College at Texas State University-San Marcos in 2001 and received a Certificate of Computer Science in 2002. The certificate partially fulfilled the entrance criteria for the 'Master of Science degree with a major in Software Engineering' degree program.

Permanent Address: 5101 Hereford Way

Austin, Texas 78727

This thesis was typed by Michael Edmund Butterfield.