1-1993

# Color space selection for JPEG image compression

Nathan Moroney

# Color Space Selection for JPEG Image Compression

Nathan Moroney

B.S. Philadelphia College of Textiles and Science (1991)

A thesis submitted for partial fulfillment
of the requirements for the degree of
Master of Science in Color Science
in the Center for Imaging Science
on the College of Imaging Arts and Sciences
of the Rochester Institute of Technology

December 1993

Nathan M. Moroney
Signature of Author

Mark D. Fairchild
Accepted by

College of Imaging Arts and Science
Rochester Institute of Technology
Rochester, New York

CERTIFICATE OF APPROVAL

---

M.S. DEGREE THESIS

---

The M.S. Degree Thesis of Nathan Moroney
has been examined and approved
by two members of the color science faculty
as satisfactory for the thesis requirement for the
Master of Science degree.

Mark D. Fairchild

Dr. Mark Fairchild, Thesis Advisor

Roy Berns

Dr. Roy Berns

# Thesis Release Permission Form

Rochester Institute of Technology
Center for Imaging Science

Title of Thesis Color Space Selection for JPEG Image Compression

I, Nathan M. Moroney, hereby grant permission to the Wallace Memorial Library of R.I.T. to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Date 12/16/93

# Color Space Selection for JPEG Image Compression

Nathan Moroney

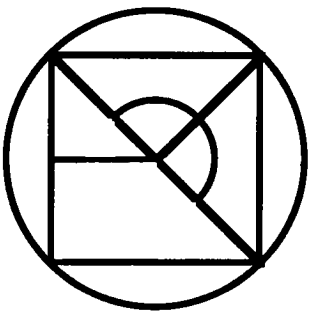
Submitted for partial fulfillment
of the requirements for the degree of
Master of Science in Color Science
in the Center for Imaging Science
at the Rochester Institute of Technology

## ABSTRACT

The Joint Photographic Experts Group's image compression algorithm has been shown to be a very efficient and powerful method of compressing images. However, there is little substantive information about which color space should be utilized when implementing the JPEG algorithm. Currently, the JPEG algorithm is set up for use with any three component color space. The objective of this research was to determine whether or not the color space selected will significantly improve image compression capabilities. The RGB, XYZ, YIQ, CIELAB, CIELUV, and CIELAB LCh color spaces were examined and compared. Both numerical measures and psychophysical techniques were used to assess the results. The final results indicate that the device space, RGB, is the worst color space to compress images. In comparison, the nonlinear transforms of the device space, CIELAB and CIELUV, are the best color spaces to compress images. The XYZ, YIQ, and CIELAB LCh color spaces resulted in intermediate levels of compression.

**Acknowledgments**

I would like to thank the following people for all of the help and assistance that I received as I carried out my thesis research: the Munsell Lab, for providing the support and environment that made all of this possible, Mark Fairchild, for suggesting such a terrific research topic and for overseeing the entire process, Mike Stokes, for getting everything started by patiently helping me to debug and implement my JPEG code, Tim Kohler, for countless programming tips, software details, and trivia, and Eli, for her hope and love.



"Learning changes us; it does what all nourishment does which also does not merely 'preserve' -as physiologists know. But at the bottom of us, really 'deep down' there is, of course, something unteachable, some granite of spiritual fatum, of predetermined decision and answer to predetermined selected questions. Whenever a cardinal problem is at stake, there speaks an unchangeable 'this is I'; about man and woman, for example, a thinker cannot relearn but only finish learning -only discover ultimately how this is 'settled in him.' At times we find certain solutions of problems that inspire strong faith in us; some call them henceforth their 'convictions.' Later -we see them only as steps to self-knowledge, signposts to the problem we are, -rather, to the great stupidity we are, to our spiritual fatum, to what is unteachable very 'deep down.' "

Friedrich Nietzsche *Beyond Good and Evil*

## Table of Contents

## Table of Contents (continued)

## List of Images

## List of Figures

## List of Figures (continued)

## List of Tables

# 1. Introduction

Any given scene viewed by a human observer is detected using roughly seven million receptors. This visual information is then automatically compressed so that it can be transmitted by the one million or so optic neurons to the brain. This means that the human visual system routinely achieves a visually lossless level of compression of about seven to one (Granrath 1981). It is encouraging to realize that such levels of compression of visual information have been occurring right under our noses for hundreds of thousands of years. However, it has only been within the past decade that the same level of compression has been possible with digital color images.

One of the most widely used and successful methods that has been developed is the Baseline JPEG compression algorithm. JPEG is an acronym standing for the Joint Photographic Experts Group and its name was originally used to identify the joint ISO / CCITT committee set up to study color image compression. The name is now also used to identify the industry standard compression algorithm which the committee has been formulating, revising and is about to approve as an international standard (Mitchell and Pennebaker 1991).

Image compression refers to any class of techniques used to reduce the amount of information needed to represent a digital image. This can be done in a number of different ways and JPEG utilizes two of the most common approaches. The first group of techniques involves methods which attempt to reduce any redundancies in the image, such as spatial, temporal or spectral correlations. The second group involves systematically throwing out less visually significant information from the image (Limb 1976).

The JPEG algorithm is an important part of the rapidly growing compression technologies. These advances in imaging science are opening entire new regions for future development. Currently, JPEG is being used to improve the storage capacity for digital images in an electronic environment. This means that storing and communicating digital

images is now faster and less expensive. In the near future, JPEG and related technologies will be vital for such products as high definition television, color faxes, video telephones and other high-end image storage and retrieval systems (Leger, Omachi and Wallace 1991).

At the present time, there is little substantive information about which color space should be utilized when implementing the JPEG algorithm (Wallace 1988). Previous research has shown that the RGB device space is a poor choice for a color image compression space. Some authors have suggested that the NTSC YIQ, NTSC YCbCr, or the CIE XYZ tristimulus space should be used when compressing color images (Pratt 1971 and Ylakoski and Ronngren 1992). These results provide a starting point but they are far from providing any concrete details about which of the multitude of color spaces available would be optimal for JPEG compression. In addition, there is minimal reported psychophysics which have been published on this issue. This thesis research investigates this topic and will provide numerical and psychophysical assessment of which color space should be used with the JPEG algorithm.

The significant differences in the various color spaces suggest that the color space selected for use with the JPEG algorithm will affect the resulting compression. The differences in the perceptual uniformity, channel redundancy, and compatibility with the algorithm are just a few of the variables that could affect the efficiency of the resulting compression. This thesis research also attempts to determine how these factors influence color image compression.

# 2. Background

The following sections discuss the details of the Baseline JPEG algorithm. Then the specifics of each of the color spaces used in this research are reviewed. Finally, a computational example is presented. This example follows a sample 8x8 block of pixels from original RGB form, through the color space transformations, through the compression steps and then back out through the decompression steps. This further illustrates the discussions of the JPEG algorithm and the color space transformations. Moreover, the example provides a useful reference for the Results and Discussion section.

## 2.1. Baseline JPEG

One of the major objectives of the JPEG committee was to establish a basic compression technique for use throughout industry. A method that would be compatible with all of the various types of hardware and software that would be performing image compression. To accomplish this goal the committee developed the Baseline JPEG algorithm. Additional changes could then be made to the Baseline algorithm according to individual users preferences, but only the Baseline algorithm would be universally implemented and utilized. This algorithm was designed to accept 8 bits per pixel images. It should be noted that in compression literature, the term pixel is used loosely to mean either a single monochrome pixel or a single channel of a color pixel. The same terminology is used in this thesis and the reader should be aware that, unless otherwise specified, the word pixel will be synonymous with a single pixel channel or an individual red, green or blue pixel component.

The Baseline JPEG algorithm is composed of three major compression steps and three major decompression steps. This is shown in flowchart form in Fig. 2.1 on the following page. The first step in the process is to break the original image down into eight

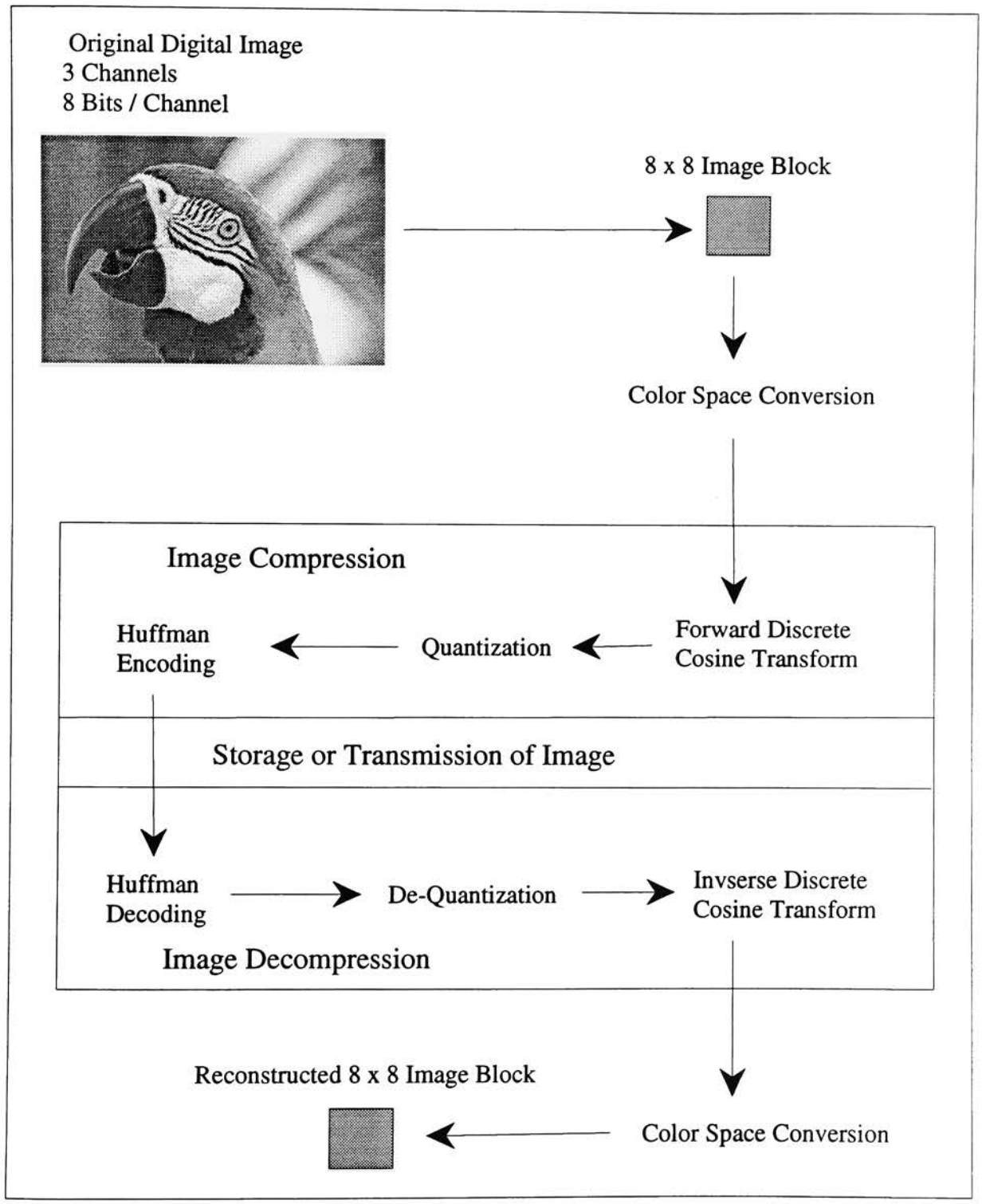



Fig. 2.1 Flowchart for Baseline JPEG Algorithm.

by eight pixel blocks. These blocks are then sent, one at a time, into the JPEG algorithm. On this chart the JPEG algorithm is represented by the large central box in the figure. The steps in the compression algorithm are a transformation from a spatial representation to a frequency representation, systematic quantization and statistical encoding. These three steps would be repeated until the entire image was in a compressed form. At this point, the image can be stored or transmitted as needed.

The steps in the decompression algorithm are decoding the bit stream, dequantization, and transforming from a frequency representation back to a spatial representation. Once again the basic computational unit is an eight by eight pixel block and when the last three steps have been repeated for all of the data an image will be reconstructed. The necessary color space transformations were performed before and after the JPEG algorithm. The CIELAB, CIELAB LCh, CIELUV, RGB, XYZ, and YIQ spaces were selected as the color spaces to be examined during the course of this research. The following sections provide additional details about each of these steps.

### 2.1.1. The FDCT and IDCT

The mathematical transformation used in the JPEG algorithm for changing from a spatial representation of the pixels to a frequency representation is the Discrete Cosine Transform or DCT. The forward discrete cosine transform or FDCT can be thought of as a harmonic analyzer (Wallace 1992). The input to the FDCT is the 8x8 pixel block with a constant, $2^{(n-1)} - 1$, subtracted from all the values. The value n is the number of bits of per pixel and, in this case, n was equal to 8 and 128 was subtracted from all the pixels. This bipolar spatial data is then decomposed into the corresponding DCT basis functions. This process is similar to the Fourier Transform, but the DCT uses all real numbers and does not assume periodicity (Ahmed 1973). The equation for the FDCT is expressed as:

$$F(u,v) = \frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cdot\cos\frac{(2x+1)u\pi}{16}\cdot\cos\frac{(2y+1)v\pi}{16}\right] \quad (2.1)$$

where:

$$C(u),C(v) = \frac{1}{\sqrt{2}} \text{ for } u,v = 0,$$
$$C(u),C(v) = 1 \text{ otherwise.}$$

The u and v in Eqn. 2.1 are the indices for the frequency space coefficients. The x and y, on the other hand, are used to index through the original 64 pixels in their spatial representation. The two cosine terms are responsible for generating the two-dimensional sinusoidal grating patterns that are the basis functions.

The actual DCT basis functions are shown in Fig. 2.2. The vertical frequency of the patterns increases from left to right and the horizontal frequency increases from top to bottom. The upper left corner is a special case because it is the lowest frequency DCT basis function and it has, effectively, no frequency. This coefficient has been given the name the DC term and the other 63 terms are commonly referred to as the AC coefficients. This figure also provides a qualitative idea of what actually happens when an eight by eight pixel block is sent through the DCT. Basically, the equation will determine which of the 64 eight by eight basis functions are needed and in what magnitude to generate the original eight by eight pixel block.

The inverse discrete cosine transform, or IDCT, is just the opposite of the FDCT and it transforms an eight by eight block of DCT coefficients to a reconstructed eight by eight block of pixels. In this case, frequency information is being transformed back into a spatial representation. This process can be thought of as a harmonic synthesizer (Wallace 1992). The equation form of the IDCT is written:

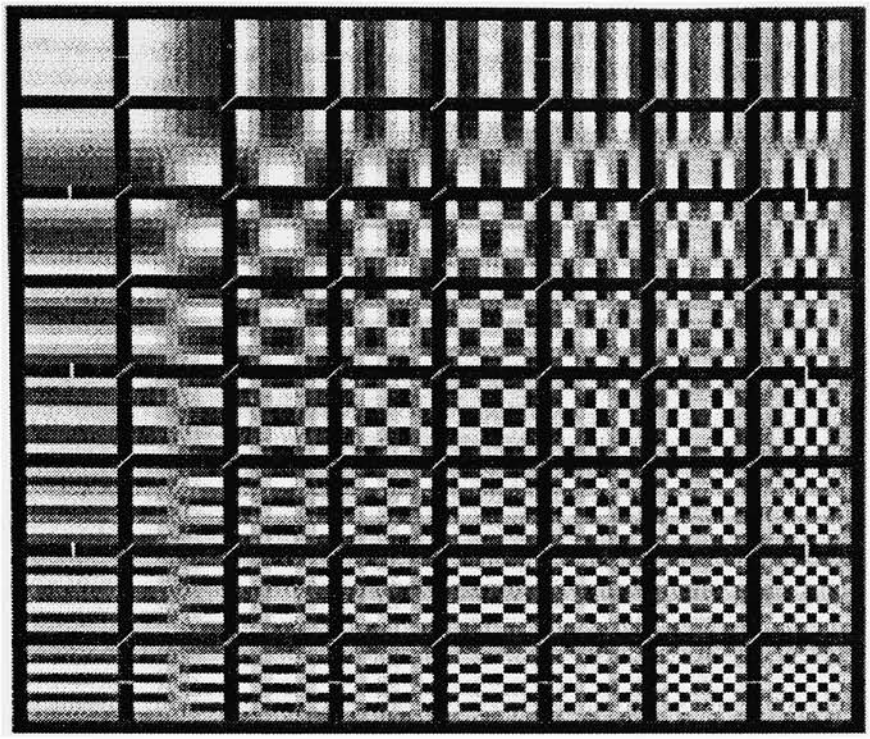


Fig. 2.2. The 64 DCT Basis Functions (From Mitchell and Pennebaker 1993).

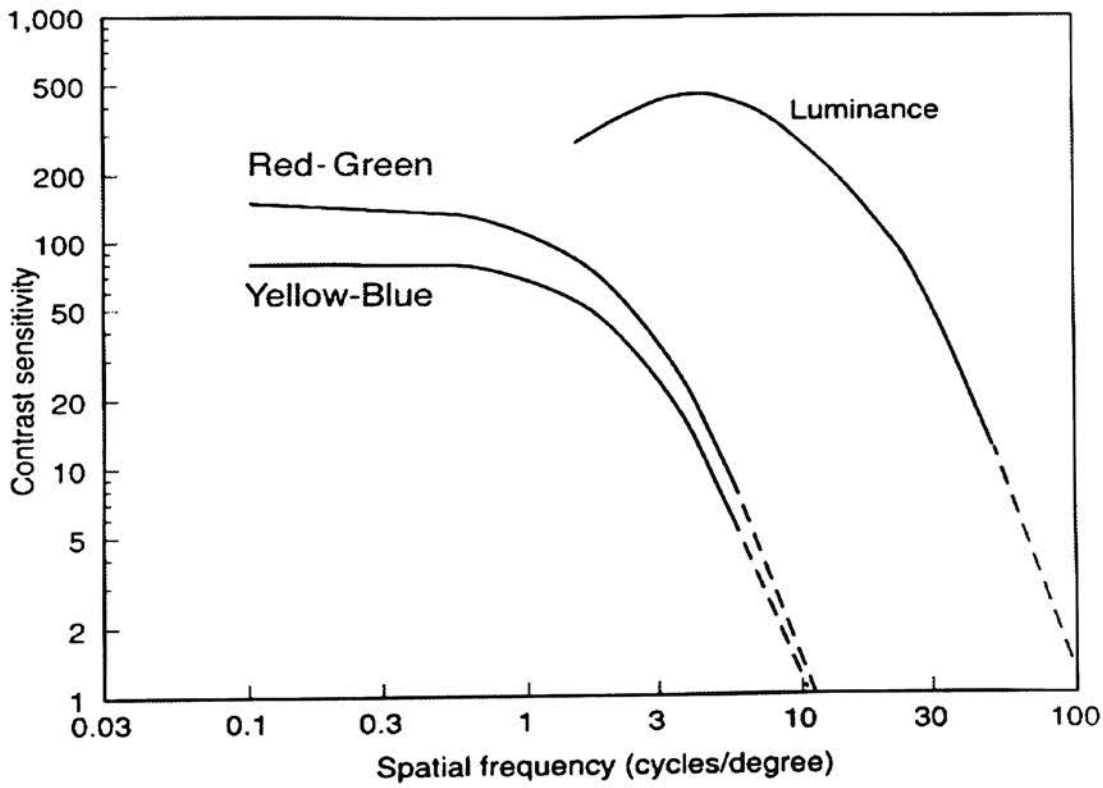



Fig. 2.3. Human Luminance and Chrominance CSF's
(From Mitchell and Pennebaker 1993).

$$f(x,y) = \frac{1}{4}\left[\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v)\cdot\cos\frac{(2x+1)u\pi}{16}\cdot\cos\frac{(2y+1)v\pi}{16}\right] \qquad (2.2)$$

where:

$$C(u),C(v) = \frac{1}{\sqrt{2}} \text{ for } u,v = 0,$$
$$C(u),C(v) = 1 \text{ otherwise.}$$

The variables u, v, x, and y perform the same roles in the IDCT as in the FDCT. The output of the IDCT must be shifted back from a signed integer to an unsigned integer by adding $2^{(n-1)}$ - 1 to all of the values in order to get the original values.

Both the FDCT and the IDCT are essentially lossless. A lossless transform is one in which none of the original information is lost. This transform is a crucial step in JPEG because it concentrates a large portion of the signal in the lower frequencies (Gonzalez and Woods 1990). As a result, the DCT does not actually compress the information but it does compact it significantly.

### 2.1.2. Quantization and Dequantization of the DCT Coefficients

The next step in the flow chart is the quantization of the FDCT coefficients. This step compresses the image by selectively discarding information. The JPEG scheme for quantizing the DCT coefficients attempts to model the human visual system. Specifically, the DCT terms are quantized according to their visual significance.

Early on the JPEG committee recognized that color information could be quantized differently than monochrome information. The largest difference was that chrominance or color information could be quantized more coarsely than the luminance or tone information (Hunt 1988). This scheme was based on the physiological properties of the visual system. To illustrate, the human luminance and chrominance contrast sensitivity functions are

shown in Fig. 2.3. This figure shows that the luminance contrast sensitivity function is roughly bell shaped and peaks around 7 cycles per degree. The chrominance contrast sensitivity function, in comparison, is plateau shaped and peaks at a lower frequency level. The maximum sensitivity is also higher for the luminance information than for the chrominance frequencies. This demonstrates that the human visual system is more sensitive to luminance frequency changes than chrominance frequency changes (Schreiber 1991). The JPEG quantization scheme attempts to model this relationship, as well as some of the details.

The JPEG committee has not provided any quantization tables, or Q-Tables, for use as standards. They have provided two example tables, shown in Fig. 2.4, for use with the algorithm and this research utilized these two tables. One table is for quantizing luminance information and the other is for chrominance quantization. It has been commented that there are some irregularities in these two tables (Klein 1992). Nevertheless, the basic concepts of quantizing chrominance information more coarsely than luminance information and of varying the rate of quantization based on the frequency of the basis functions are valid and have been supported by experimental results (Lohscheller 1984).

JPEG Luminance Q-Table

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 16 | 11 | 10 | 16 | 24 | 40 | 51 | 61 |
| 12 | 12 | 14 | 19 | 26 | 58 | 60 | 55 |
| 14 | 13 | 16 | 24 | 40 | 57 | 69 | 56 |
| 18 | 22 | 37 | 56 | 68 | 109 | 103 | 77 |
| 24 | 35 | 55 | 64 | 81 | 104 | 113 | 92 |
| 49 | 64 | 78 | 87 | 103 | 121 | 120 | 101 |
| 72 | 92 | 95 | 98 | 112 | 100 | 103 | 99 |

JPEG Chrominance Q-Table

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 17 | 18 | 24 | 47 | 99 | 99 | 99 | 99 |
| 18 | 21 | 26 | 66 | 99 | 99 | 99 | 99 |
| 24 | 26 | 56 | 99 | 99 | 99 | 99 | 99 |
| 47 | 99 | 99 | 99 | 99 | 99 | 99 | 99 |
| 99 | 99 | 99 | 99 | 99 | 99 | 99 | 99 |
| 99 | 99 | 99 | 99 | 99 | 99 | 99 | 99 |
| 99 | 99 | 99 | 99 | 99 | 99 | 99 | 99 |

Fig. 2.4 Example Luminance and Chrominance Q-Tables

Another way of illustrating the quantization scheme proposed by the JPEG committee is shown in Fig. 2.5. In this figure, the 64 DCT basis functions from Fig. 2.2 are shown with varying sizes. The size of the basis functions is indicative of how that particular DCT basis function will be quantized using the luminance Q-Table shown in Fig. 2.4. The larger the basis function, the more finely the signal will be quantized and the less information will be discarded. In comparison, the smaller the basis function the more coarsely it will be quantized and the more information will be lost. The amount of quantization for each block has been scaled relative to each blocks relative size. Notice how much larger the basis functions in the upper left portion of the figure are compared to the other basis functions.

The actual equation for quantization is to divide each FDCT coefficient by the corresponding element in the Quantization Table. This number is then rounded to the nearest integer and the entire operation can be expressed as follows:

$$F_Q(u,v) = \text{Integer Round}\left(\frac{F(u,v)}{Q(u,v)}\right) \tag{2.3}$$

where u and v are indices used to locate the specific elements in the 8x8 FDCT block and the Q-Table, F(u,v) are the FDCT coefficients and Q(u,v) are the Q-Table values.

The dequantization is the inverse operation of the quantization process. This step uses the same two quantization tables as are used for quantizing the FDCT coefficients. In this case the Q-Table values are multiplied by the IDCT coefficients. This can be written in the following form:

$$F_{Q'}(u,v) = F_Q(u,v) \cdot Q(u,v) \tag{2.4}$$

where u, v, and Q(u,v) are the same variables as specified previously and $F_Q(u,v)$ are the IDCT coefficients.

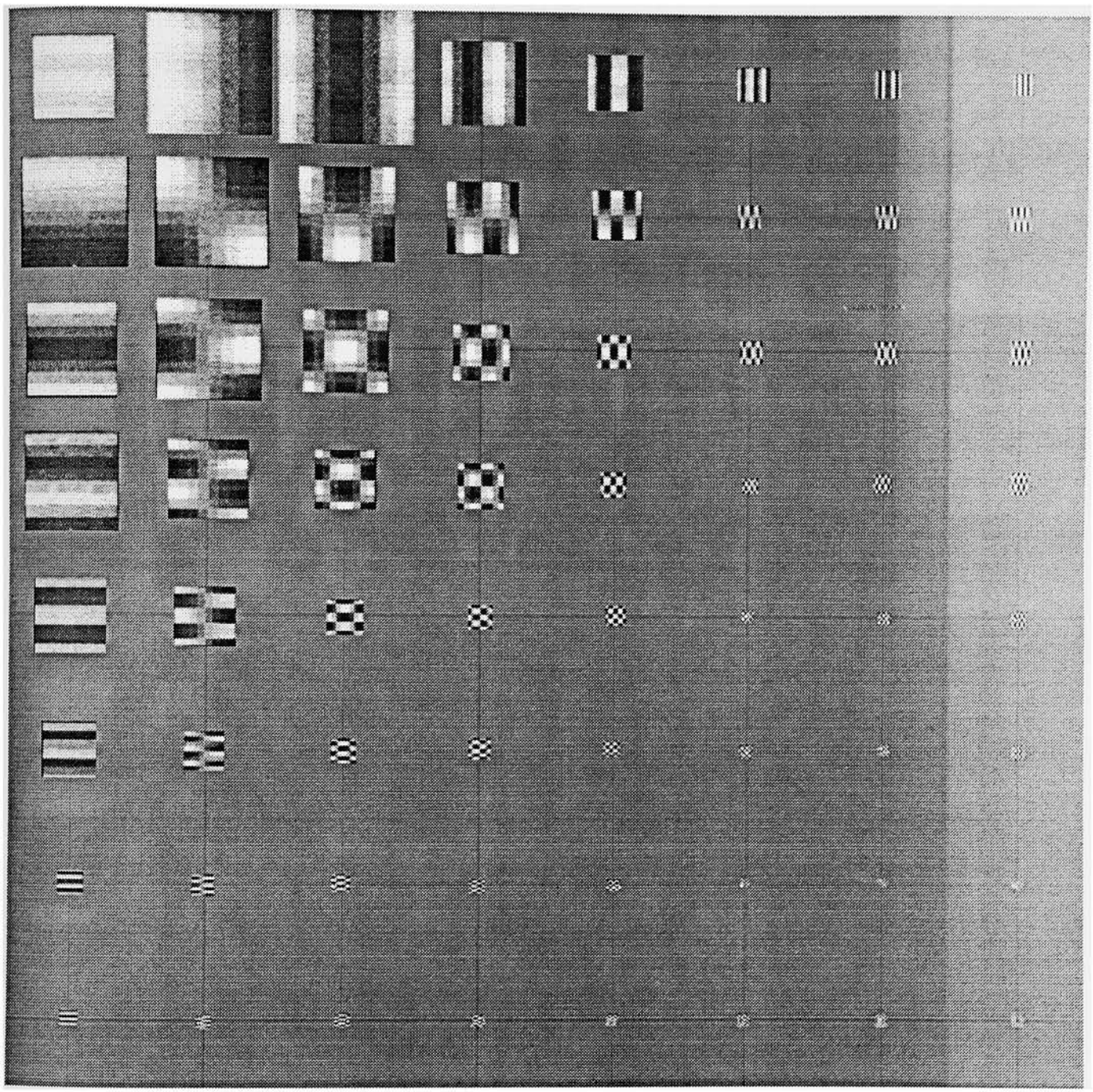


Fig. 2.5 Quantization Scaled DCT Basis Functions.

The Q-Tables are a vital part of the JPEG algorithm, not only because of the amount of compression achieved, but because they provide a mechanism for determining how much an image will be compressed. The level of compression can be determined by scaling all of the elements in the Q-Tables by some constant factor (Wallace 1990). For instance, to quantize the signal more coarsely, all of the Q-Table values should be larger than their default values. On the other hand, to quantize the signal more finely, all of the Q-Table values should be smaller than their default values. Multiplying all of the elements by a constant greater than one will increase all of the Q-Table values and will, consequently, discard more information. Multiplying all of the elements by a constant less than one will decrease all of the Q-Table values and will discard less information.

However, if the DCT coefficients are not quantized at all, assuming no round-off error, there will be no information lost. This type of compression is referred to as lossless compression. Although lossless compression is completely error free, it is not used extensively because the resulting compression of the image is minimal. Therefore, most image compression discards information in what is often referred to as lossy compression. The JPEG Baseline algorithm, with its quantization step, is a lossy algorithm. Nevertheless, it is possible to perform lossy compression in which there is no perceptible degradation in the image. This is known as visually lossless compression and results in a compression in which a maximum data compression occurs with no distortion in the image quality (Gentile, Allebach, and Walowit 1990). The first experiment performed for this thesis research deals with determining levels for visually lossless JPEG compression.

### 2.1.3. Huffman Encoding and Decoding

The final step in the Baseline JPEG algorithm is take the quantized FDCT coefficients and to encode them according to image statistics so that they will take up even less space. Basically, this step works by giving the most commonly occurring sequences

of quantized FDCT coefficients the shortest code words. Likewise, the least commonly occurring sequences of quantized FDCT coefficients are given the longest code words. Huffman encoding assigns a unique binary codeword to each sequence of FDCT coefficients. This step is similar to taking advantage of the fact that some letters in the English language occur more frequently than others. For example, a commonly occurring letter like A or E might be given a code word of 010 or 11 and a less frequent letter like Q might be given a code word of 1001110 (Raisbeck 1965). These techniques are a part of a larger field of study known as information theory.

Before the actual encoding sequence begins, the quantized DCT coefficients are reorganized in order to take advantage of the compaction of information that occurs

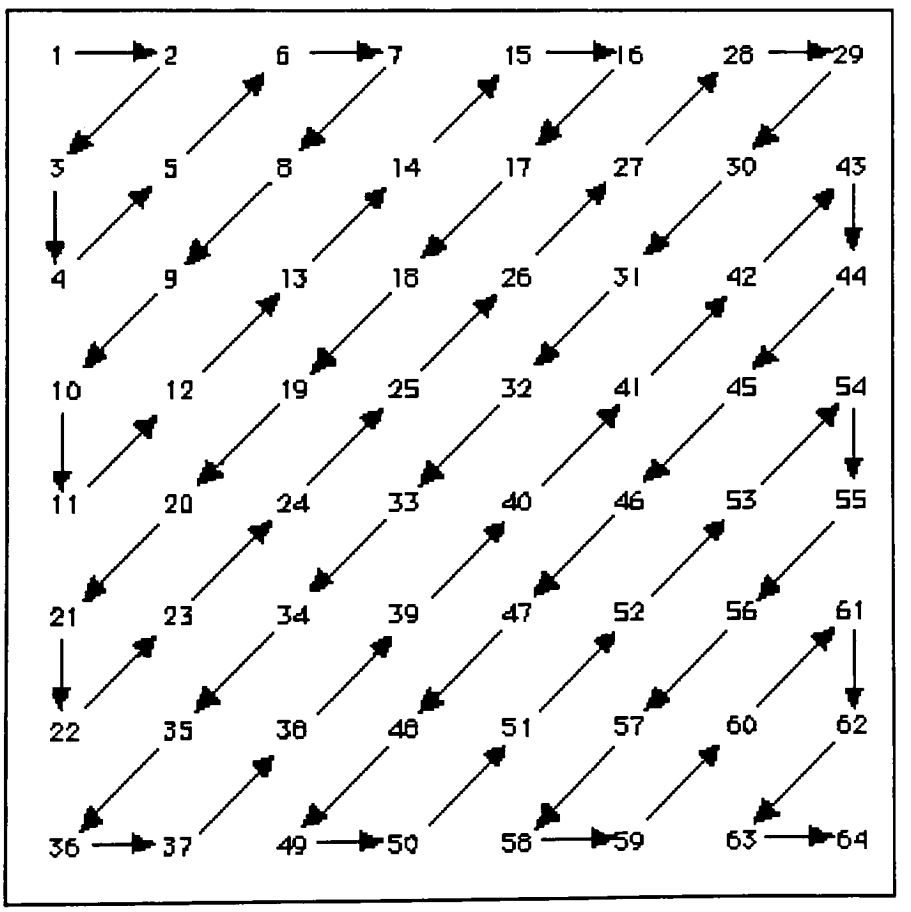



Fig. 2.6 The Zig-Zag Ordering of the Quantized DCT coefficients.

following the DCT. This new ordering is shown in Fig. 2.5 as an arrow zig-zagging from the upper left corner to the lower right corner. This encoding sequence allows all of the lower frequency values to be encoded first. The higher frequencies, which are usually all zeros after quantization, can then be encoded all at once.

The first value to be encoded is the DC term and this coefficient is a measure of the average value of the 64 coefficients in that block. Generally, there is a fairly high correlation between neighboring DC coefficients. As a result, the amplitude of the DC term to be encoded is the current DC coefficient minus the previous coefficient.

The two values that are then used to assign the DC coefficient's code word are the category and the amplitude of the term. The category is determined according to which range of values the amplitude falls in. Table 2.1 shows how the range that an amplitude falls within is converted to a category. The range intervals are based on the powers of two and can be both positive and negative. The category determines the most significant bits, or MSB, of the codeword to be assigned. Table 2.2 shows how the MSB of the codewords are assigned to the DC categories. The total number of bits to encode a given category is the codelength and Table 2.2 lists these values. This table is based on the statistical properties of a set of images analyzed in a previous study. Once again, this table is just a sample provided by the JPEG committee. There are separate tables for the luminance and the chrominance FDCT coefficients.

Next, the least significant bits or LSB of the codeword must be computed. The LSB is calculated by taking either the positive difference between the MSB and the amplitude of the DC term or the negative difference minus 1. The MSB and the LSB can the combined for the final codeword. To illustrate, if the current luminance DC coefficient was 56 and the previous DC coefficient was 51, the difference value or amplitude would be 5. Using Table 2.1, a value of 5 is in category 3. The MSB of category 3 terms is 100 and the difference between 100000 and 101 is 011. The final codeword for this DC term can then be written 100011.

Table 2.1 Categorization Table for AC and DC Coefficients

| Range | DC Category | AC Category |
|---|---|---|
| 0 | 0 | N/A |
| -1, 1 | 1 | 1 |
| -3, -2, 2, 3 | 2 | 2 |
| -7, ..., -4, 4, ..., 7 | 3 | 3 |
| -15, ..., -8, 8, ..., 15 | 4 | 4 |
| -31, ..., -16, 16, ..., 31 | 5 | 5 |
| -63, ..., -32, 32, ..., 63 | 6 | 6 |
| -127, ..., -64, 64, ..., 127 | 7 | 7 |
| -255, ..., -128, 128, ..., 255 | 8 | 8 |
| -511, ..., -256, 256, ..., 511 | 9 | 9 |
| -1023, ..., -512, 512, ..., 1023 | 10 | 10 |
| -2047, ..., -1024, 1024, ..., 2047 | 11 | 11 |
| -4095, ..., -2048, 2048, ..., 4095 | 12 | 12 |
| -8191, ..., -4096, 4096, ..., 8191 | 13 | 13 |
| -16383, ..., -8192, 8192, ..., 16383 | 14 | 14 |
| -32767, ..., -16384, 16384, ..., 32767 | 15 | N/A |

Table 2.2 Baseline entropy encoding table for DC terms.

| Category | Luminance Codelength | Luminance MSB of Codeword | Chrominance Codelength | Chrominance MSB of Codeword |
|---|---|---|---|---|
| 0 | 2 | 00 | 2 | 00 |
| 1 | 4 | 010 | 3 | 01 |
| 2 | 5 | 011 | 4 | 10 |
| 3 | 6 | 100 | 6 | 110 |
| 4 | 7 | 101 | 8 | 1110 |
| 5 | 8 | 110 | 10 | 11110 |
| 6 | 10 | 1110 | 12 | 111110 |
| 7 | 12 | 11110 | 14 | 1111110 |
| 8 | 14 | 111110 | 16 | 11111110 |
| 9 | 16 | 1111110 | 18 | 111111110 |
| 10 | 18 | 11111110 | 20 | 1111111110 |
| 11 | 20 | 111111110 | 22 | 11111111110 |

The 63 AC terms are encoded using a somewhat similar scheme, but in addition to the amplitude and category information, the number of zeros proceeding that term is also encoded. This is done because, as was previously stated, once the DCT terms have been quantized a majority of the AC coefficients are now zero. This variable length encoding uses the number of zeros proceeding the term, or runlength, and the category of the AC

coefficient to determine the most significant bits of the code word. The amplitude of the AC coefficient is just the value of that term; no differential encoding is used for the AC coefficients. The runlength can be computed by simply counting the number of zeros preceding the coefficient. The category of the AC term is determined by finding which category the amplitude of the AC coefficient falls in. The process is based on the same table used for the determining the DC categories and is also shown in Table 2.1.

Then the runlength and category values would be used to look up the MSB of the code word. This table consists of 15 possible runlengths by 15 possible sizes of codewords. There are also two other special code words. One of the special code words signifies that all of the remaining coefficients in the block are zero. This is known as the end of block or EOB codeword. The other special code word is for an AC term of amplitude 0 and a runlength of 15. Once again, there are two different tables to be used depending on whether the block contains luminance or chrominance information. The JPEG committee has provided two example tables for use with the JPEG algorithm. Due to the size of these two tables, they are listed in Appendix A for reference. These tables can then be used to determine the MSB of the AC coefficients code word. The LSB of the codeword are determined by taking either the positive difference between the MSB and the amplitude of the AC term or as the negative difference minus 1.

From here the encoded bits from the Huffman encoded DC and AC terms are joined together into one bit string. At this point the image is maximally compacted. There are numerous other details to attend to such as establishing a storage format, byte-stuffing, and other operations but this level of detail suffices for this research.

### 2.1.4. Measures of Image Compression

When the final bit string has been assembled it is possible to determine the amount compression that has occurred as a result of the JPEG algorithm. For this research, four

different measures were used to assess the resulting image compression. These measures are the final bit size of the image, the compression ratio, the number of bits per pixel for the compressed image, and the number of bits per minute squared for the compressed image.

The final size of the compressed image is nothing more than the size in computer memory measured in bits that the compressed image took up. The compression ratio was computed by dividing the number of bits in the original image by the number of bits in the compressed image. The is can be formulated as follows:

$$\text{Compression Ratio} = \frac{\text{Bits in Original Image}}{\text{Bits in Compressed Image}} \tag{2.5}$$

The resulting value can range from one to some large positive number, usually in the range of 4 to 10. No compression in the image would correspond to a compression ratio of one. Likewise an image compressed to half the size of the original would result in a compression ratio of two.

The bits per pixel measure of compression is another simple quotient (Rabbani 1990). To compute the number of bits per pixel the number of bits in the compressed image is divided by the number of pixels in the image. This can be written:

$$\text{Bits / Pixel} = \frac{\text{Bits in Compressed Image}}{\text{Number of Pixels}} \tag{2.6}$$

Remember, as was previously mentioned, pixels actually refers to the individual pixel channels. The original image has eight bits per pixel and after JPEG compression the values can range anywhere from a little under one to about two and a half.

Lastly, the bits per minute squared has been suggested as a measure of image compression that avoids the ambiguities of pixel size and distance of observer from the monitor. In this case, the number of bits is divided by the number of pixels as was done

for the bits per pixel computations. Then, this value is multiplied by the size of one pixel in minutes squared. This can be expressed as:

$$\text{Bits / Minute}^2 = \frac{\text{Bits in Compressed Image}}{\text{Number of Pixels}} \cdot \text{Pixel size in Minutes}^2 \qquad (2.7)$$

This measure of image compression is independent of pixel size and viewing distance. This is important because the perceived quality of the image is dependent on the viewing conditions (Klein 1992).

Typically, an image can be compressed anywhere from a quarter to an eighth of the original size and still be visualy indistinguishable from the original. Further compression is possible but only by sacrificing image quality. Therefore, a compression of 2 bits/pixel can usually be achieved with little or no loss of image fidelity. This dramatic reduction in image size is one of the major reasons that JPEG is used so extensively in many imaging applications.

## 2.2. Color Spaces

The JPEG committee, as was previously stated, did not select a standard color space for use with the algorithm. Initially, the YCbCr color space was considered for use as a standard color space, but this idea was abandoned. However, the YCbCr color space was used as the default color space for much of the experimental research carried out by the JPEG committee.

There has been a limited amount of research on how the use of other color spaces affects the compressibility of color images. Although, given the drastic differences among the various ways of representing color, it seems unlikely that there would be no differences among the color spaces. In fact, the XYZ and YCbCr color spaces have been shown to be better than the RGB color space for image compression (Ylakoski and Ronngren 1992 and

Peterson, Peng, Morgan, and Pennebaker 1991). Moreover, the CIELUV color space has been shown to be a good color space for the quantization of color information (Gentile, Allenbach, and Walowit 1990). Therefore, the initial hypothesis was made that the selection of a given color space would have a significant effect on the resulting compression.

The various color spaces currently available can be divided into three different categories. These divisions are device color spaces, linear transforms of device color spaces, and non-linear transforms of device color spaces. There are advantages and disadvantages for each of these three categories. The main trade-offs are between computational complexity and visual uniformity (Kasson and Plouffe 1990). There are also the additional considerations of channel redundancy and ease of sub-band coding.

The six color spaces selected for this research are the RGB, XYZ, YIQ, CIELAB, CIELAB LCh, and the CIELUV color spaces. One of the main reasons for selecting these color spaces is the widespread international use of all of these color spaces. In addition, these six spaces provide a good cross-section of the currently available color spaces. The following sections provide some background on each of these color spaces. The forward transformations to each of these color spaces are also be presented. For the sake of brevity, the inverse color space transformations are given in Appendix C.

### 2.2.1. Device Color Spaces

An example of the device color space for the CRT is the RGB or Red, Green, and Blue color space. In this color space, the RGB values are simply the digital counts defining the color image. This color space is used as an input to the CRT signal processing interface and therefore no additional computations are needed to display an image in RGB space. In comparison, any other color space will have to be transformed back into RGB space before it can be displayed. Regardless of this advantage, the RGB color space is not

very visually uniform. This means that equivalent steps in different areas of the RGB color space do not correspond to equivalent perceived changes in color. Likewise, the RGB color space is very correlated and there is a considerable redundancy of information in the three channels (Limb, Rubenstein and Thompson 1977). Lastly, the RGB color space is device dependent and cannot be directly displayed on any other device without altering the color fidelity of the original image.

The RGB color space can be further divided into two other spaces, raw RGB and monitor corrected RGB space. The raw RGB color space consists of the actual digital counts used to drive the monitor. However, by taking into account certain features of the CRT, like the gain, offset, and gamma, a new RGB space can be derived. This RGB color space is now only a linear transform away from tristimulus values. The terms needed to correct for the gain, offset, and gamma of the CRT are derived when the monitor is calibrated (Motta 1991). For the remainder of this section on color spaces, the monitor corrected RGB color space will be the one that is being utilized. Although after this section on color spaces, the raw RGB color space will be the one that is meant when the RGB color space is mentioned.

### 2.2.2. Linear Transforms of Device Space

An example of a linear transform of the device color space is the CIE XYZ or tristimulus color space. Extensive research has shown that the human visual system uses three different types of cones to perceive color. Direct measurement of these three cone sensitivities has proven to be an elusive and problematic goal. Therefore, in the 1930s several elaborate psychophysical experiments were conducted in order to derive the color matching functions. These color matching functions, while not the actual cone sensitivities themselves, are a linear transform of the cone sensitivities This color space is very useful for determining if two colors match or not. Essentially, the products of a given spectral

power distribution for an illuminant, a reflectance curve for a sample, and the color matching functions can be integrated to yield three values, the X, Y and Z values.

Previous research has demonstrated that the CIE XYZ space is a linear transform of the CRT's monitor corrected RGB space (Motta 1991). The exact linear transform necessary to convert from RGB to XYZ and back again is determined by colorimetrically calibrating the CRT. This process was carried out for the CRT used for this thesis research and the method and the results of the calibration are outlined in Appendix D. The actual transform derived can be written as 3x3 matrix and the equation form of the transform is:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 18.78 & 17.50 & 11.72 \\ 9.68 & 35.36 & 4.82 \\ 0.71 & 5.36 & 62.90 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.8}$$

The R, G, and B are the red, green, and blue digital counts for a given pixel normalized between 0 and 1 and the units for the XYZ values listed inside of the matrix are measured in lux. The 3x3 matrix shown above can then be inverted to transform from XYZ back to RGB space. The XYZ space is somewhat more visually uniform and less redundant than the RGB space.

A second example of a linear transform of a device color space is the NTSC YIQ color space. This color space is used for television and is very closely related to the YCbCr color space mentioned at the beginning of this section. The YIQ color space can be expressed as a 3x3 matrix going from a standardized NTSC RGB space to YIQ space or as a 3x3 transform going from XYZ to YIQ space (Buchsbaum 1987). Both are equivalent means of computing the same values. However, since the CIE XYZ values had already been computed the second method was used to compute the YIQ values. The exact 3x3 matrix transform used is as follows:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.000 & 1.000 & 0.000 \\ 1.389 & 0.827 & 0.453 \\ 0.938 & 1.195 & 0.233 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.9}$$

The inverse of the above matrix will go from the YIQ space back to XYZ. Like the XYZ color space, the YIQ space is a little more visually meaningful and considerably less redundant. The YIQ color space has the additional feature that it can be efficiently sub-band coded for transmission purposes.

### 2.2.3. Nonlinear Transforms of Device Space

The CIELAB color space is an example of a nonlinear transform of the device color space. The CIELAB space is among the most visually uniform color spaces currently available. However, it is somewhat computationally complex. The three coordinates of the CIELAB color space are L*, a* and b*. The lightness axis or L* represents how light or dark a given color is. The a* and b* coordinates are rectangular coordinates that locate a given color in a plane of constant lightness. A positive a* corresponds to reddish colors and a negative a* greenish colors. Likewise, a positive b* corresponds to yellowish colors and a negative b* bluish colors. In order to convert to the CIELAB space, the colors should first be in XYZ space. The following transformations will then yield CIELAB coordinates:

$$L^* = 116\left(\frac{Y}{Yn}\right)^{\frac{1}{3}} - 16 \quad \text{for } \frac{Y}{Yn} > 0.008856 \tag{2.10}$$

$$L^* = 903.3\left(\frac{Y}{Yn}\right) \quad \text{for } \frac{Y}{Yn} \le 0.008856 \tag{2.11}$$

$$a^* = 500\left[f\left(\frac{X}{Xn}\right) - f\left(\frac{Y}{Yn}\right)\right] \tag{2.12}$$

$$b^* = 200\left[f\left(\frac{Y}{Yn}\right) - f\left(\frac{Z}{Zn}\right)\right] \tag{2.13}$$

where

$$f\left(\frac{X}{Xn}\right) = \left(\frac{X}{Xn}\right)^{\frac{1}{3}} \quad \text{for } \frac{X}{Xn} > 0.008856 \tag{2.14}$$

$$f\left(\frac{X}{Xn}\right) = 7.787\left(\frac{X}{Xn}\right) + \frac{16}{116} \quad \text{for } \frac{X}{Xn} \leq 0.008856 \tag{2.15}$$

$$f\left(\frac{Y}{Yn}\right) = \left(\frac{Y}{Yn}\right)^{\frac{1}{3}} \quad \text{for } \frac{Y}{Yn} > 0.008856 \tag{2.16}$$

$$f\left(\frac{Y}{Yn}\right) = 7.787\left(\frac{Y}{Yn}\right) + \frac{16}{116} \quad \text{for } \frac{Y}{Yn} \leq 0.008856 \tag{2.17}$$

$$f\left(\frac{Z}{Zn}\right) = \left(\frac{Z}{Zn}\right)^{\frac{1}{3}} \quad \text{for } \frac{Z}{Zn} > 0.008856 \tag{2.18}$$

$$f\left(\frac{Z}{Zn}\right) = 7.787\left(\frac{Z}{Zn}\right) + \frac{16}{116} \quad \text{for } \frac{Z}{Zn} \leq 0.008856 \tag{2.19}$$

where X, Y, and Z are the tristimulus values of the sample and Xn, Yn, and Zn are the tristimulus values of the illuminant or, in this case, of the CRT's peak white. Since the CRT was calibrated to a peak white point of approximately D90 these values were 48.00 lux, 49.86 lux, and 68.97 lux respectively. The inverse color space transformations go from CIELAB back to XYZ. The result of all of these computations is a color space that is quite perceptually uniform (CIE 1984). In addition, these computations reduce some of the redundancy in the three channels.

Another non-linear transformation of device color space is CIELAB LCh space. This color space is simply the polar form of the CIELAB color space. Instead of the two rectangular coordinates a* and b*, CIELAB LCh space uses a vector with magnitude C* and an angle of $h_{ab}$. The C* vector corresponds to the perceptual attribute chroma and $h_{ab}$ corresponds to the hue of a color. The first component of CIELAB LCh space is exactly the same L* as for CIELAB color space and be calculated by using Eqns. 2.10 and 2.11. The C* and $h_{ab}$ quantities can computed using the formulae:

$$C^* = \sqrt{(a^*)^2 + (b^*)^2} \tag{2.20}$$

$$h_{ab} = \arctan\left(\frac{b^*}{a^*}\right) \tag{2.21}$$

where a* and b* are the same a* and b* computed using Eqs. 2.12-19. The inverse operations provided in Appendix C convert from CIELAB LCh coordinates back to CIELAB coordinates. One of the primary advantages of this color space is that all three axes correspond to one of the three major perceptual attributes of color. This color space is also very perceptually uniform and there is limited correlation between the channels.

Lastly, CIELUV is an example of another non-linear transformation of device space. This color space is similar to the CIELAB color space, but it is based on a chromaticity diagram. There is also an additional transformation necessary in order to get to CIELUV coordinates. The first step is to transform the tristimulus values X,Y, and Z to u' v' space using the following two equations:

$$u' = \frac{4X}{X+15Y+3Z} \tag{2.22}$$

$$v' = \frac{9Y}{X+15Y+3Z} \tag{2.23}$$

The u' and v' coordinates are then transformed to the CIELUV coordinates u* and v* according to the formulae:

$$u^* = 13L^*(u' - u'_n) \tag{2.24}$$

$$v^* = 13L^*(v' - v'_n) \tag{2.25}$$

where $u'_n$ and $v'_n$ are the u' v' coordinates of the source or the device peak white. The L* is the same L* calculated using Eqns. 2.10 and 2.11 for the CIELAB and CIELAB LCh spaces. The CIELUV color space is also a very uniform color space and the three axes are fairly uncorrelated (Wyszecki and Stiles 1982).

### 2.2.4. CRT Gamut Transforms

It is somewhat difficult to visualize how these equations actually transform the original device color space. Therefore, as an additional means of qualitatively comparing these color spaces, the three-dimensional gamut of the monitor used for this research has been plotted for five of these color spaces. The shape of the CRT gamut in each of these color spaces will provide a more intuitive understanding how Eqns. 2.8 to 2.25 transform the original RGB color information.

The CRT gamut is shown for the RGB, XYZ, YIQ, CIELAB, and CIELUV color spaces in Figs 2.7-11. In Fig. 2.7 the CRT gamut boundary is shown in monitor corrected RGB space. In this color space, the axes are the linearized red, green, and blue digital counts and the resulting form is a cube. The gray scale runs diagonally from bottom back corner to upper front corner. Fig. 2.8 shows the monitor gamut in XYZ space and Fig. 2.9 shows the gamut in YIQ space. In both of these figures, the original RGB cube has been stretched and rotated. However, the XYZ and YIQ transformations are both linear and, consequently, all of the box's edges are straight and all of the faces of the box are planar. Lastly, the CRT gamut boundary is shown in CIELAB and CIELUV color spaces in Figs 2.10 and 2.11 respectively. The CIELAB LCh color space is identical in shape to the CIELAB gamut shown in Fig 2.10 and therefore it is not shown separately. These nonlinear color spaces are quite different from the original RGB cube. The original cube has now been sheared and twisted. The edges and faces of the cube have been considerably distorted. It is also interesting to note that the neutral diagonal in the original RGB cube is now lined up along the L* lightness axis. These last two figures are attempting to represent the RGB cube in terms of a perceptually uniform space.

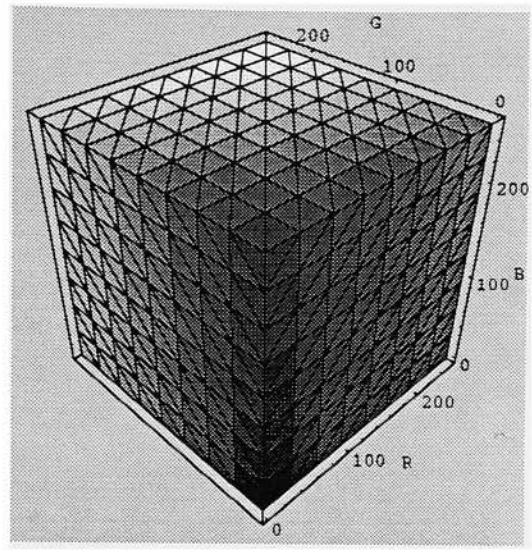



Fig. 2.7 CRT Gamut in RGB Space.

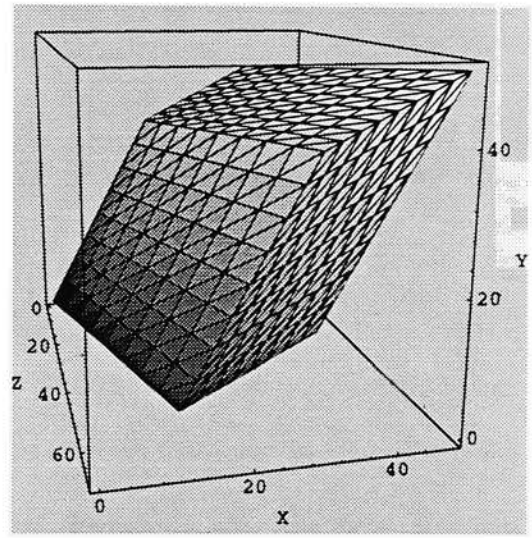



Fig. 2.8 CRT Gamut in XYZ Space.

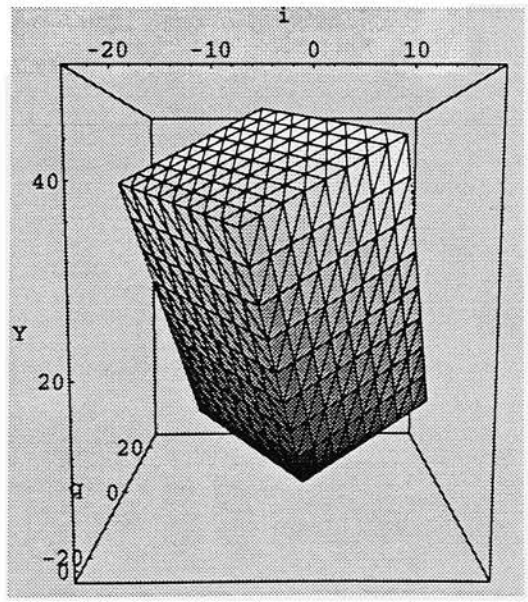



Fig. 2.9 CRT Gamut in YIQ Space.

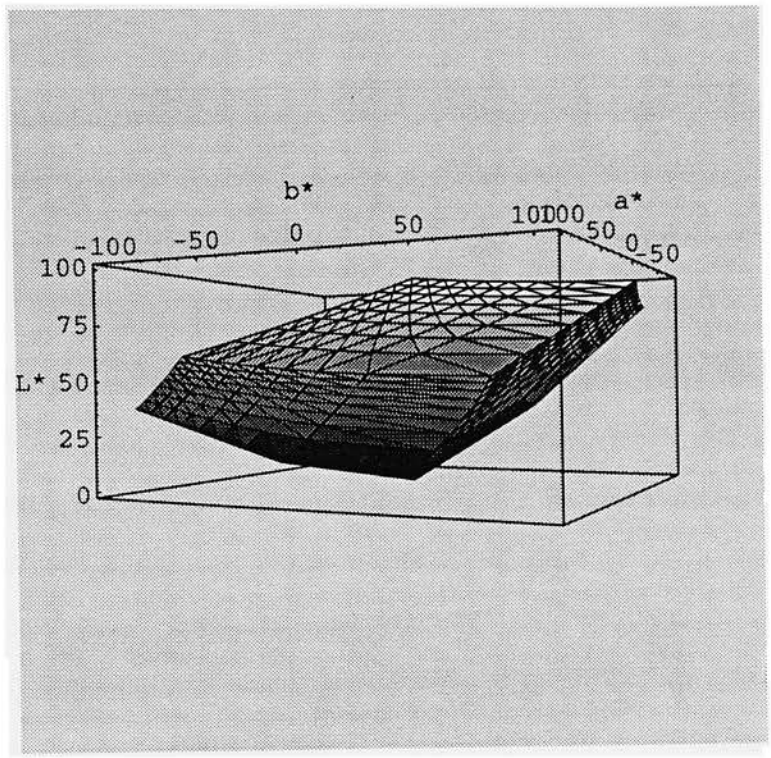



Fig. 2.10 CRT Gamut in CIELAB Space.

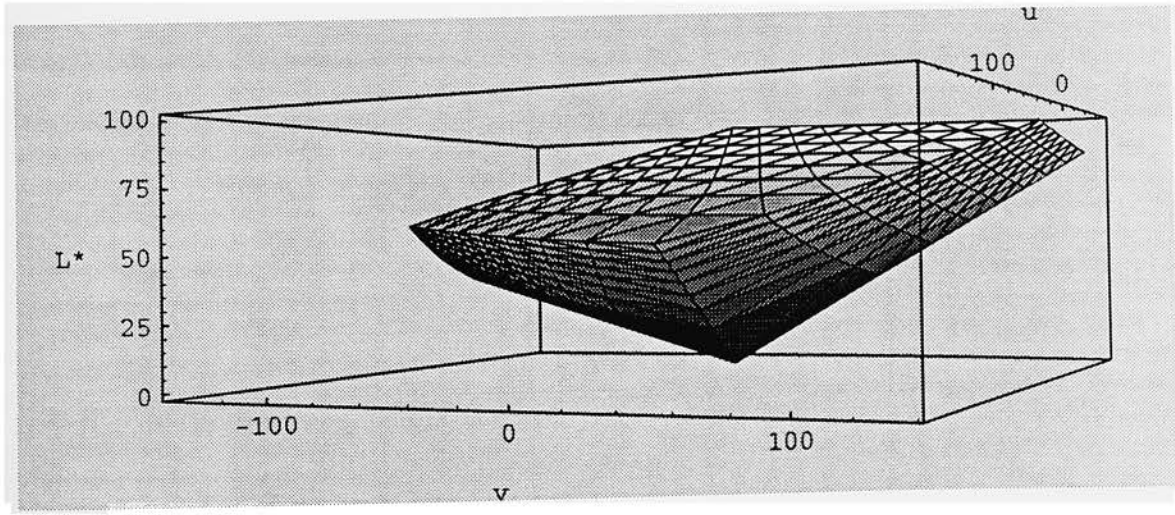



Fig. 2.11 CRT in CIELUV Space.

## 2.3. JPEG Quantization of Color Spaces

Once the color space has been selected, the only other free variable is how the data will be quantized. As was mentioned during the discussion of the Baseline JPEG algorithm, the DCT coefficients can be quantized using either the luminance Q-Table or the chrominance Q-Table. For each of the six color spaces selected, a quantization scheme was adopted for each of the three channels. These schemes are shown in Table 2.3 and, for the most part, the assignment of luminance or chrominance quantization is obvious. Quantities representing luminance or lightness, such as Y or L*, were quantized using the luminance Q-Table. Similarly, the color information axes, such as a*, b*, u*, v*, c*, $h_{ab}$, X, Z, I and Q, were quantized using the chrominance Q-Table. The decision to use the chrominance Q-Table for the X and Z planes could be debated. This scheme was used as a reasonable starting point and future research can address the issue of the optimal quantization schemes for XYZ space. This topic is further discussed in the conclusions section of this thesis The RGB color space was unique, in that it did not have a single channel for luminance type information. As a result, all three channels were quantized using the luminance Q-Table.

Table 2.3 Quantization Schemes Used for Different Color Spaces.

| Color Space | Channel 1 Q-Table | Channel 2 Q-Table | Channel 3 Q-Table |
|---|---|---|---|
| CIELAB | Luminance | Chrominance | Chrominance |
| CIELAB LCh | Luminance | Chrominance | Chrominance |
| CIELUV | Luminance | Chrominance | Chrominance |
| RGB | Luminance | Luminance | Luminance |
| XYZ | Chrominance | Luminance | Chrominance |
| YIQ | Luminance | Chrominance | Chrominance |

## 2.4. Computational Example

In order to further clarify the preceding two sections, a sample 8x8 pixel block will be processed through both the color space transformations and the Baseline JPEG algorithm. The first sequence of computations is shown in Fig 2.12. The original image is an 8x8 pixel block separated into its three component data planes. The numbers in the pixels blocks are digital counts and range from 0 to 255. The original block looked like a small collection of near achromatic triangles. This trio of 8x8 blocks is then transformed to the XYZ color space. Then the XYZ coordinates are converted to CIELAB values. Of course, before these values can actually be used by the Baseline JPEG algorithm the values must be normalized between 0 and 255. This was done by computing the maximum and minimum values that the gamut of the CRT was possible of producing for each of the three axes and using these values to scale the CIELAB coordinates. The last step in Fig 2.12 is the transformation from CIELAB to a normalized CIELAB. The values are shown as real numbers because the floating point data type was used to perform all of the computer calculations.

The next chain of computations shown in Fig 2.13 demonstrates the compression steps performed according to the JPEG algorithm. The algorithm can work on only one plane of data at a time. Therefore only one 8x8 block is shown at the top of the page. The example 8x8 block shown is the normalized b* plane from the bottom of Fig 2.12 with 128 subtracted from all of the values. This block of pixels is then sent the through the FDCT to produce an 8x8 block of FDCT coefficients. This block of coefficients is then quantized using the chrominance Q-Table shown in Fig 2.4. Finally, the block of quantized DCT coefficients is Huffman encoded to produce a bit string. Below the bit string is a breakdown of how the DC and AC terms were encoded. The bit string is the compressed form of the original 8 x 8 block of normalized a* data shown at the top of the chain. It is only 12 bits long and the resulting compression is over 40 to 1. Of course this entire

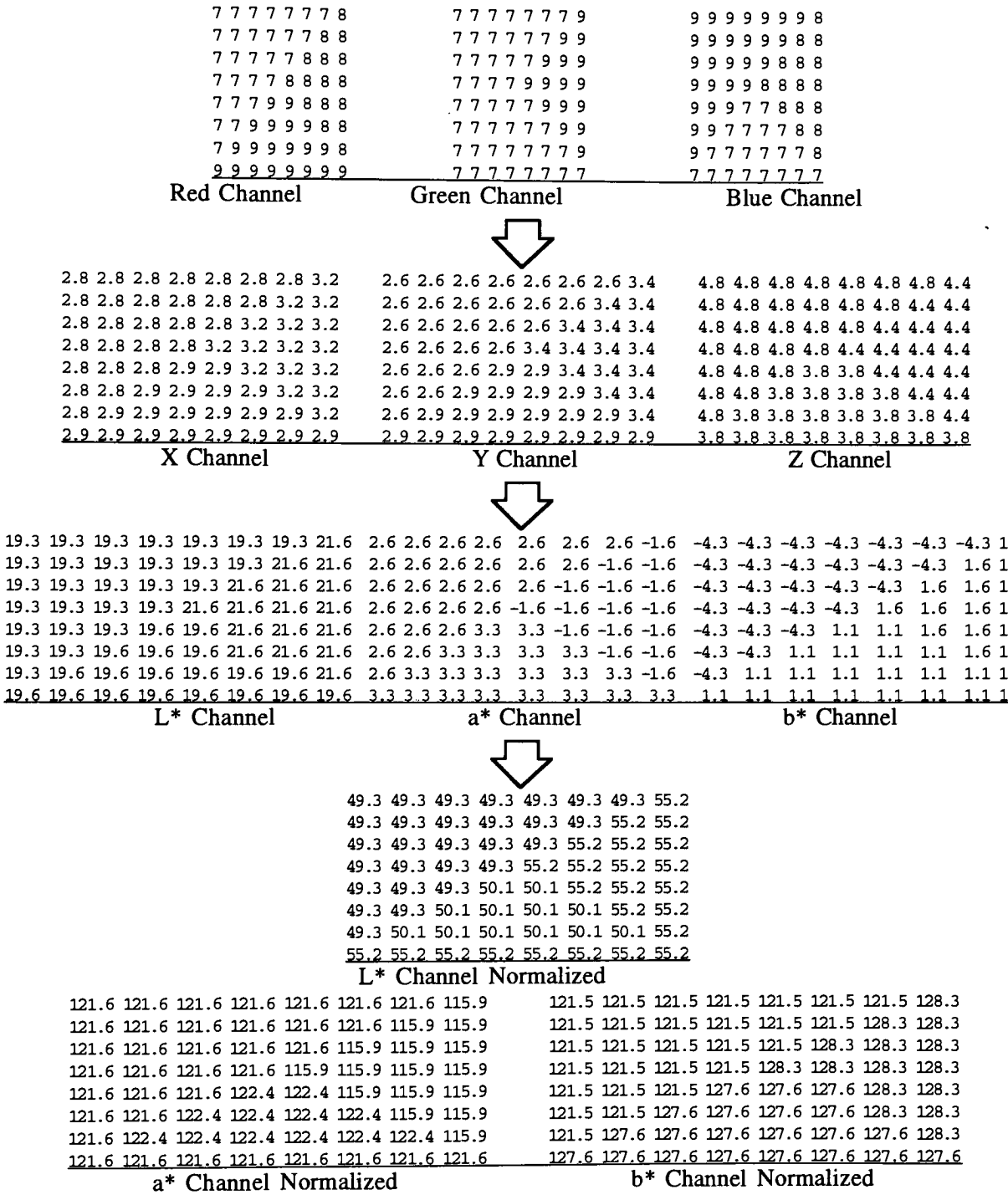



Fig. 2.12 Flowchart showing color space conversions performed when converting an 8x8 block of RGB pixels to the corresponding CIELAB values.

procedure would have to be repeated for the L* and b* blocks. Although, the L* data would be quantized with the luminance Q-Table instead of the chrominance Q-Table.

The JPEG decompression steps are also shown in Fig. 2.13. The initial step is decode the bit string, then this block of data is dequantized. Finally, the dequantized 8 x 8 block is processed through the IDCT and out comes a reconstructed block of a* values. There are some errors in the reconstructed 8 x 8 block of a* coordinates, but they are relatively minor. These a* values, and also the corresponding L* and b* values, could then be back transformed using the equations in Appendix C to yield device coordinates.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -7.5 | -7.5 | -7.5 | -7.5 | -7.5 | -7.5 | -7.5 | 0.3 |
| -7.5 | -7.5 | -7.5 | -7.5 | -7.5 | -7.5 | 0.3 | 0.3 |
| -7.5 | -7.5 | -7.5 | -7.5 | -7.5 | 0.3 | 0.3 | 0.3 |
| -7.5 | -7.5 | -7.5 | -7.5 | 0.3 | 0.3 | 0.3 | 0.3 |
| -7.5 | -7.5 | -7.5 | -0.4 | -0.4 | 0.3 | 0.3 | 0.3 |
| -7.5 | -7.5 | -0.4 | -0.4 | -0.4 | -0.4 | 0.3 | 0.3 |
| -7.5 | -0.4 | -0.4 | -0.4 | -0.4 | -0.4 | -0.4 | 0.3 |
| -0.4 | -0.4 | -0.4 | -0.4 | -0.4 | -0.4 | -0.4 | -0.4 |

**Normalized b* with 128 Subtracted.**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 119.6 | 120.1 | 120.9 | 122.1 | 123.4 | 124.5 | 125.4 | 125.9 |
| 120.1 | 120.6 | 121.5 | 122.6 | 123.9 | 125.0 | 125.9 | 126.4 |
| 121.0 | 121.4 | 122.3 | 123.5 | 124.7 | 125.9 | 126.8 | 127.2 |
| 122.1 | 122.6 | 123.5 | 124.6 | 125.9 | 127.0 | 127.9 | 128.4 |
| 123.4 | 123.9 | 124.7 | 125.9 | 127.1 | 128.3 | 129.1 | 129.6 |
| 124.5 | 125.0 | 125.9 | 127.0 | 128.3 | 129.4 | 130.3 | 130.8 |
| 125.4 | 125.9 | 126.8 | 127.9 | 129.1 | 130.3 | 131.2 | 131.6 |
| 125.9 | 126.4 | 127.2 | 128.4 | 129.6 | 130.8 | 131.6 | 132.1 |

**Reconstructed Block of 8x8 Normalized b* pixels.**

FDCT

IDCT + 128

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -24.9 | -16.6 | 0.8 | -1.7 | 0.0 | -0.5 | 0.1 | -0.1 |
| -14.7 | -3.6 | 7.0 | 0.0 | 1.2 | 0.0 | 0.4 | 0.0 |
| -0.8 | 7.6 | 3.3 | -4.3 | 0.3 | -0.9 | 0.0 | -0.2 |
| -1.7 | 0.0 | -3.6 | -3.6 | 2.7 | 0.0 | 0.4 | 0.0 |
| 0.0 | 1.4 | -0.3 | 2.8 | 3.3 | -1.9 | 0.1 | -0.3 |
| -0.4 | 0.0 | -0.9 | 0.0 | -1.4 | -3.6 | 1.3 | 0.0 |
| -0.1 | 0.4 | 0.0 | 0.5 | -0.1 | 1.2 | 3.3 | -0.6 |
| -0.2 | 0.0 | -0.1 | 0.0 | -0.4 | 0.0 | -0.3 | -3.7 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -17 | -18 | 0 | 0 | 0 | 0 | 0 | 0 |
| -18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Forward Discrete Cosine Transform Coefficients**

**Decoded and Dequantized Values**

Quantize

Dequantize

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

```
100001001000

(10,00:  DC Category 2, Amplitude -3)
(01,0:   Runlength 0, AC Category 1)
(01,0:   Runlength 0, AC Category 1)
(00:  End of Block)
```

**FDCT Coefficients Quantized Using Chrominance Q-Table**

**Huffman Encoded Bit Stream**

Fig. 2.13 Computational Example for the compression and decompression of the normalized 8x8 block of b* pixels from Fig. 2.12. The compression steps are shown going down the left side of the page and decompression steps are shown going up the right side of the page.

# 3. Experimental

The original images, the electronic imaging system, viewing conditions and other factors are covered in the paragraphs on the environment. In the following section, the details for deriving the visually lossless compression thresholds for each of the color spaces are discussed. Finally, the supra-threshold inter-color space comparisons of equivalent compressions are presented.

## 3.1. Environment

Three of the experimental images were obtained in digital form. The birds image is from the Kodak Photo CD Photo Sampler and the fruit and musicians images are SCID images. The forth image, of a pasture, was a photographic print digitized using a UMAX UC 1200 S scanner at a resolution of 200 dots per inch. Black and white representations of these images are shown in Images 3.1 to 3.4. All of the images were 768 by 512 pixels and approximately 1.17 MBytes in size. The images had a precision of 8 bits per channel and three channels per pixel.

The JPEG algorithm was written as an ANSI C program on a Hewlett-Packard 9000/375 system. The central portion of this program can be found in Appendix B. The JPEG program was then incorporated into the color reproduction software or CRS used at the Munsell Color Science Laboratory. All image manipulations were then carried out using this software. This processing can be summarized as a color space transformation, JPEG compression followed directly by decompression, and finally a color space transformation back to the device color space for display. The CRS software was already written to accomplish the first and last color space transformations. Some additional image processing was performed on a SUN 3/160 workstation. A diagram of the entire system used to compress and display the experimental images is shown in Fig. 3.1.

Image 3.1 Birds

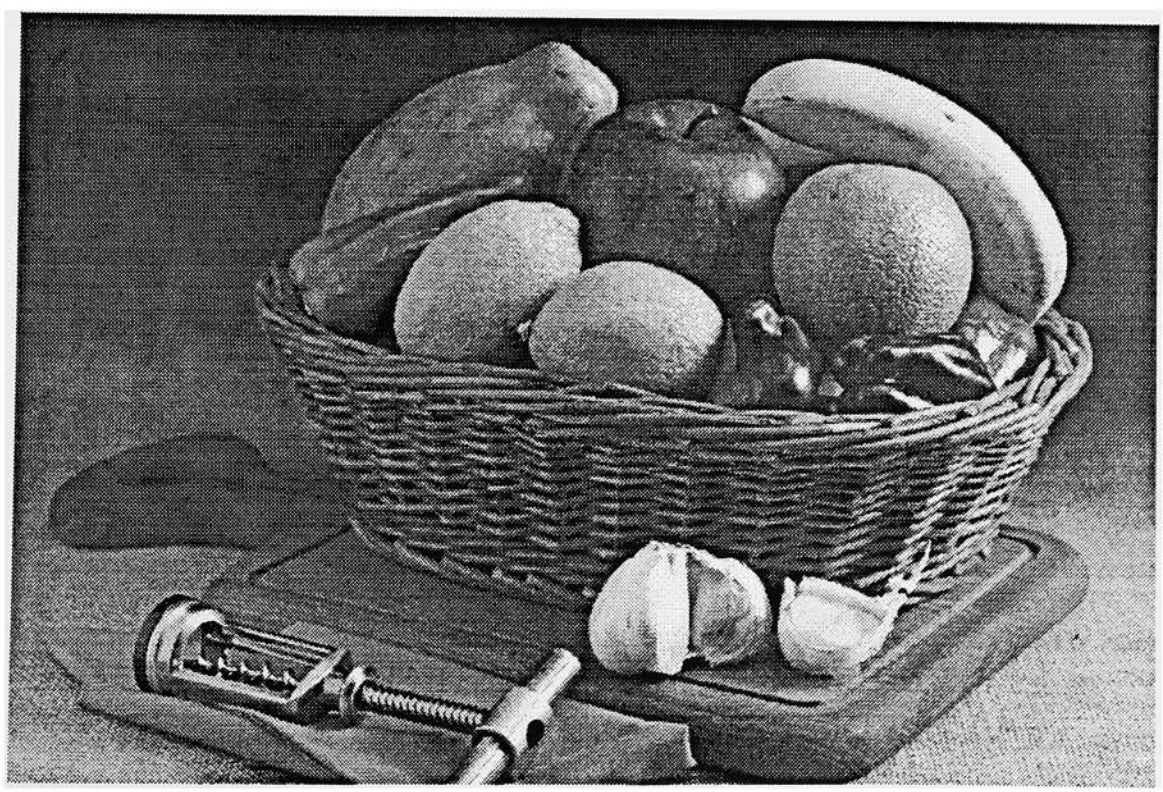


Image 3.2 Fruit

Image 3.3 Musicians

Image 3.4 Pasture

The images were then displayed on a PIXAR II Image Computer as the original. The CRT, a Sony GDM-1950, was colorimetrically calibrated using a previously published physical model. The white point of the monitor was calibrated to D90. This white point was chosen because the red gun of the particular monitor being used was especially weak. In addition, the 12 bit PIXAR look-up tables were set to a gamma of 1/2.3 in order to linearize the CRT gamma of approximately 2.3. This results in an approximately linear relationship between the device digital counts and luminance.

The colorimetric characterization of the CRT was carried out according to an established technique (Motta 1991) and was tested using an independent data set consisting of six step cyan, magenta, and yellow ramps and also 14 randomly generated color patches. This independent data set evinced that the CRT was colorimetrically calibrated to an average $DE^*_{ab}$ of about 3.3. The details of the CRT calibration can be found in Appendix D. The results of this calibration yielded a CRT which was colorimetrically characterized to an average $DE^*_{ab}$ of 3.3. Typically, the characterization process outlined in Appendix D results in a CRT that is calibrated to an average $DE^*_{ab}$ of about 1. The unusually high value occured because the monitor suffered an unexplained spontaneous decalibration in the middle of the experimental stage of the research. The monitor was recalibrated as near as possible to the original settings but time constraints made it impractical to attempt further adjustments. Furthermore, because of the nature of the psychophysical experiments, the calibration error was acceptible.

The observers were then seated three feet from the CRT in a darkened room. The pixel size of a PIXAR II pixel is roughly 0.026 cm or 0.011 in. At a distance of three feet, this meant that the images were three pixel channels per $minute^2$ or one RGB pixel per $minute^2$. The specifics of this calculation can be found in Appendix H. While this pixel per $minute^2$ level is considerably less than a perfect display, it is an acceptable level given the state of current display technology (Klein and Carney 1991). A picture of the viewing conditions used for both of the psychophysical experiments is shown in Fig. 3.2.

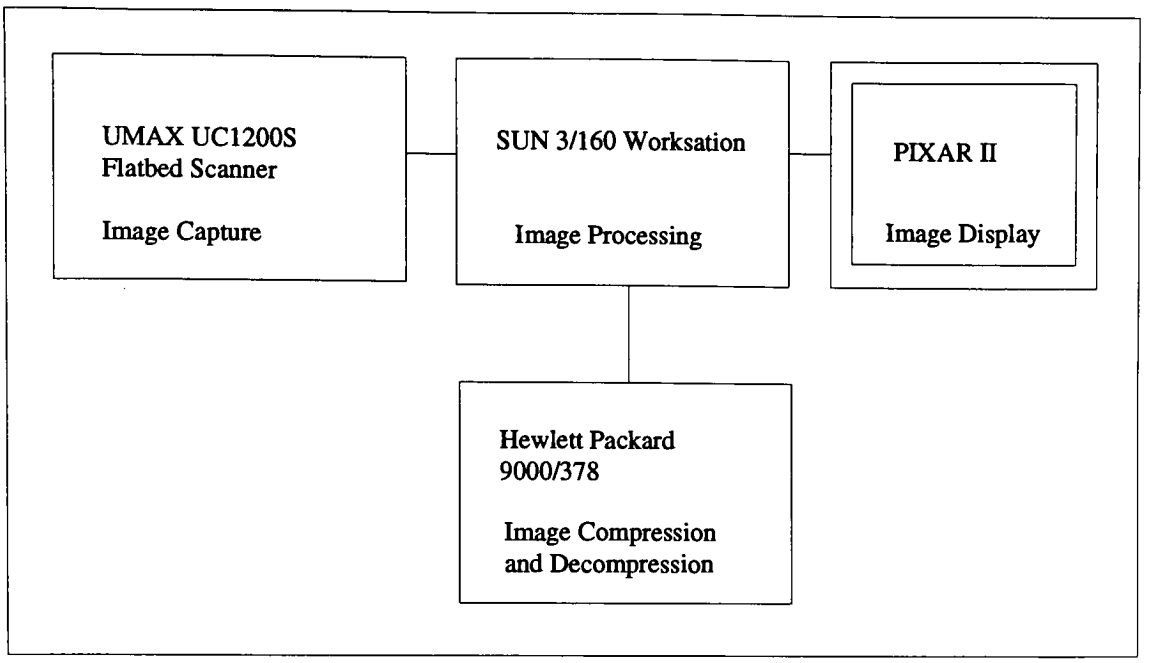



Fig. 3.1 Diagram of the electronic imaging system used to perform the computations and the psychophysics.

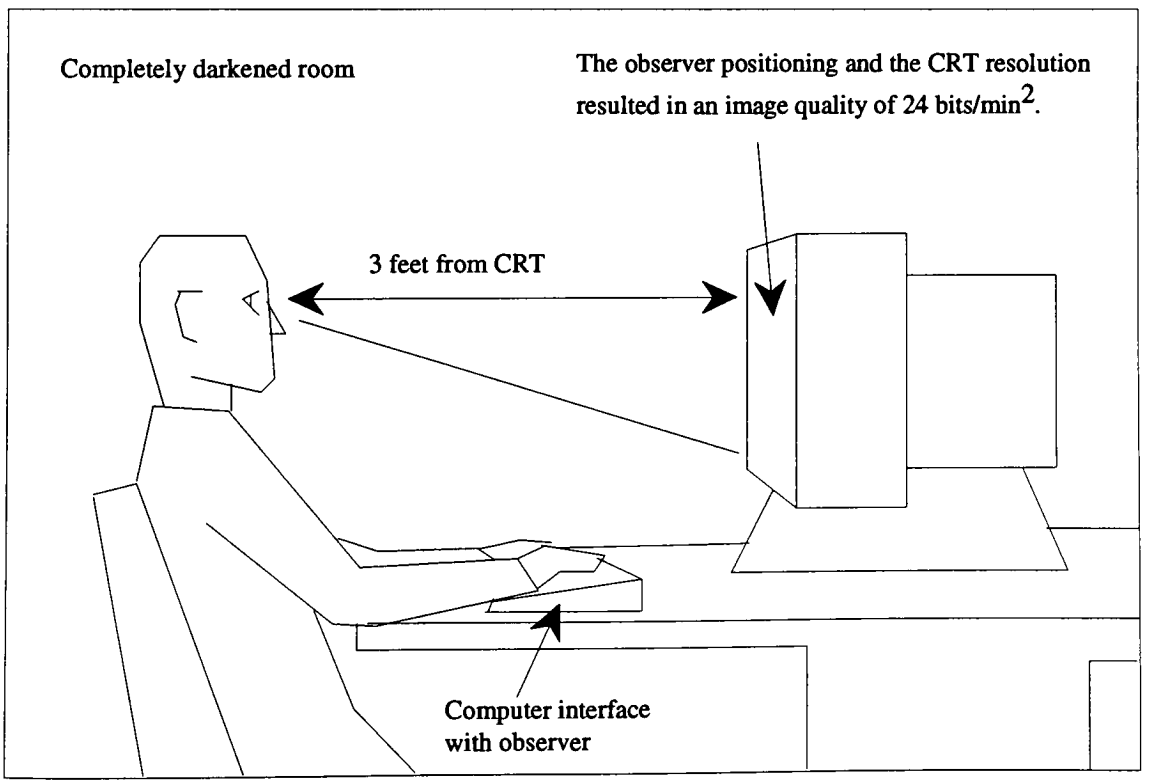



Fig. 3.2 Experimental viewing conditions for experiments one and two.

Finally, having established all of the other variables a preliminary experiment was performed to test the experimental set-up and also to select the compression levels to be used in the following experiments. The details of this preliminary experiment can be found in Appendix E. The set-up was deemed appropriate and images were compressed to levels that were determined to be close to the visual threshold.

## 3.2. Experiment One

Unlike previously reported psychophysical experiments utilizing magnitude estimation or scaling techniques, the first psychophysical experiment was a forced-choice paired-comparison experiment (Stein, Watson, and Hitchner 1989 and Wallace, Vivian, and Poulsen 1988). The two independent variables were color space and the bit level of compression of the image. The objective was to determine the threshold for visually lossless image compression for each of the color spaces. This visual threshold could then be quantified based on the size in bits of the compressed image. The results of this experiment would allow for comparisons of the visual thresholds for a given image for each of the color spaces. Furthermore, the color spaces could be ranked according to the relative differences in the thresholds.

The experiment consisted of the four images listed in the previous section. Each of the images was then compressed to eight different bit levels in each of the six color spaces. This resulted in a total of 168 images to be examined by all of the observers. The eight compression levels were selected according the results of the preliminary psychophysical experiment. The maximum and minimum levels of compression for each of the images and color spaces is shown in Table 3.1. The values listed in Table 3.1 are the compression level of the image expressed using the various compression measures introduced in the previous section. The bit size of the compressed image, compression ratio, bits/pixel and bits/min$^2$ are all different ways of expressing the same level of compression. This table

provides some idea of the range of images used in experiment one. A total of 24 observers participated in the first experiment. All of these observers were familiar with the idea of visual experiments and a majority had participated in psychophysical experiments before. However, the observers had widely different backgrounds and experience in detecting errors in digital images. For instance, a few observers had examined compressed images before the experiment whereas other observers had never before looked for errors in digital images.

Table 3.1 List of Images used for Experiment One.

| Image | KBytes for Compressed Image | Compression Ratio | Bits/Pixel | Bits/Min$^2$ |
|---|---|---|---|---|
| Birds LAB | 138 - 262 | 8.54 - 4.50 | 0.94 - 1.78 | 2.82 - 5.34 |
| Birds LCh | 152 - 313 | 7.77 - 3.77 | 1.03 - 2.12 | 3.09 - 6.36 |
| Birds LUV | 139 - 254 | 8.48 - 4.64 | 0.94 - 1.72 | 2.82 - 5.16 |
| Birds RGB | 159 - 431 | 7.42 - 2.74 | 1.08 - 2.92 | 3.24 - 8.76 |
| Birds XYZ | 160 - 338 | 7.37 - 3.49 | 1.09 - 2.30 | 3.27 - 6.90 |
| Birds YIQ | 138 - 244 | 8.54 - 4.83 | 0.94 - 1.66 | 2.82 - 4.98 |
| Fruit LAB | 164 - 316 | 7.19 - 3.73 | 1.11 - 2.14 | 3.33 - 6.42 |
| Fruit LCh | 156 - 371 | 7.56 - 3.18 | 1.06 - 2.52 | 3.18 - 7.56 |
| Fruit LUV | 148 - 319 | 7.97 - 3.69 | 1.00 - 2.16 | 3.00 - 6.48 |
| Fruit RGB | 190 - 545 | 6.21 - 2.16 | 1.29 - 3.69 | 3.87 - 11.07 |
| Fruit XYZ | 181 -462 | 6.51 - 2.55 | 1.23 - 3.13 | 3.69 - 9.39 |
| Fruit YIQ | 156 - 283 | 7.56 - 4.17 | 1.06 - 1.92 | 3.18 - 5.76 |
| Musicians LAB | 150 - 294 | 7.86 - 4.01 | 1.02 - 1.99 | 3.06 - 5.97 |
| Musicians LCh | 171 - 462 | 6.89 - 2.55 | 1.16 - 3.13 | 3.48 - 9.39 |
| Musicians LUV | 152 - 293 | 7.77 - 4.02 | 1.03 - 1.99 | 3.09 - 5.97 |
| Musicians RGB | 195 - 443 | 6.04 - 2.66 | 1.03 - 3.00 | 3.09 - 9.00 |
| Musicians XYZ | 183 - 345 | 6.44 - 3.42 | 1.24 - 2.34 | 3.72 - 7.02 |
| Musicians YIQ | 149 - 244 | 7.91 - 4.83 | 1.01 - 1.66 | 3.03 - 4.98 |
| Pasture LAB | 149 - 357 | 7.91 - 3.30 | 1.01 - 2.42 | 3.03 - 7.26 |
| Pasture LCh | 159 - 415 | 7.42 - 2.84 | 1.08 - 2.82 | 3.24 - 8.46 |
| Pasture LUV | 148 - 342 | 7.97 - 3.45 | 1.00 - 2.32 | 3.00 - 6.96 |
| Pasture RGB | 216 - 663 | 5.46 - 1.79 | 1.47 - 4.50 | 4.41 - 13.50 |
| Pasture XYZ | 207 - 535 | 5.69 - 2.20 | 1.40 - 3.63 | 4.20 - 10.89 |
| Pasture YIQ | 155 - 329 | 7.61 - 3.58 | 1.05 - 2.23 | 3.15 - 6.69 |

The observers were shown the original image and a compressed-decompressed version of the original one at a time on the monitor. The subjects were then instructed to

select the image that was not the original. They were also told that the image that was not the original was of a lower image quality. For this experiment, loss of image quality was defined to be any perceptible degradation in resolution or color of the image. This procedure was then repeated 168 times for each of the experimental images. The order of the images, as well as whether the original was shown first or not, was randomized for each of the observers. The exact instruction sheet which was provided to the subject for experiment one can be found in Appendix F.

## 3.3. Experiment Two

The second psychophysical experiment performed was very similar to the first experiment. The same hardware and viewing conditions were used. Moreover, the paired-comparison technique was used again. The same four images were used again but this time the compressed and decompressed images for the different color spaces were compared to each other instead of to the original image. For this experiment, all of the images were compressed to the same bit level for each color space. This aim bit level was selected by using the results of the first experiment. The best color compression space for each of the images was compressed to a level just above the visually lossless threshold. The subsequent size, in bits, of the image compressed in the best color compression space was used as the aim bit level.

Therefore, there were four images compressed and decompressed in six different color spaces to the same level of compression for a total of 24 images. A list of the images and their compression levels is shown in Table 3.2. Because the JPEG algorithm is not a driven algorithm, the aim bit levels had to be achieved iteratively. Nevertheless, all of the test images used had less than a plus or minus 0.0025 error in the final bits/pixel compression level .

Table 3.2 List of Images used for Experiment Two.

| Image | Bits for Compressed Image | Compression Ratio | Bits/Pixel | Bits/Min2 |
|---|---|---|---|---|
| Birds LAB | 1108622 | 8.5125 | 0.9398 | 2.8194 |
| Birds LCh | 1109117 | 8.5087 | 0.9402 | 2.8206 |
| Birds LUV | 1109256 | 8.5077 | 0.9403 | 2.8209 |
| Birds RGB | 1111409 | 8.4912 | 0.9421 | 2.8263 |
| Birds XYZ | 1110360 | 8.4992 | 0.9412 | 2.8236 |
| Birds YIQ | 1111489 | 8.4906 | 0.9422 | 2.8266 |
| Fruit LAB | 1108103 | 8.5165 | 0.9394 | 2.8182 |
| Fruit LCh | 1110310 | 8.4996 | 0.9412 | 2.8236 |
| Fruit LUV | 1108418 | 8.5141 | 0.9396 | 2.8188 |
| Fruit RGB | 1109692 | 8.5043 | 0.9407 | 2.8221 |
| Fruit XYZ | 1108744 | 8.5116 | 0.9399 | 2.8197 |
| Fruit YIQ | 1109410 | 8.5065 | 0.9405 | 2.8215 |
| Musicians LAB | 1202774 | 7.8462 | 1.0196 | 3.0588 |
| Musicians LCh | 1204882 | 7.8345 | 1.0214 | 3.0642 |
| Musicians LUV | 1201089 | 7.8572 | 1.0182 | 3.0546 |
| Musicians RGB | 1200473 | 7.8612 | 1.0176 | 3.0528 |
| Musicians XYZ | 1205025 | 7.8315 | 1.0215 | 3.0645 |
| Musicians YIQ | 1202779 | 7.8461 | 1.0196 | 3.0588 |
| Pasture LAB | 1152675 | 8.1872 | 0.9771 | 2.9313 |
| Pasture LCh | 1150326 | 8.2039 | 0.9751 | 2.9253 |
| Pasture LUV | 115456 | 8.1732 | 0.9788 | 2.9364 |
| Pasture RGB | 1150588 | 8.2021 | 0.9753 | 2.9259 |
| Pasture XYZ | 1151919 | 8.1926 | 0.9765 | 2.9295 |
| Pasture YIQ | 1151638 | 8.1946 | 0.9763 | 2.9289 |

The images were then judged one at a time relative to each other. This time the observers were instructed to select the image with the better image quality. Once again, the precise instructions given to the observers can be found in Appendix F. This meant that, altogether, the observer had to make 60 comparisons. The experiment was run using a total of 25 observers. These observers, as was the case for the subjects in the first experiment, had mixed levels of experience.

# 4. Results and Discussion

The following three sections present the results of the two psychophysical experiments. In addition to using probit and logistic analysis to assess the results, several other quantitative measures are presented. Image processing and colorimetric error metrics are used to further investigate the results of the second experiment. A sequence of error images are also presented for the second experiment. The final section addresses various topics that provide further insight into the specifics of the results.

## 4.1. Experiment One

The responses for all of the observers were merged together to form one data set. This set of data consisted of a list of image and color space names, eight levels of compression for each of the 24 image and color space combinations, the number of subjects that evaluated that image, and finally how many times that image was correctly judged to be the compressed image. A frequency could then be computed by dividing the number of correct identifications by the number of observers. This frequency reflects how perceptible the level of compression was to the observers. For instance, at the lower levels of compression it was very difficult to discriminate between the original and the compressed image. Consequently, the observers had to guess and the resulting frequency that the observers were correct was about 50 percent. On the other hand, the images that were highly compressed and had a considerable amount of deterioration in the image quality were very easy to differentiate from the original. The frequency that the observers were correct for these images was closer to 100 percent. The intermediate levels of compression had a frequency correct level somewhere between 50 and 100 percent.

These frequencies could then be plotted versus the level of compression measured in terms of bits/pixel Then probit analysis could be performed on the data in order to

determine the threshold for visually lossless compression. Probit analysis is a statistical technique in which a cumulative normal curve is fitted to an experimental data set (Bartleson 1984). This idea is illustrated in Fig 4.1 where the black diamonds are actual data points for the CIELAB LCh birds images and the line is the fitted cumulative normal curve. The threshold for visually lossless compression can then be determined by taking the mid-point of the fitted cumulative normal. In the case of Fig 4.1, the mid-point is 75 percent and the threshold is 1.24 bits/pixel.

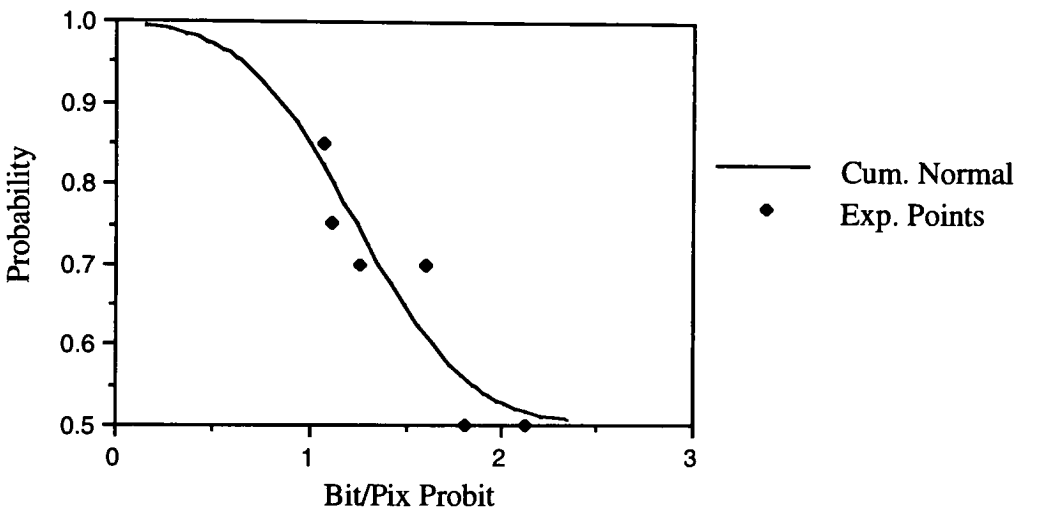



Fig. 4.1 Sample Probit Analysis Showing the Fitted Cummulative Normal Curve and Experimental Data Points.

### 4.1.1. Probit Analysis of Data

The probit analyses for the data from experiment one were performed using the SAS statistical package. This SAS probit analysis procedure begins by providing a statistical measure of whether it is appropriate to use probit analysis on the given data set. The two major measures are the log-likelihood and chi-squared tests (SAS 1990). In every instance, these two indicators supported the use of probit analysis for the data. In addition, the program provided a measure of the uncertainty of the mean or the threshold computed

by the program. This value, the standard deviation of the data or $\sigma$, could then be used to determine whether or not two different visual thresholds were statistically different from one another.

The results of the probit analyses of frequency observer correct versus bit per pixel level of compression are shown in Figs. 4.2 to 4.6. The Figs. 4.2 to 4.5 are for each of the individual images and Fig. 4.6 is an overall average of the results for all four images. The x axis in all five of these plots is an nominal scale denoting the six color spaces tested and the y axis is the compression level of the image measured bits per pixel. The thresholds of visually lossless compression are shown by small squares for each of the color spaces. These thresholds are the 75% level on the cumulative normal curve fitted to the data during the probit procedure. These five figures also have error bars around their thresholds expressing the uncertainties of the data. The error bars for Figs. 4.2 to 4.5 express the uncertainty of the observers responses and the error bars in Fig. 2.6 reflect the uncertainty of the mean of these images. These error bars are two standard errors and were computed using the sigma computed by the probit procedure. The exact equation for standard error is expressed:

$$\text{Standard Error} = \frac{\sigma}{\sqrt{n}} \tag{4.1}$$

The n in Eqn. 4.1 is the total number of observations in the data set. Plus or minus two standard errors were then computed and plotted around the thresholds in order to provide some measure of the uncertainty of the thresholds (Barry 1978). Fiducial limits could have also been used to express the uncertainties of the thresholds but, the SAS probit analysis procedure did not consistently provide these limits.

It should be noted that two means or thresholds are statistically different from one another if the threshold of one does not overlap the upper, or lower in some cases, reach of the other thresholds error bar and vice versa. For example in Fig. 4.2, the threshold for the

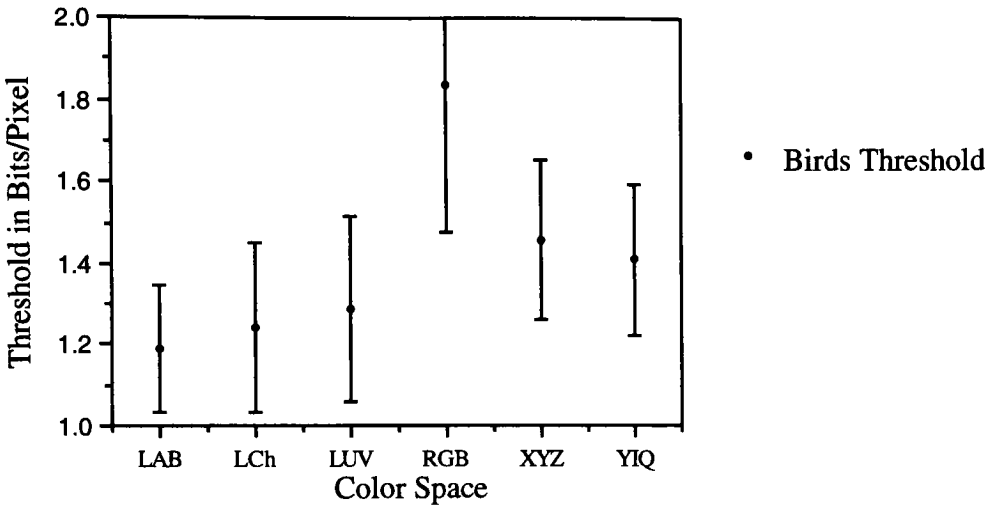



Fig. 4.2 Visually Lossless Thresholds of Compression in Bits/Pixel for the Birds Image.

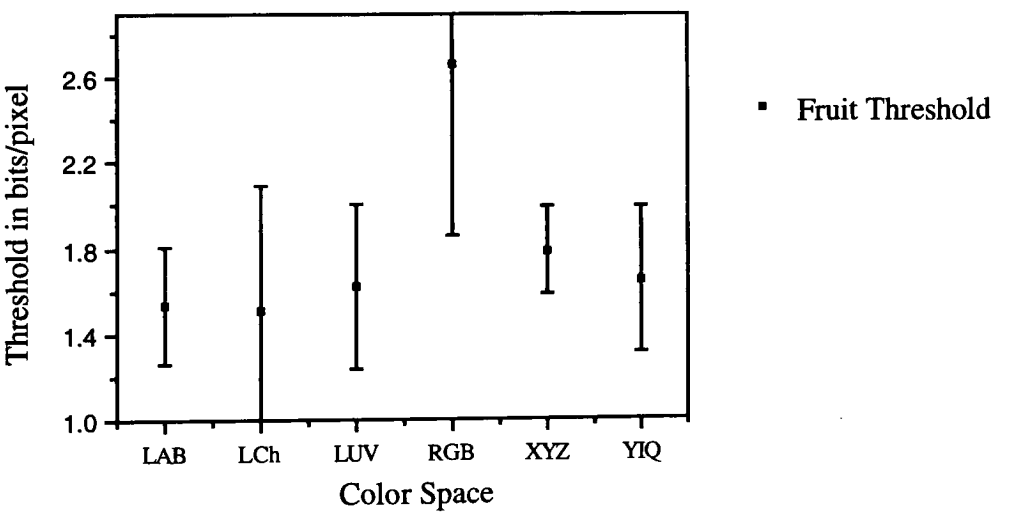



Fig. 4.3 Visually Lossless Thresholds of Compression in Bits/Pixel for the Fruit Image.

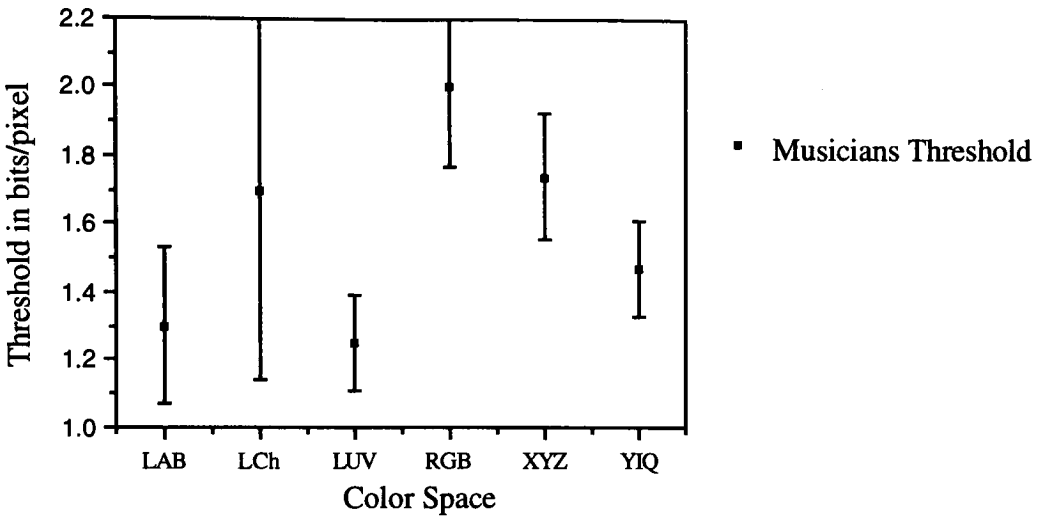



Fig. 4.4 Visually Lossless Thresholds of Compression in Bits/Pixel for the Musicians Image.

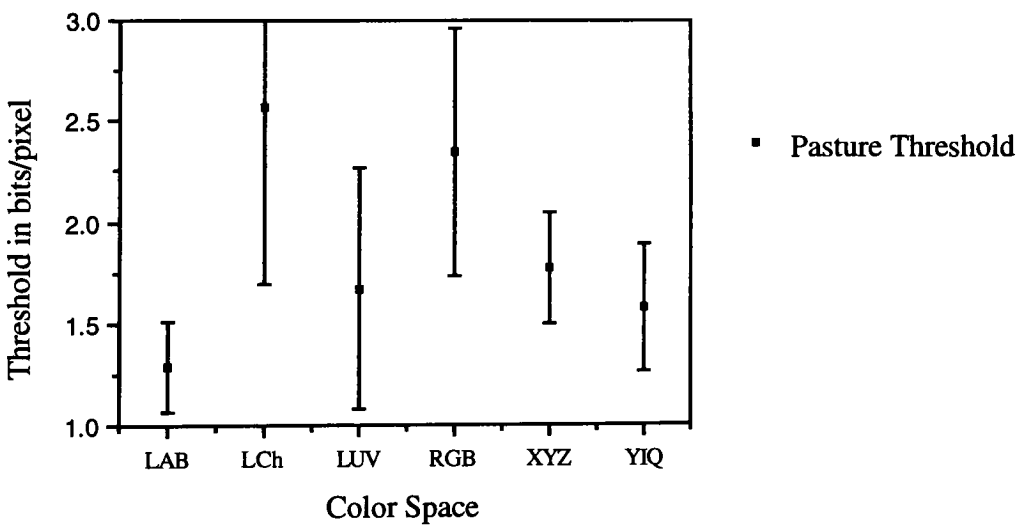



Fig. 4.5 Visually Lossless Thresholds of Compression in Bits/Pixel for the Pasture Image.

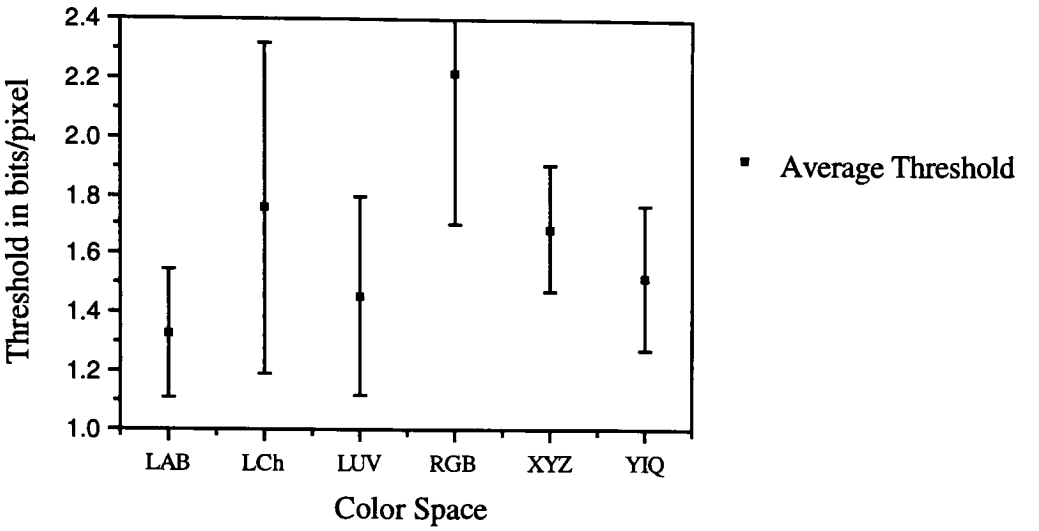



Fig. 4.6 Visually Lossless Thresholds of Compression in Bits/Pixel for the Average of the Four Images.

CIELAB color space is 1.190 bits per pixel and the lower reach of the YIQ error bar is 1.221 bits per pixel. Likewise, the threshold for the YIQ space is 1.409 bits per pixel and the upper reach of the CIELAB error bar is 1.347 bits per pixel. In neither case does any part of one color spaces error bar overlap on the other spaces threshold. Although it is acceptable for the error bars themselves to overlap as long as the error bars for one the color spaces does not overlap the mean threshold of the other. Therefore, it is statistically valid to conclude that in Fig. 4.2 the thresholds for the CIELAB and YIQ color spaces are indeed different.

There are several systematic trends present in Figs. 4.2 to 4.6. It is evident that the RGB color space was the worst compression space. In comparison, the CIELAB and CIELUV color spaces were the best color spaces for compression The YIQ and XYZ color spaces were close third and fourth, respectively. The CIELAB LCh color space was inconsistent in its performance. In some cases it did fairly well and in others it did poorly. Specific details of these results are discussed in the following sections.

The thresholds and calculated standard errors are also presented in table form in Tables 4.1 and 4.2. These tables list the results for each of the four images as well as the overall averages for all of the images. The two tables also clearly demonstrate the image dependent nature of the results. Therefore, it will be useful to examine the overall results for each of the image as well as for each color space. Using the values listed in Table 4.1, the row averages of the thresholds in bits per pixel are 1.404 for birds, 1.793 for fruit, 1.578 for musicians, and 1.870 for pasture. This means that, on the average, the birds image with its predominance of low frequencies compressed to the lowest level and the pasture image with its high frequencies was compressed the least. This result is consistent with the way that JPEG works and also with another author's results (Goodenow 1993).

Table 4.1 Visually lossless Thresholds for Compression in Bits/Pixel .

| Image | LAB | LCh | LUV | RGB | XYZ | YIQ |
|---|---|---|---|---|---|---|
| Birds | 1.19 | 1.24 | 1.29 | 1.84 | 1.46 | 1.41 |
| Fruit | 1.53 | 1.51 | 1.62 | 2.66 | 1.79 | 1.65 |
| Musicians | 1.30 | 1.70 | 1.25 | 2.00 | 1.74 | 1.47 |
| Pasture | 1.29 | 2.57 | 1.67 | 2.34 | 1.77 | 1.57 |
| Overall | 1.33 | 1.76 | 1.46 | 2.21 | 1.69 | 1.53 |

Table 4.2 Two Standard Errors in Bits/Pixel for the Thresholds in Table 4.1.

| Image | LAB | LCh | LUV | RGB | XYZ | YIQ |
|---|---|---|---|---|---|---|
| Birds | 0.16 | 0.21 | 0.23 | 0.36 | 0.19 | 0.19 |
| Fruit | 0.27 | 0.58 | 0.39 | 0.81 | 0.21 | 0.34 |
| Musicians | 0.23 | 0.56 | 0.14 | 0.24 | 0.18 | 0.14 |
| Pasture | 0.22 | 0.89 | 0.60 | 0.62 | 0.28 | 0.32 |
| Overall | 0.22 | 0.56 | 0.34 | 0.51 | 0.21 | 0.25 |

A similar analysis can be performed with the standard errors listed in Table 4.2. The row averages for these data are 0.223 for birds, 0.433 for fruit, 0.250 for musicians, and 0.485 for pasture. These values suggest that not only do low frequency images compress to a lower level but that there is less uncertainty in these thresholds. A probable explanation for this phenomenon is the variation in the observers' experience and

thoroughness. Relatively small compression artifacts in low frequency regions are much easier to identify than equivalent artifacts in high frequency regions. Consequently, the observer would either have to be experienced in looking for these deteriorations in high frequency regions or would have to be very patient in examining the image.

## 4.2. Experiment Two

The second experiment was performed to test the results of the first experiment using a different experimental design and also to investigate observer preferences for supra-threshold levels of compression. Specifically, a given image was compressed to approximately the same aim level of compression for all six color spaces. These compressions would then be compared one at a time to each other and the observers would select the preferred compression. The responses for all of the subjects could then be combined together and then the Law of Comparative Judgements was applied to derive an interval scale.

The first step in this analysis is calculating a frequency matrix for each of the images. A sample frequency matrix is shown for birds image in Table 4.3. The first row and the first column in the table list all six of the color spaces that were tested. The numbers entered into the table are the number of times that the column color space was selected as being better than the row color space. The diagonal elements of this matrix are zero because it is assumed that when the image is compared to itself neither image will be preferred over the other. The mirror positions above and below the diagonal should also sum to the total number of observers. For example, 10 observers selected CIELAB over CIELUV and 15 observers selected CIELUV over CIELAB and these two values sum to 25, the total number of observers.

Table 4.3 Frequency Matrix for Birds Image Using 25 Observers.

| | LAB | LCH | LUV | RGB | XYZ | YIQ |
|---|---|---|---|---|---|---|
| LAB | 0 | 0 | 15 | 0 | 0 | 5 |
| LCH | 25 | 0 | 23 | 0 | 0 | 21 |
| LUV | 10 | 2 | 0 | 0 | 0 | 4 |
| RGB | 25 | 25 | 25 | 0 | 3 | 25 |
| XYZ | 25 | 25 | 25 | 22 | 0 | 25 |
| YIQ | 20 | 4 | 21 | 0 | 0 | 0 |

Once the frequency matrix has been computed, the next step is to calculate the proportionality matrix. This operation is performed by dividing each of the elements in the frequency matrix by the total number of observations. The diagonal elements in this matrix are all set to 0.5. This is because if a given image was compared to itself it is assumed that half the time one of the images will be selected over the other. This is equivalent to saying that the observers will have to guess since there is no difference between the image and itself (Fairchild 1992). The proportionality matrix that could be derived from the data shown in Table 4.3 is shown in Table 4.4. Corresponding positions above and below the diagonal should sum to 1.0 for the proportionality matrix.

Table 4.4 Proportionality Matrix for Birds Using 25 Observers.

| | LAB | LCH | LUV | RGB | XYZ | YIQ |
|---|---|---|---|---|---|---|
| LAB | 0.5 | 0.0 | 0.6 | 0.0 | 0.0 | 0.2 |
| LCH | 1.0 | 0.5 | 0.9 | 0.0 | 0.0 | 0.8 |
| LUV | 0.4 | 0.1 | 0.5 | 0.0 | 0.0 | 0.2 |
| RGB | 1.0 | 1.0 | 1.0 | 0.5 | 0.1 | 1.0 |
| XYZ | 1.0 | 1.0 | 1.0 | 0.9 | 0.5 | 1.0 |
| YIQ | 0.8 | 0.2 | 0.8 | 0.0 | 0.0 | 0.5 |

Often the proportionality matrix will then be converted to a Z-Score matrix. The columns of this Z-Score matrix can then be summed to yield an interval scale. However, because of the large number of zeros and ones in the proportionality matrix, it is not necessarily appropriate to use this method of analysis. This is because the Z-score for zero is negative infinity and positive infinity for one. Instead, the proportionality matrix can be

transformed using the logisitic function (Bartleson 1984). This transformation can be computed using the following formula:

$$V = \ln\left[\frac{(f+0.5)}{(N+0.5-f)}\right] \tag{4.2}$$

where V is the computed logistic value, f is the proportionality matrix value and N is the total number of observations. In this case the frequency was a value between 0 and 1 and N was 25. The diagonals matrix of logistically transformed data will also be equal to zero. The logistic transformation of the proportionality matrix in Table 4.4 produces the logistic matrix shown in Table 4.5.

Table 4.5 Matrix of Logistically Transformed Data for Birds using 25 observers.

| | LAB | LCH | LUV | RGB | XYZ | YIQ |
|---|---|---|---|---|---|---|
| LAB | 0.00 | -3.93 | 0.39 | -3.93 | -3.93 | -1.32 |
| LCH | 3.93 | 0.00 | 2.24 | -3.93 | -3.93 | 1.56 |
| LUV | -0.39 | -2.24 | 0.00 | -3.93 | -3.93 | -1.56 |
| RGB | 3.93 | 3.93 | 3.93 | 0.00 | -1.86 | 3.93 |
| XYZ | 3.93 | 3.93 | 3.93 | 1.86 | 0.00 | 3.93 |
| YIQ | 1.32 | -1.56 | 1.57 | -3.93 | -3.93 | 0.00 |
| Sum | 12.72 | 0.13 | 12.06 | -13.86 | -17.58 | 6.54 |

Summing the columns in Table 4.5 produces an interval scale for comparing the different color spaces compressions. This interval scale is an approximate image quality metric of the compressed images. The higher the scale value, the better the perceived image quality for that color space compression and the lower the value, the worse the image quality. For instance, the CIELAB and CIELUV compressions of the birds image had the highest scale values, and therefore were judged to be the best compressions of that image. The XYZ and RGB compressions, on the other hand, had the lowest scale values and were judged to be the worst compressions of the birds image.

### 4.2.1. Analysis of the Logistically Transformed Data

The analysis procedure outlined in the previous section for the birds image was repeated for all of the experimental images. In addition, an overall analysis was done on the pooled data for all four images for an overall interval scale ranking. The results of all of these analyses is listed in Table 4.6.

Table 4.6 Logistic interval scales derived for each of the color spaces and images.

| Image | LAB | LCh | LUV | RGB | XYZ | YIQ |
|---|---|---|---|---|---|---|
| Birds | 12.72 | 0.13 | 12.06 | -13.86 | -17.58 | 6.54 |
| Fruit | 15.52 | 4.83 | 10.36 | -10.66 | -19.66 | -0.39 |
| Musicians | 11.59 | 4.65 | 10.36 | -12.93 | -15.90 | 2.24 |
| Pasture | 8.61 | -3.90 | 9.95 | -7.31 | -15.14 | 7.87 |
| Overall | 14.06 | 0.86 | 13.20 | -13.04 | -19.66 | 4.57 |

The results shown in Table 4.6 show that the nonlinear color spaces, CIELAB and CIELUV, were judged to be the best color spaces for compression for all four experimental images. The CIELAB LCh and YIQ compressions were tied for third and the RGB image

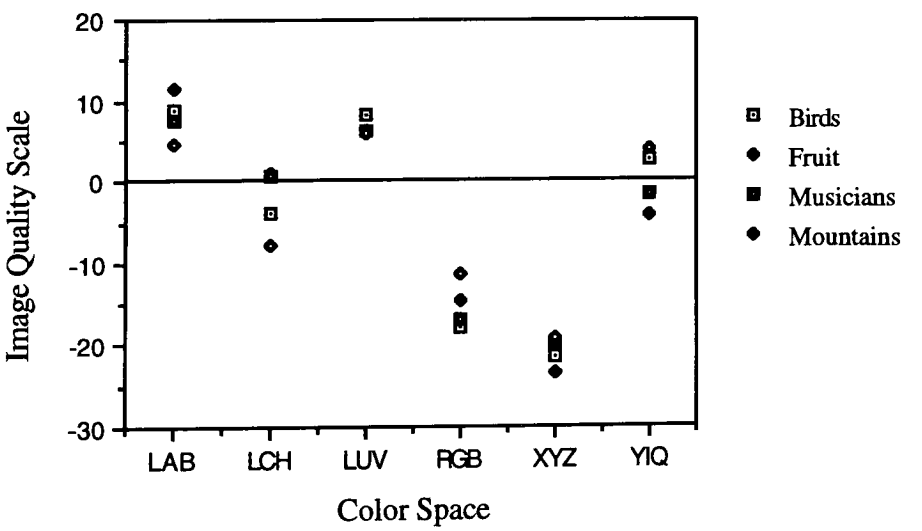



Fig. 4.7 Image Quality Scales Derived Using Logistic Analysis for the Six Color Spaces.

was fifth. Lastly, the XYZ color space compression was ranked as the worst in every single case. These results were also plotted in Figs. 4.7 and 4.8. For both of these plots, the x axis is the nominal scale of color spaces and the y axis is the logistic interval scale. The line drawn at the zero logistic scale value has no special significance and is only included to provided a horizontal element of reference for evaluating the figures.

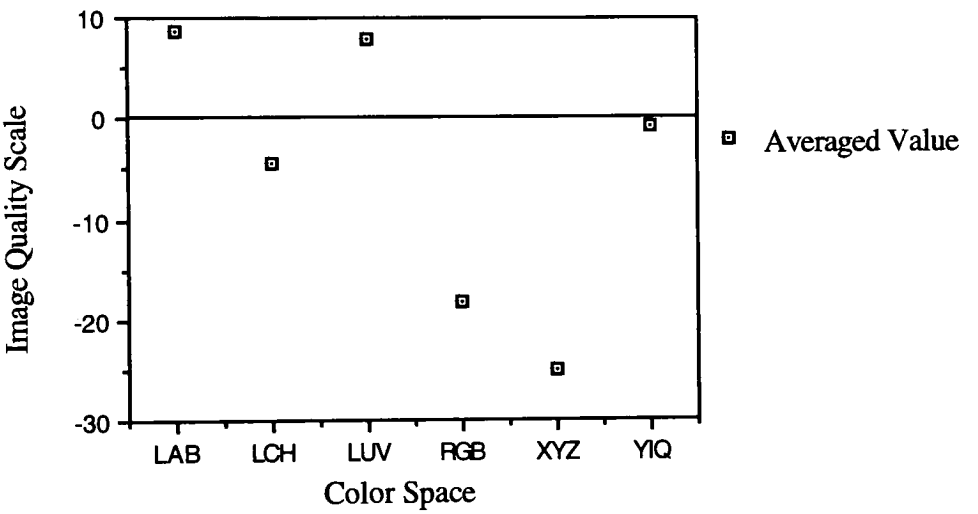



Fig. 4.8 Overall Image Quality Scale Derived Using Logistic Analysis.

The results from Tables 4.6 and Figs. 4.7 and 4.8 are partially in agreement with the results from the first experiment. The CIELAB and CIELUV images were judged best, followed by the CIELCh and YIQ color image compressions. However, the XYZ compressions were judged to be worse than the RGB compressions. It should be noted that the first experiment was designed to determine visually lossless thresholds for each of the color spaces. In comparison, the second experiment consisted of images that were all compressed to some supra-threshold level of compression. It would therefore seem likely that the at low levels of compression the XYZ color space does better than the RGB space, but at higher levels of compression this relationship is reversed. This is probably because

at higher levels of compression, the X and Z planes are so severely quantized that there is a drastic deterioration in the image quality of the XYZ image relative to the RGB image.

### 4.2.2. Image Processing Measures of Error

These psychophysical results can also be investigated in terms of various numeric measures. The various error metrics discussed in the following section are fairly common measures used in image processing. Despite the fact that there is evidence that these measures of error do not correlate perfectly with the perceived image error (Griswold 1980), they are presented here to supplement the psychophysical results and also to more fully investigate the compression process. The five measures of error used in this section are maximum error, minimum error, root mean square error, peak signal to noise ratio and difference in image entropy. For all of the following calculations, the RGB formats of all of the images were used. As a result, the individual channels ranged from 0 to 255.

The maximum and minimum errors, or max and min errors, are just the largest and the smallest errors between the original image and the compressed and decompressed image. For the following two sections, the original image will be referred to as the standard image and the compressed and decompressed image will be called the test image. Subsequently, the max error would be the largest positive error value resulting when the test image was subtracted from the standard image and the min error would be the largest negative value resulting from the same operation. Of course, there are three separate planes to the image and therefore this value is actually averaged over the three channels to yield a single number. These errors could theoretically range anywhere between 255 and -255, but usually tended to be much less. The max and min errors provide an indication of the extent and direction of the error.

The root mean square error or RMS error provides a general measure of the image error. This error does not include any quantification of the tendencies in image error, just the magnitude of the error. The RMS error can be computed using the formula:

$$\text{Root Mean Square Error} = \sqrt{\frac{\sum_{p=0}^{T}\left(\left(\frac{Rt+Gt+Bt}{3}\right)-\left(\frac{Rs+Gs+Bs}{3}\right)\right)^2}{T}} \tag{7.3}$$

where p is the index used to access individual pixels and T is the total number of pixels in the image. The Rt, Gt, and Bt variables are the three channels of the given test pixel and Rs, Gs, and Bs variables are the three channels for the corresponding standard pixel. The quantity under the square root sign is simply the mean square error or MSE between the two images. The RMS error could range from 0 for no error in the image to a possible worst case value of 255 for nothing but error in the test image. Generally, the RMS error was in the single digits for minor errors or the teens for larger errors.

The peak signal to noise ratio or PSNR is a metric of the amount of distortion which has occurred in the original image. The PSNR value is supposed to reflect how much noise has been introduced to an original image that has been transmitted in some way. In compression research, this value is used to indicate how much noise has been generated in the original image by the compression algorithm (Rabbani and Jones 1991). The standard way of computing PSNR is purely by convention and the units of PSNR are decibels. The equation for computing PSNR is as follow:

$$\text{Peak Signal to Noise Ratio} = 20 \cdot \text{Log}\left(\frac{MSE}{255}\right) \text{ decibels} \tag{7.4}$$

where MSE is the Mean Squared Error between the test and standard images computed as the quantity under the square root sign in Eqn. 7.3. Typically, the PSNR value ranges

from around 25 for images with considerable noise added to the original to about 35 for images with diminutive amounts of noise.

Lastly, the entropy of each of the images was computed. Image entropy, expressed in terms of bits/pixel, is often used in image compression studies (Melnychunk, Barry, and Mathieu 1990). The entropy of an image is essentially a measure of the amount of information in an image, where the information is modeled using probability theory (Mauro 1985). In order to visualize this, picture a 8 bit per pixel monochrome image as a dice game. Each gray level in this image would be determined according to a roll of the die. If the image is a perfectly random distribution of all possible gray values, then a 256 sided die would be needed to generate this image. At least 8 bits would be needed to communicate the results of each toss of the die. The entropy of this image would be 8 bits per pixel and would reflect that there is a large amount of information in this image. However, 256 sided dice are hard to come by so suppose a regular 6 sided dice is used instead. Using the 6 sided die still results in a random image, but now there are only 6 possible gray levels in the resulting images. Three bits per pixel would be sufficient to communicate the results of each toss of the die and the entropy of this image would be around three bits per pixel. Therefore, there is less information in the image generated by the 6 sided dice than in the image generated by the 256 sided dice.

Hence, computing the entropy of an image is some what analogous to determining how many sides are needed on a die in order to create a given image. Additional conditions and extensions can be made on this premise but, for this research the simplest measure of entropy will be used (Raisbeck 1965). The exact equation used to calculate image entropy is as follows:

$$\text{Entropy} = -\sum_{R=0}^{255}\sum_{G=0}^{255}\sum_{B=0}^{255}\left(\left(\frac{H}{T}\right)\cdot \text{Log}_2\left(\frac{H}{T}\right)\right) \tag{7.5}$$

where R,G, and B are used to index through the multi-dimensional histogram of the image, H is the total number of pixels with R,G, and B as their digital counts in the image, and T is number pixels in the image. In order to implement this equation, the resolution of the three-dimensional histogram was reduced from 1x1x1 digital count cubic bins to 4x4x4 digital count cubic bins. Although this histogram reduction introduces exiguous error, it is a practical method of dealing with the computational intensive process of generating a three-dimensional histogram with over 16 million possible bins (Russ 1992). The resulting image entropy is expressed in terms of bits per pixel and for the four experimental images used in this research was ranged from 3 to 4 bits per pixel.

The results for all of the image processing error computations are listed in Tables 4.7 and 4.8. In Table 4.7 the results are averaged for each of the color spaces and in Table 4.8 the results are averaged for each of the images. Except for the entropy column in Table 4.7, all of the values listed were calculated using the equations described previously. The entropy value listed in Table 4.7 is simply the entropy of the test image minus the entropy of the standard image. This entropy difference can then be used to gauge how much information was lost or gained during the compression process.

Based on the image processing error metrics in Table 4.7, the CIELAB and CIELUV color image compressions were again were found to be the best color spaces for JPEG compression. Although, the YIQ color space is very close behind and in some cases is even better than the CIELAB and CIELUV compressions. All of the error metrics also show the XYZ space to still be the worst color compression space. In fact, according to the entropy differences, the errors in XYZ compression were so severe that information was added to some of the images. This added information takes the form of new colors being incorporated into the image as a result of the gross quantization errors in the X and Z planes. Examining the total number of bins in the 3-dimensional histogram has shown this to be occurring with the XYZ images. Finally, the CIELAB LCh and RGB color compressions have intermediate levels of error.

Table 4.7 Image Processing Error Metrics Averaged for Each of the Color Spaces.

| Color Space | Maximum Error (0 to 255) | Minimum Error (0 to -255) | RMS Error (0 to 255) | PSNR in decibels | Entropy Difference in Bits/ Pixel |
|---|---|---|---|---|---|
| LAB | 50.75 | -75.92 | 6.75 | 31.90 | -0.12 |
| LCh | 70.25 | -86.00 | 8.01 | 30.24 | -0.12 |
| LUV | 55.17 | -76.67 | 6.75 | 31.86 | -0.09 |
| RGB | 85.83 | -92.00 | 8.66 | 29.37 | -0.03 |
| XYZ | 103.42 | -105.42 | 12.75 | 26.26 | 0.04 |
| YIQ | 59.33 | -58.33 | 7.07 | 31.39 | -0.05 |

Table 4.8 Image Processing Error Metrics Averaged for Each of the Images

| Image | Maximum Error (0 to 255) | Minimum Error (0 to -255) | RMS Error (0 to 255) | PSNR in decibels | Entropy in Bits/Pixel |
|---|---|---|---|---|---|
| Birds | 74.22 | -83.95 | 6.42 | 32.18 | 4.06 |
| Fruit | 99.00 | -107.83 | 9.14 | 29.13 | 3.63 |
| Musicians | 53.28 | -64.61 | 6.63 | 31.91 | 3.13 |
| Pasture | 56.67 | -73.61 | 11.14 | 27.46 | 3.73 |

The values listed in Table 4.8 summarize the errors for each of the experimental images. The RMS errors were largest and the PSNR lowest when the high frequency pasture and fruit images were compressed. In contrast, the max and min errors were largest when the low frequency birds and musicians images were compressed. This means that, on the average, the high frequency images were distorted the most but that the low frequency images had a larger range of possible error.

### 4.2.3. Colorimetric Measures of Error

In addition to the five image processing error metrics, four colorimetric measures of error were computed for the images created for the second experiment. The four measures that were calculated were average CIELAB $\Delta L^*RMS$, $\Delta C^*RMS$, $\Delta H^*RMS$, and

$\Delta E^*_{RMS}$. These quantities correspond to the average root mean square errors in lightness, chroma, hue, and total color between two images. Prior research has found that colorimetric error metrics correlate well with the perceived color errors between two images (Stokes 1992).

Initially the average $\Delta L^*$, $\Delta C^*$, and $\Delta H^*$ errors were calculated between the compressed and decompressed images and the original image. However, because both positive and negative errors were occurring, the magnitude of these average errors was not representative of the extent of the colorimetric error occurring during compression. Consequently, the average RMS errors were computed for $\Delta L^*$, $\Delta C^*$, and $\Delta H^*$. The average CIELAB $\Delta L^*_{RMS}$, $\Delta C^*_{RMS}$, and $\Delta E^*_{RMS}$ errors can be calculated according to the following formulae:

$$\overline{\Delta L^*_{RMS}} = \sqrt{\sum_{p=0}^{T} (L^*test - L^*std)^2} \tag{7.6}$$

$$\overline{\Delta C^*_{RMS}} = \sqrt{\sum_{p=0}^{T} (C^*test - C^*std)^2} \tag{7.7}$$

$$\overline{\Delta E^*} = \sum_{p=0}^{T} \sqrt{(L^*test - L^*std)^2 - (a^*test - a^*std)^2 - (b^*test - b^*std)^2} \tag{7.8}$$

where p is the index used to access individual pixels, T is the total number of pixels in the image, test refers of the compressed and decompressed image, and std applies to the standard or original image. The CIELAB and CIELAB LCh coordinates for the pixels can be computed using Eqns. 2.10 to 2.21. The hue-difference or $\Delta H^*$ is not calculated using the hue angle, $h_{ab}$. Instead, a vector is computed which is the difference between two points in CIELAB LCh space which is caused only by the change in hue between the two points. The equation used to determine average CIELAB $\Delta H^*_{RMS}$ can be written as follows:

$$\overline{\Delta H^*_{RMS}} = \sqrt{\sum_{p=0}^{T} \left( \left( \sqrt{(\Delta E^*)^2 - (L^*test - L^*std)^2 - (C^*test - C^*std)^2} \right)^2 \right)} \qquad (7.9)$$

where p is the pixel index and T is the total number of pixels and the DE* is the same quantity which is listed inside the summation S in Eqn. 7.8.

The results of the colorimetric error computations are shown in Table 4.9 and 4.10. In Table 4.9 the results are averaged for each of the color spaces. CIELAB and CIELUV had the smallest colorimetric errors of all of the color spaces. The CIELAB LCh, YIQ, and RGB color spaces all had intermediate levels of colorimetric error. Once more, the XYZ space was last with largest errors for each of the colorimetric error metrics. The RGB, XYZ, and YIQ color compressions all had average $DE^*_{ab}$ larger than three. Therefore it is probable there was a perceptible color difference between the original image

Table 4.9 Colorimetric Error Metrics Averaged for Each of the Color Spaces.

| Color Space | Average $\Delta L^*RMS$ | Average $\Delta C^*RMS$ | Average $\Delta H^*RMS$ | Average $\Delta E^*ab$ |
|---|---|---|---|---|
| CIELAB | 1.86 | 6.29 | 6.16 | 2.80 |
| CIELCh | 2.20 | 7.41 | 7.27 | 2.93 |
| CIELUV | 1.92 | 6.50 | 6.34 | 3.01 |
| RGB | 2.78 | 7.96 | 7.61 | 4.48 |
| XYZ | 3.18 | 10.61 | 10.39 | 7.50 |
| YIQ | 2.20 | 7.04 | 6.91 | 4.30 |

and the image compressed and decompressed in these spaces. For all of the color spaces, the chroma and hue errors were two or three times as large as the lightness errors. This difference is likely due to the fact that the JPEG algorithm quantizes the chrominance information much more coarsely than the luminance information. This table demonstrates that as the image quality deteriorates as a result of compression, the color information is the first to be sacrificed.

Table 4.10 lists the colorimetric errors averaged for each of the images. As was the case with the image processing errors, the high frequency images tended to have larger errors. The low frequency birds image had very low colorimetric errors. At this point in the discussion, it is encouraging to note that, in general, the results from the psychophysical experiments, the image processing error measures, and the colorimetric

Table 4.10 Colorimetric Error Metrics Averaged for Each of the Images

| Image | Average $\Delta L^*RMS$ | Average $\Delta C^*RMS$ | Average $\Delta H^*RMS$ | Average $\Delta E^*$ |
|---|---|---|---|---|
| Birds | 1.48 | 6.29 | 6.07 | 2.89 |
| Fruit | 2.79 | 8.62 | 8.41 | 4.78 |
| Musicians | 1.87 | 6.22 | 6.20 | 4.29 |
| Pasture | 3.28 | 9.41 | 9.11 | 4.75 |

error measures are all in agreement. The CIELAB and CIELUV color spaces are the best color spaces for JPEG image compression. The RGB and XYZ color spaces are the worst and the YIQ and CIELCh are in between.

### 4.2.4. Error Images

One last technique which is often used to demonstrate the errors which occur during image compression are error images. These images attempt to provide a qualitative impression of the errors present in compressed image by creating an image which represents only the error in the final image (Rabbani and Jones 1991). The pixel values for the error image are computed using the following algorithm:

$$\text{Error Image Pixel} = \left(\left(\frac{Rt+Gt+Bt}{3}\right)-\left(\frac{Rs+Gs+Bs}{3}\right)\right)\cdot\left(\frac{\text{Max Error}}{128}\right)+128 \qquad (7.10)$$

where Rt, Gt, and Bt are the red, green, and blue digital counts for given pixel in the test image, Rs, Gs, and Bs are the digital counts for the corresponding pixel in the standard

image, and Maximum Error is some scaling factor usually based on the largest pixel error the given image or sequence of images. The value computed for the error image pixel using Eqn. 7.10 is assigned to the red, green, and blue channels of the pixel in the error image. Consequently, the error image is a monochrome image which highlights the error in test image. Images 4.1 to 4.6 are examples of error images for the six color space compressions of the fruit image used for the second experiment. Notice that the 128 added at the end of Eqn. 7.10 scales the error image so that an error of zero comes out as a medium gray. The value of max error used for these images was the largest error which occurred in all six of these images, in this case the minimum error of 144 for the XYZ compression.

These images provide an interesting visual representation of the errors generated during the compression process. The CIELAB and CIELUV error images are almost entirely gray and only a few details are visible. The CIELAB LCh and YIQ error images exhibit larger errors but the image is still mainly a uniform gray. In comparison, the RGB, and especially the XYZ, error images show a considerable amount of error. In fact, the errors are large enough to provide a outline of almost all of the significant features in the image. These very dark and very light error regions are generally along the edges and in other high frequency areas of the image. However, these images do not accurately denote the errors that were visible in the low frequency regions of the image.

## 4.3. Analysis

The results of the two experiments provide considerable corroboration that the best color compression spaces are the nonlinear transforms of the device color space and that the worst color compression spaces are the RGB and XYZ spaces. However, some additional

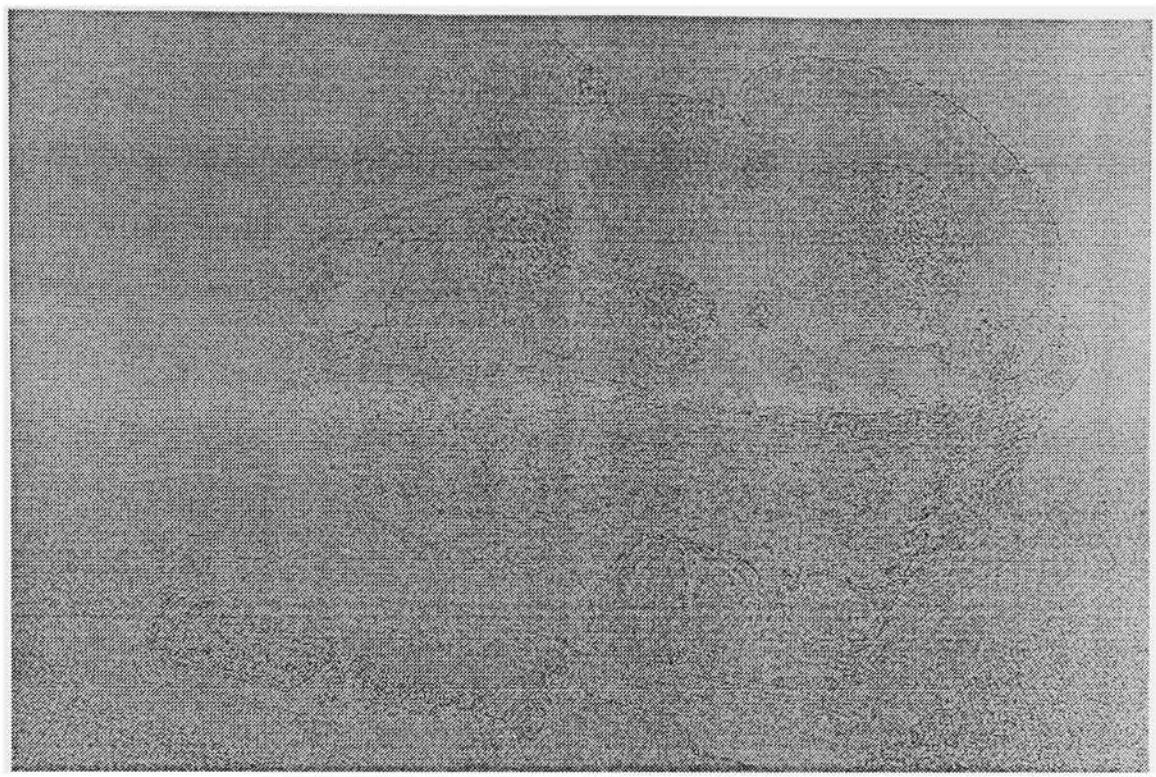


Image 4.1 CIELAB Fruit Error Image.

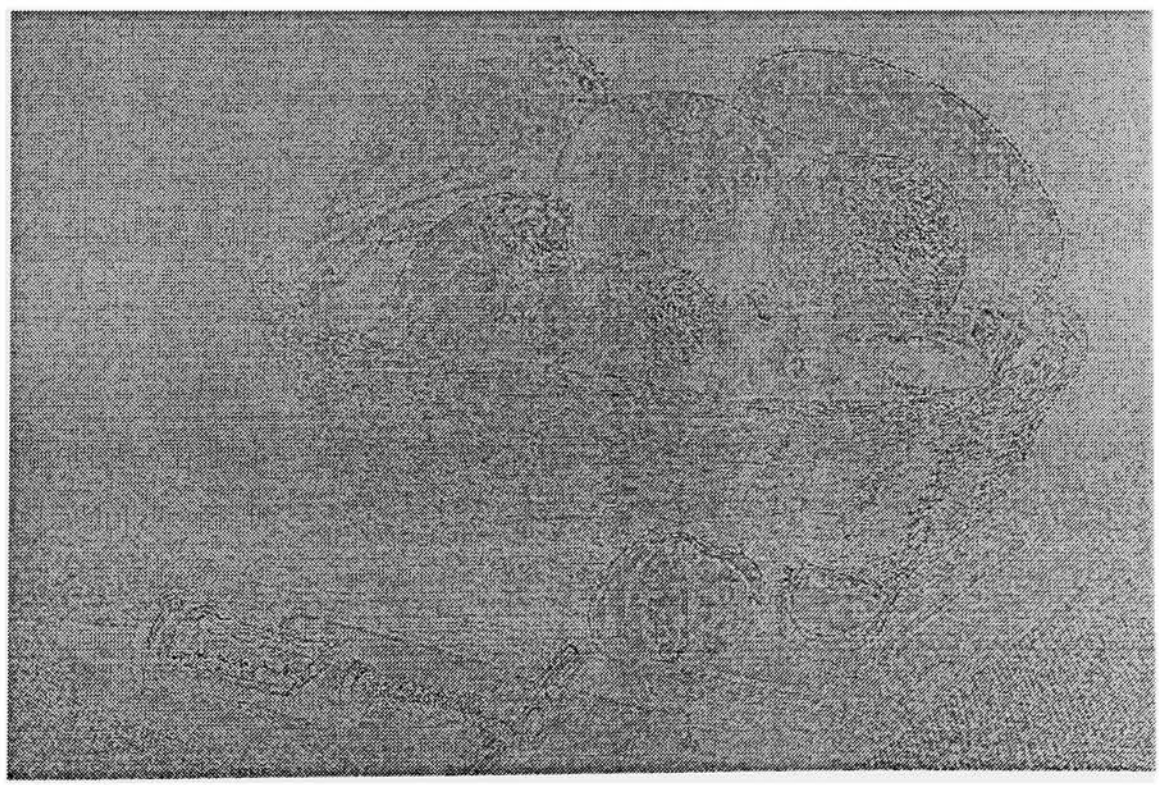


Image 4.2 CIELAB LCh Fruit Error Image.

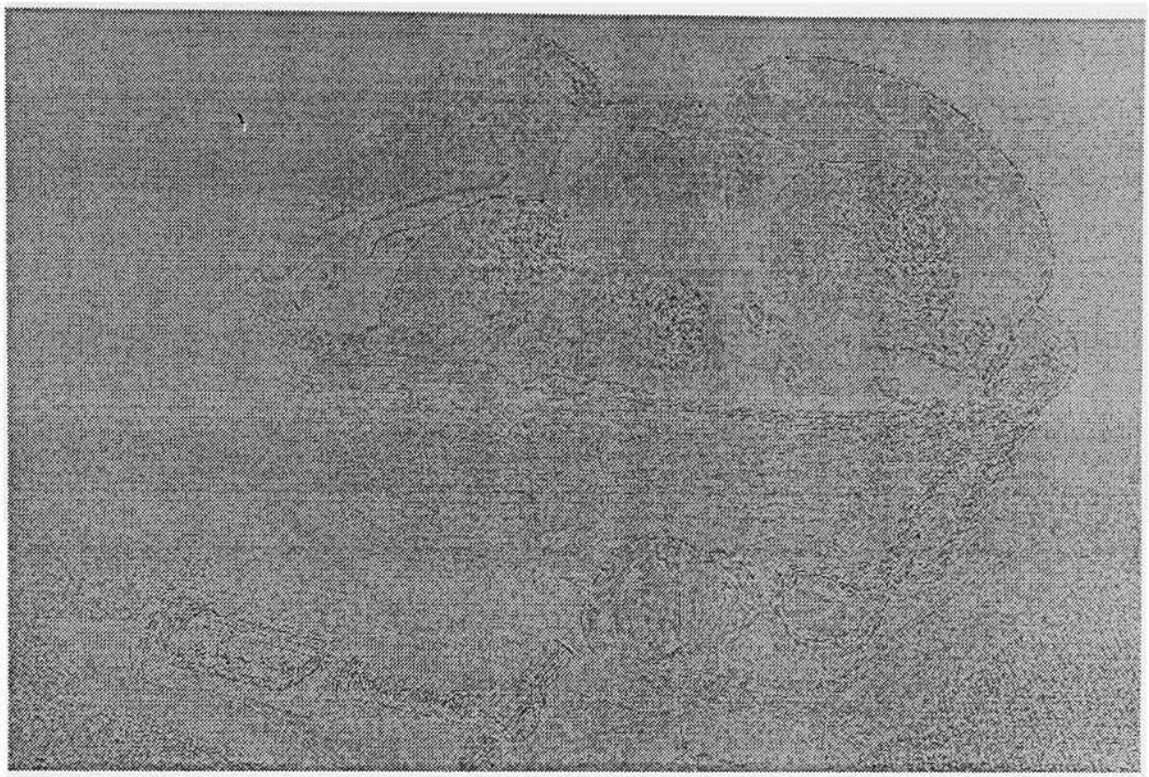


Image 4.3 CIELUV Fruit Error Image.

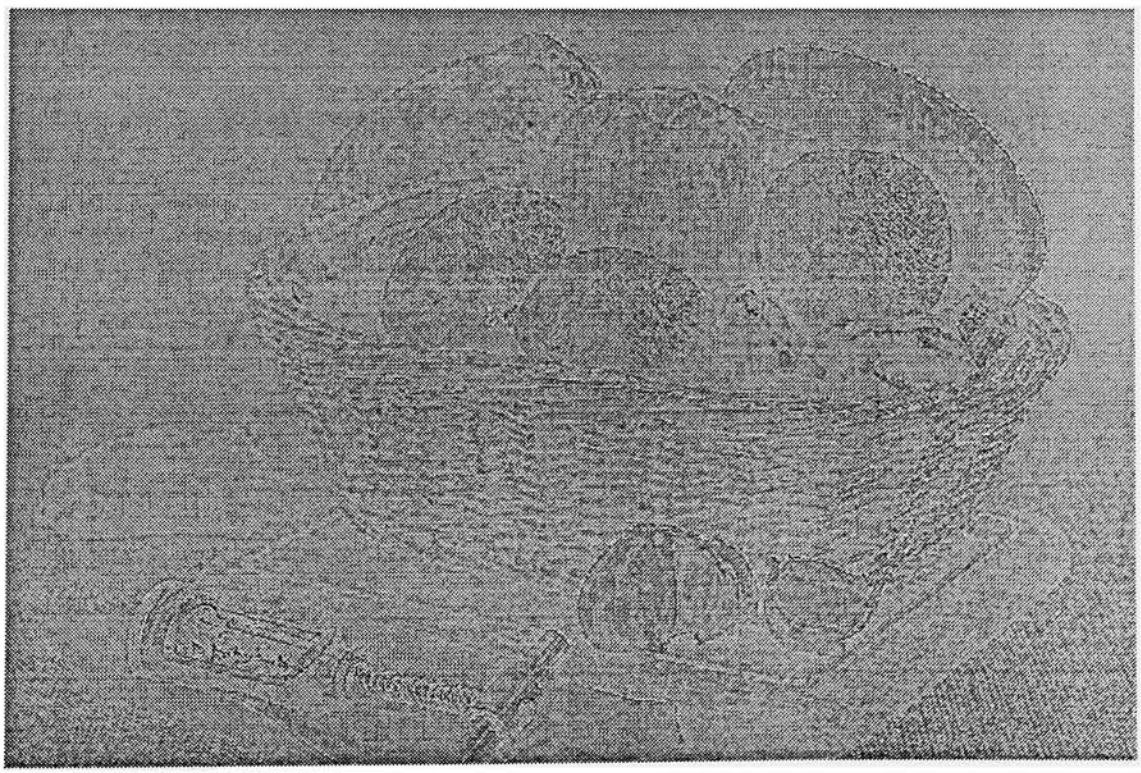


Image 4.4 RGB Fruit Error Image.

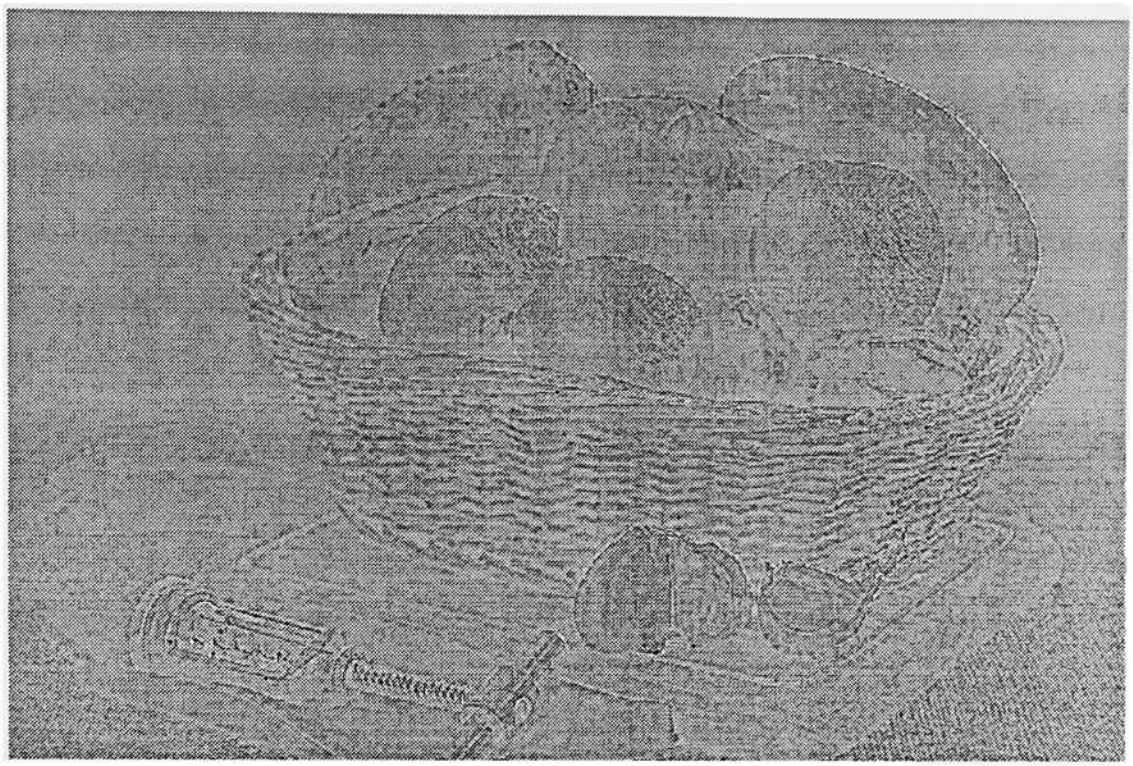


Image 4.5 XYZ Fruit Error Image

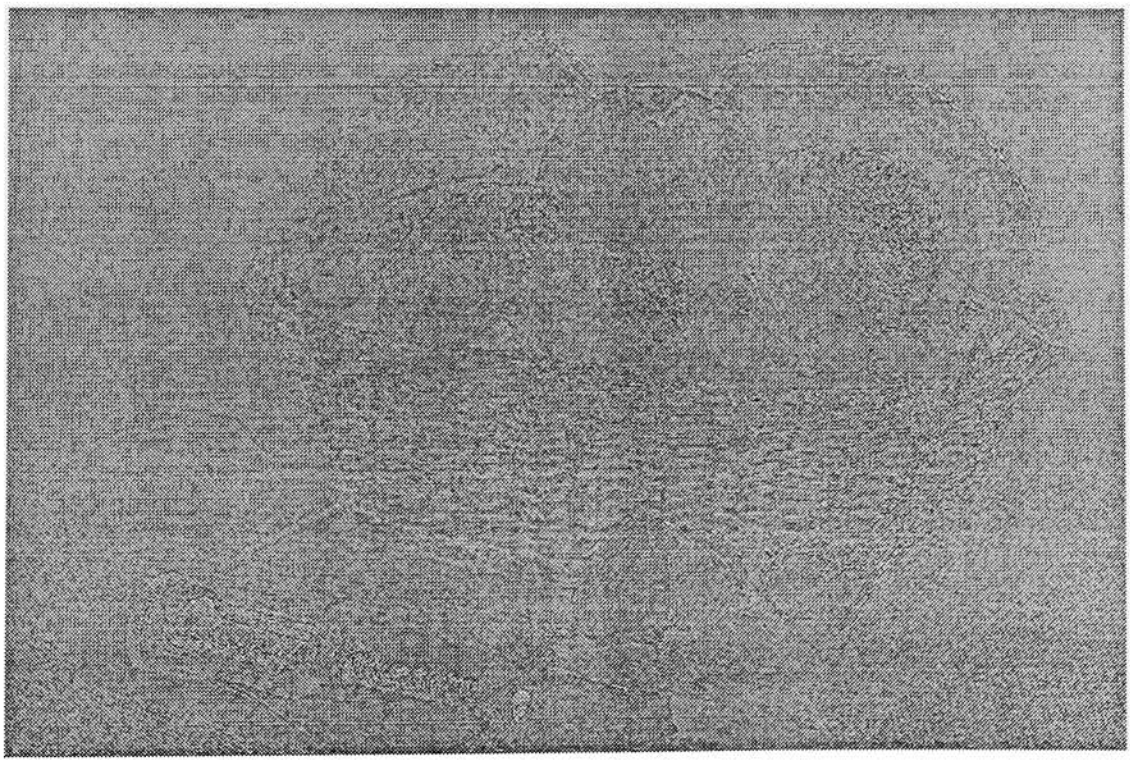


Image 4.5 YIQ Fruit Error Image

discussion should be focused on the specifics of the results. Three specific topics that will be examined in this section are channel redundancy, the perceptual uniformity of L* and the problem of quantizing polar hue information. Lastly, there will be brief discussion of the percent improvement in compression that could be attained if a color space of other than the RGB color space was used for compression.

### 4.3.1. Channel Redundancy

The issue of channel redundancy was noted briefly in the Background section on color spaces. However, this aspect of color spaces is one of the significant reasons that the RGB and XYZ color spaces performed so poorly. In order to provide some measure of the redundancy of information in the three channels of each of the color spaces, the standard deviations for each of the channels were computed separately for the fruit image. Before this calculation was performed, the pixel values for all of the color space images were normalized between 0 and 255. The standard deviation was then calculated using the usual equation:

$$\text{Standard Deviation} = \sqrt{\left(\left(\overline{X^2}\right) - \left(\overline{X}\right)^2\right)} \tag{7.11}$$

where $\overline{X}$ is the mean pixel value for a given channel and $\overline{X^2}$ is the mean squared pixel value for that channel. This calculation was repeated for each of the three channels of the fruit image. The value of the standard deviations for each of the three channels was then summed and each of the individual standard deviations was divided by this total. This converted the standard deviation values to normalized percentages. The resulting list of percentages can be found in Table 4.11.

Table 4.11 Normalized Standard Deviations for Each of the Individual Channels for the Different Color Space Representations of the Fruit Image.

| Color Space | Channel One | Channel Two | Channel Three |
|---|---|---|---|
| LAB | 61.86% | 17.69% | 20.45% |
| LCh | 40.31% | 18.55% | 41.14% |
| LUV | 67.94% | 17.69% | 14.37% |
| RGB | 28.50% | 39.47% | 32.03% |
| XYZ | 35.38% | 35.33% | 29.29% |
| YIQ | 61.28% | 28.08% | 10.64% |

As can be seen in Table 4.11, the deviations in the three channels for the RGB and XYZ color space representations of the fruit image are all about approximately equal. This suggests that there is a considerable redundancy in the RGB and XYZ color space representations. In comparison, the CIELAB, CIELUV and YIQ color spaces have a majority of their deviation concentrated in one channel. The L* and Y channels contain most of the information for these images. The CIELAB LCh values are unusual in that the $h_{ab}$ channel variation is as large as the L* variation. In this case, this large value for the $h_{ab}$ channel is occurring as a result of the polar nature of $h_{ab}$. This will be further demonstrated in the following section on quantizing $h_{ab}$.

Future research could also compare these color spaces to the Karhunen-Louve color conversion. The Karhunen-Louve conversion is an orthogonal transformation that results in statistically uncorrelated channels and an optimal distribution of energy in each of the three channels (Pratt 1971). This space would provide a base level of channel independence for comparison purposes.

### 4.3.2. Perceptual Uniformity of L*

One of the major differences between the nonlinear color spaces and the other color spaces is the way in which the lightness information was transformed. This is especially true in the expansion of the darker regions or shadows of the images. The equations defining L*, Eqns. 2.10 and 2.11, were derived so that it would be more perceptually uniform than Y. The extent of this transformation in terms of CRT luminances may to be

obvious from those two equations. Therefore, a Jones diagram was generated in Fig. 4.9. This figure shows the conversion from trios of equivalent linear red, green, and blue digital counts to Y in the lower right quadrant. The upper right quadrant shows the conversion from Y to L* and the the upper left quadrant shows the conversion from L* to normalized L*. The lower left quadrant shows the conversion from matched red, green, and blue digital counts directly to normalized L*. Notice that a digital count trio of 51 transforms to a normalized L* over 128. This means that the bottom 20% of the digital count triplets expands to occupy over 50% of the normalized L* scale. This is shown in Fig. 4.9 as an arrow starting at 51 on the RGB axis and circling the origin in a counter-clockwise manner until it comes back to 51 on the RGB axis. This additional precision means that the shadows will be more accurately quantized and the lightness information will be compressed in a more perceptually uniform manner.

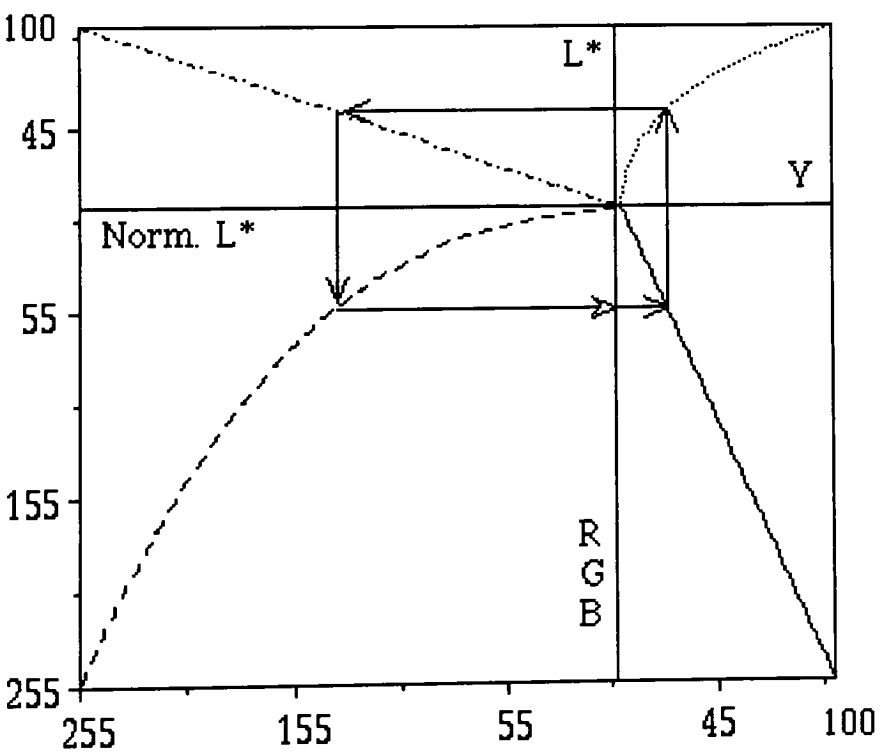



Fig. 4.9 L* Transform Jones Diagram.

### 4.3.3 Quantization of Polar $h_{ab}$ Information

The main reason that CIELAB LCh space did not do as well as the CIELAB and CIELUV color spaces was because of $h_{ab}$. This is because the polar nature of $h_{ab}$ generated high frequencies in the data even when there was none present in the original image. One case that is easy to imagine is for reddish pixels that happened to fall along the polar discontinuity between the positive a*-negative b* quadrant and the positive a*-positive b* quadrant of CIELAB space. Small variations could would cause the $h_{ab}$ to alternate between 0° and 360°. This would transform a low frequency shifts in CIELAB coordinates into a high frequency shifts in $h_{ab}$. Consequently, it would take more bits to represent the high frequency $h_{ab}$ data than it would to represent the lower frequency CIELAB data.

However, it is likely that blocks containing pixels on both sides of the polar $h_{ab}$ discontinuity at 360° are fairly uncommon. Another, more probable case would be the generation of high frequency $h_{ab}$ data as a result of minor variations in a region of near achromatics. An example of this is illustrated using a flowchart of the $h_{ab}$ compression and decompression in Fig. 4.10. This flowchart is similar to the one presented in Fig. 2.13 and, in fact, the original source RGB pixel block is the same. The $h_{ab}$ block of 8x8 pixels at the top of Fig. 4.10 is just the hue angle plane of the CIELAB LCh representation of the RGB 8x8 blocks at the top of Fig. 2.12. This allows the results of the two compressions to be compared. It is evident the $h_{ab}$ representation of the original data definitely exhibits higher frequencies than was present in original red, green, or blue pixel planes. This is also true when the $h_{ab}$ plane is compared to the XYZ and CIELAB representations of the original RGB data in Fig. 2.12.

This shift to higher frequencies is because the CIELAB coordinates of original near achromatic triangles were scattered around the CIELAB origin. This made the $h_{ab}$ vary widely in order to represent these near achromatics. Consequently, the final bit string for

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 85.6 | 85.6 | 85.6 | 85.6 | 85.6 | 85.6 | 85.6 | -33.1 |
| 85.6 | 85.6 | 85.6 | 85.6 | 85.6 | 85.6 | -33.1 | -33.1 |
| 85.6 | 85.6 | 85.6 | 85.6 | 85.6 | -33.1 | -33.1 | -33.1 |
| 85.6 | 85.6 | 85.6 | 85.6 | -33.1 | -33.1 | -33.1 | -33.1 |
| 85.6 | 85.6 | 85.6 | -115.1 | -115.1 | -33.1 | -33.1 | -33.1 |
| 85.6 | 85.6 | -115.1 | -115.1 | -115.1 | -115.1 | -33.1 | -33.1 |
| 85.6 | -115.1 | -115.1 | -115.1 | -115.1 | -115.1 | -115.1 | -33.1 |
| -115.1 | -115.1 | -115.1 | -115.1 | -115.1 | -115.1 | -115.1 | -115.1 |

**Normalized $h_{ab}$ with 128 Subtracted.**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 250.8 | 215.2 | 189.1 | 177.5 | 208.7 | 240.7 | 212.6 | 82.6 |
| 191.0 | 231.3 | 208.5 | 213.7 | 222.6 | 212.5 | 92.9 | 106.9 |
| 210.3 | 165.3 | 231.6 | 255.6 | 232.8 | 69.0 | 73.1 | 104.1 |
| 224.1 | 219.7 | 177.9 | 233.5 | 88.6 | 64.9 | 87.6 | 99.2 |
| 240.5 | 229.9 | 221.1 | 0.0 | 28.3 | 104.2 | 91.9 | 108.7 |
| 220.4 | 230.1 | 5.6 | 16.5 | 0.0 | 29.9 | 121.3 | 94.5 |
| 191.8 | 0.0 | 38.8 | 6.2 | 9.3 | 26.1 | 23.3 | 78.4 |
| 0.0 | 49.4 | 0.0 | 5.6 | 29.9 | 27.0 | 17.5 | 0.0 |

**Reconstructed Block of 8x8 Normalized $h_{ab}$ pixels.**

FDCT

IDCT + 128

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -54.5 | 270.0 | 91.4 | 28.4 | 0.1 | 8.6 | 6.6 | 2.2 |
| 494.4 | 59.3 | -192.8 | 0.0 | -40.7 | 0.0 | -10.6 | 0.0 |
| -91.3 | -123.9 | -100.3 | 70.2 | 37.7 | 13.8 | -0.2 | 3.2 |
| 34.9 | 0.1 | 151.3 | 59.3 | -56.2 | -0.1 | -18.8 | -0.1 |
| -0.1 | -22.4 | -37.9 | -45.6 | -100.3 | 30.6 | 15.9 | 4.5 |
| 22.8 | -0.2 | 9.1 | 0.0 | 83.2 | 59.6 | -12.6 | 0.2 |
| -6.4 | -6.8 | 0.0 | -8.7 | -15.8 | -19.5 | -100.5 | 10.7 |
| -3.6 | 0.2 | 17.0 | -0.0 | -8.1 | -0.2 | 52.6 | 59.2 |

**Forward Discrete Cosine Transform Coefficients**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -51 | 270 | 96 | 47 | 0 | 0 | 0 | 0 |
| 486 | 63 | -182 | 0 | 0 | 0 | 0 | 0 |
| -96 | -130 | -112 | 99 | 0 | 0 | 0 | 0 |
| 47 | 0 | 198 | 99 | -99 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | -99 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 99 | 99 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | -99 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 99 | 99 |

**Decoded and Dequantized Values**

Quantize

Dequantize

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -3 | 15 | 4 | 1 | 0 | 0 | 0 | 0 |
| 27 | 3 | -7 | 0 | 0 | 0 | 0 | 0 |
| -4 | -5 | -2 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 2 | 1 | -1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**FDCT Coefficients Quantized Using Chrominance Q-Table**

```
1001110000001110010010 1
1010011100011010100011 1
0100001010010011111101 1
1011110101100101111011 1
1110010111101001111001 1
1110101111101001101010 11
```

**Huffman Encoded Bit Stream**

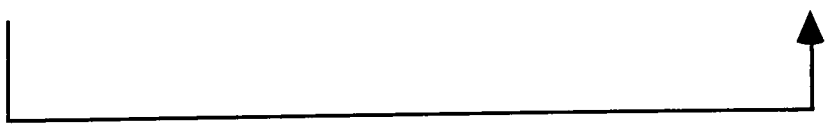


Fig. 4.10. Computational example for the compression and decompression of a normalized 8x8 block of $h_{ab}$ pixels based on the same RGB blocks shown at the top of Fig. 2.12. The compression steps are shown going down the page and decompression steps are shown going up the page.

the $h_{ab}$ compression shown in Fig. 4.10 is over 120 bits long. In comparison, the b* compression was only 12 bits long. This is a clear demonstration of the inefficiency of using the $h_{ab}$ representation for JPEG compression.

### 4.3.4. Improvement Over RGB Compression

The results of the first experiment showed that not only is JPEG an image dependent algorithm, but is also a color space dependent algorithm. The possibility of improving the JPEG compression based on color space selection was investigated using the visually lossless thresholds derived during the first experiments. In a majority of cases, the RGB compression had the highest threshold. Therefore, all of the comparisons in improvement in compression will be made relative to the RGB color compression.

The first step in comparing the different color spaces is to determine how compressed the threshold image for that color space was relative to the original image. Specifically, the compressed threshold image can be expressed as a fraction or percentage of the size of the original image. The exact formula to compute this percent reduction in the original image data can be written:

$$P = \left( \frac{\text{Bits in Compressed Image}}{\text{Bits in Original Image}} \right) \cdot 100 \qquad (7.12)$$

where P is the percent of the original size that the threshold image has been compressed to. This calculation was performed for all of the images and for all of the color spaces. Next, the percentage for the RGB compression was subtracted from all of the percentages for all of the other color spaces. This yielded a difference in percent reduction over the original image relative to the RGB color space compression. Table 4.12 lists all of these differences for each of the images and also averaged for all of the color spaces.

Table 4.12 Percent Improvement over RGB Compression.

| Color Space | Percent Difference for Birds | Percent Difference for Fruit | Percent Difference for Musicians | Percent Difference for Pasture | Overall Percent Difference |
|---|---|---|---|---|---|
| LAB | 8.1% | 14.2% | 8.8% | 13.2% | 11.1% |
| LCh | 7.5% | 14.5% | 3.8% | -2.9% | 5.7% |
| LUV | 9.8% | 13.1% | 9.4% | 8.4% | 10.2% |
| XYZ | 7.2% | 11.0% | 3.4% | 7.1% | 7.2% |
| YIQ | 7.7% | 12.7% | 6.8% | 9.6% | 9.2% |
| Average for Each Column | 8.1% | 13.1% | 6.4% | 7.1% | 8.7% |

Based on the values for the overall percentage differences in Table 4.12, all of the color spaces did at least 5% better than the RGB color space for JPEG compression. The CIELAB and CIELUV color compressions were over 10% better than the RGB compressions and the YIQ compression was in a close third with a 9% improvement over RGB. The XYZ compression was in forth and the CIELAB LCh space was in fifth with 5% improvement. The last row in Table 4.12 lists the improvement over RGB averaged for each of the columns in the table. It is interesting to note that, on the average, using a color space other than RGB provided the most improvement in compression for the fruit image and the least improvement for the musicians image. Of course these conclusions are dependent on the basis of the comparisons and the conditions of the compression. In this case, the images were compared according to the level of visually lossless compression for Baseline JPEG algorithm.

# 5. Conclusions

The results of both of the pyschophysical experiments and the numeric measures of error indicate that nonlinear color spaces are the preferred color spaces for JPEG image compression. Color spaces such as CIELAB and CIELUV performed the best of the six color spaces tested. There was no significant difference between the CIELAB and CIELUV color spaces. The device color space, RGB, was the worst for JPEG image compression. Linear transforms of the RGB space produced intermediate results. The YIQ and XYZ color spaces were better than RGB and worse than CIELAB and CIELUV for visually losseless compression. However, at higher levels of compression, the XYZ color space was worse than the RGB compression. Finally, CIELAB LCh was very inconsistent and was both a good and a poor color compression space. The $h_{ab}$ information was very difficult to compress efficiently and, as a result, the CIELAB LCh compression ranged varied quite a bit.

This thesis research has produced some interesting results, but there are still a multitude of issues and problems to be addressed. First of all, the JPEG algorithm is designed for any three component color space. However, as this research has shown, some color spaces are better for compression than others. Regardless, there are other ways in which the JPEG algorithm could possibly be enhanced to be more conducive for compression in a given color space. Altering the Q-Tables or modifying the Huffman Tables are examples of two possible changes which could be made to the JPEG algorithm in order to customize it for a specific color space.

The results of this research suggest that tristimulus space is not very efficient for use as a color compression space. It may be possible to improve the tristimulus compression by altering the quantization scheme. For instance, the Luminance Q-Table could be used for all three planes or the chrominance plane could be applied to just the Z plane. Another option would be to convert the tristimulus data from XYZ to Yxy and the

quantize the Yxy data according to a Luminance-Chrominance-Chrominance scheme. Although, the results do not indicate how much of an improvement these altemate schemes would have yielded. It seems likely though that these schemes could not be any worse than the XYZ scheme used for this thesis and any better than the compression achieved using the YIQ color space.

The compression efficiency of the CIELAB LCh space could also be improved by determining a method of quantizing the hue angle plane. This could possibly be done by changing the representation of hue angle or altering the way in which it is quantized. This additional computational effort could improve the CIELAB LCh compression to the level of the other nonlinear color spaces. One possible method of doing this would be to encode the $h_{ab}$ of the first pixel in a given block and then differentially encode all the remaining hue information as $\Delta H^*$ or the hue difference from the preceding hue value.

Similarly, based on observer feedback and comments, it would appear that the loss of chroma in the image is definitely a noticeable error. This dramatic loss of chroma is also supported by the magnitude of chroma errors in the Table 4.9 listing the colorimetric errors for the second set of experimental images. A possible improvement in the compression could be gained by performing some sort of transformation on the color channels before compressing them. For instance, the a* and b* planes could be run through a quadratic or cubic look up table before compressing them. This would assign a higher priority to maintaining the higher chroma colors. This strategy could also be used with any of the other color spaces although further experiments would be necessary to determine if this transformation would truly improve the compression.

Of course there is the topic of expanding the scope of the investigation to include a wider range of color spaces. For example, the OSA, HSL, YCbCr, YES and many other color spaces are also available for testing. It seems likely that these other color spaces would produce results similar to those of the device, linear, and non-linear color spaces used for this research. More research is necessary to support this hypothesis.

Lastly, the study could be expanded to cover a wider visual range of image compression. The results of the XYZ compressions suggest that there may be visual differences in the compression depending on the extent to which the image is compressed. This would then raise the question of which color space is optimal for image compression for all levels of image compression.

## References


N. Ahmed, T. Natarajan, K. R. Rao, "Discrete Cosine Transform", *IEEE Transcations on Computers*, **23** 1316-1322 (1974).

B. A. Barry, *Errors in Practical Measurement in Science, Engineering and Technology*, John Wiley and Sons, New York, 30-33 (1978).

C. J. Bartleson, F. Grum, *Optical Radiation Measurements*, Vol. 5. Academic Press, New York, 368-439 (1984).

G. Buchsbaum, "Color Signal Coding: Color Vision and Color Television", *Col. Res. App.*, **12**, 266-269 (1987).

CIE, "Colorimetry", *Publication 15.2*, Comission International d'Eclairage, Austria, Vienna, (1984).

M. Fairchild, *Psychophysics for Imaging*, Short Course Notes, Rochester Institute of Technology, (1992).

R. S. Gentile, J. P. Allebach, E. Walowit, "Visually Lossless Compression of Color Images", SPIE **1258** *Image Communications and Workstations*, 190-201 (1990).

R. S. Gentile, J. F. Allenbach, E. Walowit, "Quantization of Color Images based on Uniform Color Spaces", *Journ. Img. Tech.* **16** 11-21 (1990).

R. C. Gonzalez, R. E. Woods, *Digital Image Processing*, Addison-Wesley Pub. Co., New York, 1990, pp. 307-405.

D. Goodenow, "JPEG Compression", *PRINT RIT The Journal of the RIT School of Printing Management and Science*, **1** 6-9 (1993).

D. J. Granrath, "The Role of Human Visual Models in Image Processing", *Proc. of the IEEE.*, **69** 552-561 (1981).

N. C. Griswold, "Perceptual Coding in the Cosine Transform Domain", *Optical Engineering*, **19** 306-311 (1980).

B. G. Haskell, A. N. Netravali, *Digital Pictures: Representation and Compression*, Plenum Press, New York, 1988, pp 147-441.

R. W. G. Hunt, *The Reproduction of Color*, Fountain Press, Tolworth, England, 373-399, (1988).

J. M. Kasson, W. Plouffe, "Requirements for Computer Interchange Color Spaces", SPIE **1258** *Image Communications and Workstations*, 172-183 (1990).

S. A. Klein, T. Carney, "'Perfect' displays and 'perfect' image compression in space and time", SPIE **1453**, *Human Vision, Visual Processing, and Digital Display II*, 190-204 (1991).

S. A. Klein, A. D. Silverstein, T. Carney, "Relavence of Human Vision to JPEG-DCT Compression", SPIE **1666**, *Human Vision, Visual Processing, and Digital Display III*, 200-215 (1992).

A. Leger, T. Omachi, G. K. Wallace, "JPEG Still Picture Compression Algorithm", *Optical Engineering*, **30** 947-954 (1991).

J. O. Limb, "Visual Perception Applied to the Encoding of Pictures", SPIE **87**, *Advances in Image Transmission Techniques* 80-87 (1976).

J. O. Limb, C. B. Rubinstein, J. E. Thompson, "Digital Coding of Color Video Signals-A Review", *IEEE Trans. on Com.*, **COM-25** 1349-1384 (1977).

H. Lohscheller, "A Subjecticely Adapted Image Communication System," *IEEE Trans. on Com.*, **COM-32** 1316-1322 (1984).

J. L. Mannos and D. J. Sakrison, "The Effects of a Visual Fidelity Criterion on the Encoding of Images", *IEEE Trans. on Info. Theory*, **IT-20** 525-536 (1974).

J. Mauro, "Digital Image Compression Theory and Techniques, Part I: Information Theory", *Electronic Imaging* Feb, 1985 60-66.

J. Mauro, "Digital Image Compression Theory and Techniques, Part II: Image Theory", *Electronic Imaging* March, 1985 61-63.

P. W. Melnychuck, M. J. Barry, M. S. Mathieu, "Effect of Noise and MTF on the Compressibility of High Resolution Color Images", SPIE **1244** *Image Processing Algorithms and Techniques* 255-262 (1990).

J. L. Mitchell, W. B. Pennebaker, "Evolving JPEG Color Data Compression Standards", *Standards for Electronic Imaging Systems*, SPIE Optical Engineering Press, Bellingham, Washington, 1991, pp 68-97.

J. L. Mitchell, W. B. Pennebaker, *JPEG Still Image Compression Standard*, Van Nostrand Reinhold, New York, (1993).

R. J. Motta, "An Analytical Model for the Colorimetric Characterization of Color CRTs", MS Thesis, Rochester Institute of Technology, Center for Imaging Science, (1991).

H. A. Peterson, H. Peng, J. H. Morgan, W. B. Pennebaker, "Quantization of Color Images in the DCT Domain", SPIE **1453** *Human Vision, Visual Processing, and Digital Display II*, 210-222 (1991).

W. K. Pratt, " Spatial Transform Coding", *IEEE Trans. on Com. Tech.* **COM-19**, 980-992 (1971).

M. Rabbani, *Digital Image Compression*, Short Course Notes, Rochester Institute of Technology, (1990).

M. Rabbani, P. W. Jones, *Digital Image Compression Techniques*, SPIE Optical Engineering Press, Bellingham, Washington, 1991.

G. Raisbeck, *Information Theory: An Introduction for Scientists and Engineers*, MIT Press, Cambridge, Massachusetts, 1965.

J. C. Russ, *The Image Processing Handbook*, CRC Press, Boca Raton, 232-233 (1992).

SAS/STAT Users's Guide, Version 6, Forth Ed., 1071-1126 (1990).

W. F. Schreiber, *Fundamentals of Electronic Imaging Systems*, Springer-Verlag, New York, 1991, pp 104-133, 198-207.

C. S. Stein, A. B. Watson, L. E. Hitchner, "Psychophysical Rating of Image Compression Techniques", SPIE **1077**, *Human Vision, Visual Processing, and Digital Display*, 198-207 (1989).

M. D. Stokes, "Colorimetric Tolerances of Digital Images", MS Thesis, Rochester Institute of Technology, Center for Imaging Science (1991).

G. K. Wallace, "The JPEG Picture Compression Standard", *IEEE Transactions on Consumer Electronics*, **38** xviii-xxxiv (1992).

G. K. Wallace "Overview of the JPEG (ISO/CCITT) Still Image Compression Standard" SPIE **1244**, *Image Processing Algoritms and Techniques*, 220-233 (1990).

G. Wallace, R. Vivian, H. Poulsen, "Subjective Testing Results for Still Picture Compression Algorithms for International Standardization",*GLOBECOM 88* 1022-1027 (1988).

G. Wyszecki, W.S. Stiles, *Color Science: Concepts and Methods, Qualitative Data and Formulae*, John Wiley and Sons, 1982.

I. Ylakoski, H. Ronngren, "CIE XYZ Colour Space used for Image Compression", 1992 Targa Proceedings, 139-151.

## Appendix A. Huffman Encoding Tables for AC Terms

Tables H.1 and H.2 list the code lengths and code words that would be assigned to various zero run lengths and coefficient sizes for AC terms. These tables were derived using the average statistics of large collection of experimental images. There are two tables for either luminance or chrominance AC coefficients respectively. The actual text for the tables was copied directly out of the 1990 ISO/CCITT DRAFT. Notice that the code length is only the length, in bits, of the most significant bits or code word and that the complete code word is not derived until the least significant bits are computed. The total code length for both the most significant and least significant bits can be computed by taking the MSB code length and adding the size of the coefficient to that value. For instance, using Table H.1 the total code word length for a run length 1, size 7 term would be 16 plus 7 or 23 bits altogether. These two tables were then used to create look-up tables in the JPEG C program that simply converted from run length and size straight to the total number of bits.

Table A.1. Huffman Encoding Table for Luminance AC Terms

| Run/Size | Code length | Code word |
|---|---|---|
| 0/0 (EOB) | 4 | 1010 |
| 0/1 | 2 | 00 |
| 0/2 | 2 | 01 |
| 0/3 | 3 | 100 |
| 0/4 | 4 | 1011 |
| 0/5 | 5 | 11010 |
| 0/6 | 7 | 1111000 |
| 0/7 | 8 | 11111000 |
| 0/8 | 10 | 1111110110 |
| 0/9 | 16 | 1111111110000010 |
| 0/A | 16 | 1111111110000011 |
| 1/1 | 4 | 1100 |
| 1/2 | 5 | 11011 |
| 1/3 | 7 | 1111001 |
| 1/4 | 9 | 111110110 |
| 1/5 | 11 | 11111110110 |
| 1/6 | 16 | 1111111110000100 |
| 1/7 | 16 | 1111111110000101 |
| 1/8 | 16 | 1111111110000110 |

| | | |
|---|---|---|
| 1/9 | 16 | 1111111110000111 |
| 1/A | 16 | 1111111110001000 |
| 2/1 | 5 | 11100 |
| 2/2 | 8 | 11111001 |
| 2/3 | 10 | 1111110111 |
| 2/4 | 12 | 111111110100 |
| 2/5 | 16 | 1111111110001001 |
| 2/6 | 16 | 1111111110001010 |
| 2/7 | 16 | 1111111110001011 |
| 2/8 | 16 | 1111111110001100 |
| 2/9 | 16 | 1111111110001101 |
| 2/A | 16 | 1111111110001110 |
| 3/1 | 6 | 111010 |
| 3/2 | 9 | 111110111 |
| 3/3 | 12 | 111111110101 |
| 3/4 | 16 | 1111111110001111 |
| 3/5 | 16 | 1111111110010000 |
| 3/6 | 16 | 1111111110010001 |
| 3/7 | 16 | 1111111110010010 |
| 3/8 | 16 | 1111111110010011 |
| 3/9 | 16 | 1111111110010100 |
| 3/A | 16 | 1111111110010101 |
| 4/1 | 6 | 111011 |
| 4/2 | 10 | 1111111000 |
| 4/3 | 16 | 1111111110010110 |
| 4/4 | 16 | 1111111110010111 |
| 4/5 | 16 | 1111111110011000 |
| 4/6 | 16 | 1111111110011001 |
| 4/7 | 16 | 1111111110011010 |
| 4/8 | 16 | 1111111110011011 |
| 4/9 | 16 | 1111111110011100 |
| 4/A | 16 | 1111111110011101 |
| 5/1 | 7 | 1111010 |
| 5/2 | 11 | 11111110111 |
| 5/3 | 16 | 1111111110011110 |
| 5/4 | 16 | 1111111110011111 |
| 5/5 | 16 | 1111111110100000 |
| 5/6 | 16 | 1111111110100001 |
| 5/7 | 16 | 1111111110100010 |
| 5/8 | 16 | 1111111110100011 |
| 5/9 | 16 | 1111111110100100 |
| 5/A | 16 | 1111111110100101 |
| 6/1 | 7 | 1111011 |
| 6/2 | 12 | 111111110110 |
| 6/3 | 16 | 1111111110100110 |
| 6/4 | 16 | 1111111110100111 |
| 6/5 | 16 | 1111111110101000 |
| 6/6 | 16 | 1111111110101001 |
| 6/7 | 16 | 1111111110101010 |
| 6/8 | 16 | 1111111110101011 |
| 6/9 | 16 | 1111111110101100 |
| 6/A | 16 | 1111111110101101 |
| 7/1 | 8 | 11111010 |
| 7/2 | 12 | 111111110111 |
| 7/3 | 16 | 1111111110101110 |
| 7/4 | 16 | 1111111110101111 |
| 7/5 | 16 | 1111111110110000 |
| 7/6 | 16 | 1111111110110001 |
| 7/7 | 16 | 1111111110110010 |
| 7/8 | 16 | 1111111110110011 |
| 7/9 | 16 | 1111111110110100 |
| 7/A | 16 | 1111111110110101 |
| 8/1 | 9 | 111111000 |
| 8/2 | 15 | 111111111000000 |
| 8/3 | 16 | 1111111110110110 |
| 8/4 | 16 | 1111111110110111 |
| 8/5 | 16 | 1111111110111000 |
| 8/6 | 16 | 1111111110111001 |
| 8/7 | 16 | 1111111110111010 |

| | | | |
|---|---|---|---|
| 8/8 | | 16 | 1111111110111011 |
| 8/9 | | 16 | 1111111110111100 |
| 8/A | | 16 | 1111111110111101 |
| 9/1 | | 9 | 111111001 |
| 9/2 | | 16 | 1111111110111110 |
| 9/3 | | 16 | 1111111110111111 |
| 9/4 | | 16 | 1111111111000000 |
| 9/5 | | 16 | 1111111111000001 |
| 9/6 | | 16 | 1111111111000010 |
| 9/7 | | 16 | 1111111111000011 |
| 9/8 | | 16 | 1111111111000100 |
| 9/9 | | 16 | 1111111111000101 |
| 9/A | | 16 | 1111111111000110 |
| A/1 | | 9 | 111111010 |
| A/2 | | 16 | 1111111111000111 |
| A/3 | | 16 | 1111111111001000 |
| A/4 | | 16 | 1111111111001001 |
| A/5 | | 16 | 1111111111001010 |
| A/6 | | 16 | 1111111111001011 |
| A/7 | | 16 | 1111111111001100 |
| A/8 | | 16 | 1111111111001101 |
| A/9 | | 16 | 1111111111001110 |
| A/A | | 16 | 1111111111001111 |
| B/1 | | 10 | 1111111001 |
| B/2 | | 16 | 1111111111010000 |
| B/3 | | 16 | 1111111111010001 |
| B/4 | | 16 | 1111111111010010 |
| B/5 | | 16 | 1111111111010011 |
| B/6 | | 16 | 1111111111010100 |
| B/7 | | 16 | 1111111111010101 |
| 8/8 | | 16 | 1111111111010110 |
| B/9 | | 16 | 1111111111010111 |
| B/A | | 16 | 1111111111011000 |
| C/1 | | 10 | 1111111010 |
| C/2 | | 16 | 1111111111011001 |
| C/3 | | 16 | 1111111111011010 |
| C/4 | | 16 | 1111111111011011 |
| C/5 | | 16 | 1111111111011100 |
| C/6 | | 16 | 1111111111011101 |
| C/7 | | 16 | 1111111111011110 |
| C/8 | | 16 | 1111111111011111 |
| C/9 | | 16 | 1111111111100000 |
| C/A | | 16 | 1111111111100001 |
| D/1 | | 11 | 11111111000 |
| D/2 | | 16 | 1111111111100010 |
| D/3 | | 16 | 1111111111100011 |
| D/4 | | 16 | 1111111111100100 |
| D/5 | | 16 | 1111111111100101 |
| D/6 | | 16 | 1111111111100110 |
| D/7 | | 16 | 1111111111100111 |
| D/8 | | 16 | 1111111111101000 |
| D/9 | | 16 | 1111111111101001 |
| D/A | | 16 | 1111111111101010 |
| E/1 | | 16 | 1111111111101011 |
| E/2 | | 16 | 1111111111101100 |
| E/3 | | 16 | 1111111111101101 |
| E/4 | | 16 | 1111111111101110 |
| E/5 | | 16 | 1111111111101111 |
| E/6 | | 16 | 1111111111110000 |
| E/7 | | 16 | 1111111111110001 |
| E/8 | | 16 | 1111111111110010 |
| E/9 | | 16 | 1111111111110011 |
| E/A | | 16 | 1111111111110100 |
| F/0 | (ZRL) | 11 | 11111111001 |
| F/1 | | 16 | 1111111111110101 |
| F/2 | | 16 | 1111111111110110 |
| F/3 | | 16 | 1111111111110111 |
| F/4 | | 16 | 1111111111111000 |
| F/5 | | 16 | 1111111111111001 |

| | | |
|---|---|---|
| F/6 | 16 | 1111111111111010 |
| F/7 | 16 | 1111111111111011 |
| F/8 | 16 | 1111111111111100 |
| F/9 | 16 | 1111111111111101 |
| F/A | 16 | 1111111111111110 |

Table A.2. Huffman Encoding Table for Chrominance AC Terms

| Run/Size | Code length | Code word |
|---|---|---|
| 0/0 (EOB) | 2 | 00 |
| 0/1 | 2 | 01 |
| 0/2 | 3 | 100 |
| 0/3 | 4 | 1010 |
| 0/4 | 5 | 11000 |
| 0/5 | 5 | 11001 |
| 0/6 | 6 | 111000 |
| 0/7 | 7 | 1111000 |
| 0/8 | 9 | 111110100 |
| 0/9 | 10 | 1111110110 |
| 0/A | 12 | 111111110100 |
| 1/1 | 4 | 1011 |
| 1/2 | 6 | 111001 |
| 1/3 | 8 | 11110110 |
| 1/4 | 9 | 111110101 |
| 1/5 | 11 | 11111110110 |
| 1/6 | 12 | 111111110101 |
| 1/7 | 16 | 1111111110001000 |
| 1/8 | 16 | 1111111110001001 |
| 1/9 | 16 | 1111111110001010 |
| 1/A | 16 | 1111111110001011 |
| 2/1 | 5 | 11010 |
| 2/2 | 8 | 11110111 |
| 2/3 | 10 | 1111110111 |
| 2/4 | 12 | 111111110110 |
| 2/5 | 15 | 111111111000010 |
| 2/6 | 16 | 1111111110001100 |
| 2/7 | 16 | 1111111110001101 |
| 2/8 | 16 | 1111111110001110 |
| 2/9 | 16 | 1111111110001111 |
| 2/A | 16 | 1111111110010000 |
| 3/1 | 5 | 11011 |
| 3/2 | 8 | 11111000 |
| 3/3 | 10 | 1111111000 |
| 3/4 | 12 | 111111110111 |
| 3/5 | 16 | 1111111110010001 |
| 3/6 | 16 | 1111111110010010 |
| 3/7 | 16 | 1111111110010011 |
| 3/8 | 16 | 1111111110010100 |
| 3/9 | 16 | 1111111110010101 |
| 3/A | 16 | 1111111110010110 |
| 4/1 | 6 | 111010 |
| 4/2 | 9 | 111110110 |
| 4/3 | 16 | 1111111110010111 |
| 4/4 | 16 | 1111111110011000 |
| 4/5 | 16 | 1111111110011001 |
| 4/6 | 16 | 1111111110011010 |
| 4/7 | 16 | 1111111110011011 |
| 4/8 | 16 | 1111111110011100 |
| 4/9 | 16 | 1111111110011101 |
| 4/A | 16 | 1111111110011110 |
| 5/1 | 6 | 111011 |
| 5/2 | 10 | 1111111001 |
| 5/3 | 16 | 1111111110011111 |
| 5/4 | 16 | 1111111110100000 |
| 5/5 | 16 | 1111111110100001 |
| 5/6 | 16 | 1111111110100010 |
| 5/7 | 16 | 1111111110100011 |

| | | |
|---|---|---|
| 5/8 | 16 | 1111111110100100 |
| 5/9 | 16 | 1111111110100101 |
| 5/A | 16 | 1111111110100110 |
| 6/1 | 7 | 1111001 |
| 6/2 | 11 | 11111110111 |
| 6/3 | 16 | 1111111110100111 |
| 6/4 | 16 | 1111111110101000 |
| 6/5 | 16 | 1111111110101001 |
| 6/6 | 16 | 1111111110101010 |
| 6/7 | 16 | 1111111110101011 |
| 6/8 | 16 | 1111111110101100 |
| 6/9 | 16 | 1111111110101101 |
| 6/A | 16 | 1111111110101110 |
| 7/1 | 7 | 1111010 |
| 7/2 | 11 | 11111111000 |
| 7/3 | 16 | 1111111110101111 |
| 7/4 | 16 | 1111111110110000 |
| 7/5 | 16 | 1111111110110001 |
| 7/6 | 16 | 1111111110110010 |
| 7/7 | 16 | 1111111110110011 |
| 7/8 | 16 | 1111111110110100 |
| 7/9 | 16 | 1111111110110101 |
| 7/A | 16 | 1111111110110110 |
| 8/1 | 8 | 11111001 |
| 8/2 | 16 | 1111111110110111 |
| 8/3 | 16 | 1111111110111000 |
| 8/4 | 16 | 1111111110111001 |
| 8/5 | 16 | 1111111110111010 |
| 8/6 | 16 | 1111111110111011 |
| 8/7 | 16 | 1111111110111100 |
| 8/8 | 16 | 1111111110111101 |
| 8/9 | 16 | 1111111110111110 |
| 8/A | 16 | 1111111110111111 |
| 9/1 | 9 | 111110111 |
| 9/2 | 16 | 1111111111000000 |
| 9/3 | 16 | 1111111111000001 |
| 9/4 | 16 | 1111111111000010 |
| 9/5 | 16 | 1111111111000011 |
| 9/6 | 16 | 1111111111000100 |
| 9/7 | 16 | 1111111111000101 |
| 9/8 | 16 | 1111111111000110 |
| 9/9 | 16 | 1111111111000111 |
| 9/A | 16 | 1111111111001000 |
| A/1 | 9 | 111111000 |
| A/2 | 16 | 1111111111001001 |
| A/3 | 16 | 1111111111001010 |
| A/4 | 16 | 1111111111001011 |
| A/5 | 16 | 1111111111001100 |
| A/6 | 16 | 1111111111001101 |
| A/7 | 16 | 1111111111001110 |
| A/8 | 16 | 1111111111001111 |
| A/9 | 16 | 1111111111010000 |
| A/A | 16 | 1111111111010001 |
| B/1 | 9 | 111111001 |
| B/2 | 16 | 1111111111010010 |
| B/3 | 16 | 1111111111010011 |
| B/4 | 16 | 1111111111010100 |
| B/5 | 16 | 1111111111010101 |
| B/6 | 16 | 1111111111010110 |
| B/7 | 16 | 1111111111010111 |
| B/8 | 16 | 1111111111011000 |
| B/9 | 16 | 1111111111011001 |
| B/A | 16 | 1111111111011010 |
| C/1 | 9 | 111111010 |
| C/2 | 16 | 1111111111011011 |
| C/3 | 16 | 1111111111011100 |
| C/4 | 16 | 1111111111011101 |
| C/5 | 16 | 1111111111011110 |
| C/6 | 16 | 1111111111011111 |

| | | | |
|---|---|---|---|
| C/7 | | 16 | 1111111111100000 |
| C/8 | | 16 | 1111111111100001 |
| C/9 | | 16 | 1111111111100010 |
| C/A | | 16 | 1111111111100011 |
| D/1 | | 11 | 11111111001 |
| D/2 | | 16 | 1111111111100100 |
| D/3 | | 16 | 1111111111100101 |
| D/4 | | 16 | 1111111111100110 |
| D/5 | | 16 | 1111111111100111 |
| D/6 | | 16 | 1111111111101000 |
| D/7 | | 16 | 1111111111101001 |
| D/8 | | 16 | 1111111111101010 |
| D/9 | | 16 | 1111111111101011 |
| D/A | | 16 | 1111111111101100 |
| E/1 | | 14 | 11111111100000 |
| E/2 | | 16 | 1111111111101101 |
| E/3 | | 16 | 1111111111101110 |
| E/4 | | 16 | 1111111111101111 |
| E/5 | | 16 | 1111111111110000 |
| E/6 | | 16 | 1111111111110001 |
| E/7 | | 16 | 1111111111110010 |
| E/8 | | 16 | 1111111111110011 |
| E/9 | | 16 | 1111111111110100 |
| E/A | | 16 | 1111111111110101 |
| F/0 | (ZRL) | 10 | 1111111010 |
| F/1 | | 15 | 111111111000011 |
| F/2 | | 16 | 1111111111110110 |
| F/3 | | 16 | 1111111111110111 |
| F/4 | | 16 | 1111111111111000 |
| F/5 | | 16 | 1111111111111001 |
| F/6 | | 16 | 1111111111111010 |
| F/7 | | 16 | 1111111111111011 |
| F/8 | | 16 | 1111111111111100 |
| F/9 | | 16 | 1111111111111101 |
| F/A | | 16 | 1111111111111110 |

# Appendix B. JPEG C Code

```
/*************************************************************************/
/*                                                                       */
/*  file: cexec_jpeg.c                                                   */
/*  author: Nathan Moroney and Mike Stokes                               */
/*          Date: 1/27/93                                                */
/*                Mike Stokes (mdspci@judd.cis.rit.edu)                  */
/*                Munsell Color Science Laboratory (MCSL)                */
/*                Center for Imaging Science                             */
/*                Rochester Institute of Technology                      */
/*                Rochester, NY  14623                                   */
/*                                                                       */
/*  description:                                                         */
/*    Baseline JPEG Program                                              */
/*    Forward Discrete Cosine Transform.....................................*/
/*    Input image is first broken down into 8 by 8 blocks.               */
/*    These blocks are shifted from unsigned to signed integers first.*/
/*  Calculations are based on the following formula:                     */
/*  F(u,v) = 1/4 C(u)C(v)                                                */
/*             [SUM(x=0-7)SUM(y=0-7) f(x,y)*                             */
/*              cos[(2x+1)uPI]/16*                                       */
/*               cos[(2y+1)vPI]/16 ]                                     */
/*  Inverse Discrete Cosine Transform.....................................*/
/*  Calculations are based on the following equation:                    */
/*  f(x,y) = 1/4[ SUM(u=0-7)SUM(v=0-7) C(u)C(v)*                         */
/*          F(u,v) * cos[(2x+1)uPI]/16 *                                 */
/*                        cos[(2y+1)vPI]/ 16 ]                           */
/*   This program will use the Luminance Quantization table for all      */
/* three channels if the color space is RGB.  Otherwise the program      */
/* will use the luminance quantization table for the first channel       */
/* (L*, Y) and the chrominance quatization table for the other two       */
/* components (a*&b*, u*&v*, x&y, X&Z, i&q).                             */
/*   In addition, this program will only work for images that are        */
/* both horizontally and vertically multiples of 8 (due to the           */
/* sampling of the image into 8x8 blocks.                                */
/*                                                                       */
/*-***********************************************************************/

#include <crs.h>

# define PIE 3.14159265
# define P 8   /* Eight levels of Quatization */
# define HALFSCL 128.0 /* 128 for 0 to 255 images and 0.5 for 0 to 1
images. */

int cexec_jpeg(crs, win, xfcn_jpeg)
crsimg_ptr crs;
win_ptr win;
xfcn_ptr xfcn_jpeg;
{
  register int xb, yb, xp, yp, xt, yt;
  int xtmp, ytmp;
  int tmp2, width, height;
  int index;
  int plane, ll;

  int qtab1_fact[8][8], qtab2_fact[8][8], qtab3_fact[8][8];
```

```
  float dctbasis[64][64], idctbasis[64][64], temp, basis, sum[3], CC;

/* source is the integer array of for the source 8x8 image block */
float source[8][8][3];

/* dctcof is the array of dct coefficients. */
float dctcof[8][8][3];

/* qtcof is the array of quantized dct coefficients. */
int qtcof[8][8][3];
int quantcof[64][3];

/* zzag[64][3] is the rearranged (flattened)
   array of quantized dct coefficients.  This step must be done before
   the bit rate can be computed.  */
int zzag[64][3];
int zagorder[64] = {    1,  2,  9,  17, 10,  3,  4, 11,
                  18, 25, 33, 26, 19, 12,  5,  6,
                  13, 20, 27, 34, 41, 49, 42, 35,
                  28, 21, 14,  7,  8, 15, 22, 29,
                  36, 43, 50, 57, 58, 51, 44, 37,
                  30, 23, 16, 24, 31, 38, 45, 52,
                  59, 60, 53, 46, 39, 32, 40, 47,
                  54, 61, 62, 55, 48, 56, 63, 64  };

int DCLbitlut [12] = {2,3,3,3,3,3,4,5,6,7,8,9};
int DCCbitlut [12] = {2,2,2,3,4,5,6,7,8,9,10,11};
int DCbitlut [12][3];
int ACLbitlut [16][10] = { 2,  2,  3,  4,  5,  7,  8,  10, 16, 16,
                     4,  5,  7,  9,  11, 16, 16, 16, 16, 16,
                     5,  8,  10, 12, 16, 16, 16, 16, 16, 16,
                     6,  9,  12, 16, 16, 16, 16, 16, 16, 16,
                     6,  10, 16, 16, 16, 16, 16, 16, 16, 16,
                     7,  11, 16, 16, 16, 16, 16, 16, 16, 16,
                     7,  12, 16, 16, 16, 16, 16, 16, 16, 16,
                     8,  12, 16, 16, 16, 16, 16, 16, 16, 16,
                     9,  15, 16, 16, 16, 16, 16, 16, 16, 16,
                     9,  16, 16, 16, 16, 16, 16, 16, 16, 16,
                     9,  16, 16, 16, 16, 16, 16, 16, 16, 16,
                     10, 16, 16, 16, 16, 16, 16, 16, 16, 16,
                     10, 16, 16, 16, 16, 16, 16, 16, 16, 16,
                     11, 16, 16, 16, 16, 16, 16, 16, 16, 16,
                     16, 16, 16, 16, 16, 16, 16, 16, 16, 16,
                     16, 16, 16, 16, 16, 16, 16, 16, 16, 16 };
int ACCbitlut [16][10] = { 2,  3,  4,  5,  5,  6,  7,  9,  10, 12,
                     4,  6,  8,  9,  11, 12, 16, 16, 16, 16,
                     5,  8,  10, 12, 15, 16, 16, 16, 16, 16,
                     5,  8,  10, 12, 16, 16, 16, 16, 16, 16,
                     6,  9,  16, 16, 16, 16, 16, 16, 16, 16,
                     6,  10, 16, 16, 16, 16, 16, 16, 16, 16,
                     7,  11, 16, 16, 16, 16, 16, 16, 16, 16,
                     7,  11, 16, 16, 16, 16, 16, 16, 16, 16,
                     8,  16, 16, 16, 16, 16, 16, 16, 16, 16,
                     9,  16, 16, 16, 16, 16, 16, 16, 16, 16,
                     9,  16, 16, 16, 16, 16, 16, 16, 16, 16,
                     9,  16, 16, 16, 16, 16, 16, 16, 16, 16,
                     9,  16, 16, 16, 16, 16, 16, 16, 16, 16,
                     11, 16, 16, 16, 16, 16, 16, 16, 16, 16,
```

```
                    14, 16, 16, 16, 16, 16, 16, 16, 16, 16,
                    15, 16, 16, 16, 16, 16, 16, 16, 16, 16 };

int ACbitlut [16][10][3];
int DCprev[3] = {0, 0, 0};
int numbits[3] = {0, 0, 0};
int numzero[3] = {0, 0, 0};
int DCcur[3], absDCcur[3], category[3], abszzag[3], stop[3], eob[3];
int xc, yc, zc, count, twopow, DCflag, ACflag;
int order, rangemax, zcount, bits, cat;
int totalbits = 0;

/* dqtcof is the array of de-quantized dct coefficients. */
float dqtcof[8][8][3];

/* lqtab_fact is a 2D array that will contain the pre-calculated values
of
   the luminance qtable times the quantization factor (QFACT).
Similarly,
   cqtab_fact is a 2D array that will contain the pre-calculated values
of the
   chrominance qtable times the quantization factor. */

  float lqtable[8][8] = {
    16.0, 11.0, 10.0, 16.0, 24.0, 40.0, 51.0, 61.0,
    12.0, 12.0, 14.0, 19.0, 26.0, 58.0, 60.0, 55.0,
    14.0, 13.0, 16.0, 24.0, 40.0, 57.0, 69.0, 56.0,
    14.0, 17.0, 22.0, 29.0, 51.0, 87.0, 80.0, 62.0,
    18.0, 22.0, 37.0, 56.0, 68.0, 109.0, 103.0, 77.0,
    24.0, 35.0, 55.0, 64.0, 81.0, 104.0, 113.0, 92.0,
    49.0, 64.0, 78.0, 87.0, 103.0, 121.0, 120.0, 101.0,
    72.0, 92.0, 95.0, 98.0, 112.0, 100.0, 103.0, 99.0
    };
/* A sample Quantization table for a monochrome signal
   or the Luminace channel for a device-independent color space.*/

  float cqtable[8][8] = {
    17.0, 18.0, 24.0, 47.0, 99.0, 99.0, 99.0, 99.0,
    18.0, 21.0, 26.0, 66.0, 99.0, 99.0, 99.0, 99.0,
    24.0, 26.0, 56.0, 99.0, 99.0, 99.0, 99.0, 99.0,
    47.0, 66.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
    99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
    99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
    99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0,
    99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0, 99.0
    };

  switch (crs->hdr.comp_type) {
    case CP_SHORT :
      crs_warn("cexec_jpeg : char computations not yet implemented");
      return(NOT_OK);
      break;
    case CP_LONG :
      crs_warn("cexec_jpeg : long computations not yet implemented");
      return(NOT_OK);
      break;
    case CP_FLOAT :
    case CP_DOUBLE :
```

```
      width = (crs->hdr.image_width >> 3) << 3;
      height = (crs->hdr.image_height >> 3) << 3;

      ll = crs->hdr.image_width * crs->hdr.color_planes;

      /* A brief set of pre-calculations of the q-table values
multiplied
         times the quantization factor (QFACT).  This is done to make
the
         algorithm a little bit faster.          */

      for (xp = 0; xp < 8; xp++) {
        for (yp = 0; yp < 8; yp++) {
          if (xfcn_jpeg->val[1] == 1.0) {
           qtab1_fact[xp][yp] = lqtable[xp][yp];
           qtab2_fact[xp][yp] = lqtable[xp][yp];
           qtab3_fact[xp][yp] = lqtable[xp][yp];
           /* Set up quantization tables for RGB.*/
        }
        else if (xfcn_jpeg->val[1] == 2.0) {
           qtab1_fact[xp][yp] = cqtable[xp][yp];
           qtab2_fact[xp][yp] = lqtable[xp][yp];
           qtab3_fact[xp][yp] = cqtable[xp][yp];
           /* Set up quantization tables for XYZ.*/
        }
        else {
           qtab1_fact[xp][yp] = lqtable[xp][yp];
           qtab2_fact[xp][yp] = cqtable[xp][yp];
           qtab3_fact[xp][yp] = cqtable[xp][yp];
           /* Set up Q-tables for other color spaces.*/
        }

          qtab1_fact[xp][yp] *= xfcn_jpeg->val[0];
          qtab2_fact[xp][yp] *= xfcn_jpeg->val[0];
        qtab3_fact[xp][yp] *= xfcn_jpeg->val[0];
          for (xt = 0; xt < 8; xt++) {
            for (yt = 0; yt < 8; yt++) {
            dctbasis[xp*8+xt][yp*8+yt] = cos((((2*xt)+1)*xp*PIE)/16.0)
                                * cos((((2*yt)+1)*yp*PIE)/16.0);
            idctbasis[xp*8+xt][yp*8+yt] = cos((((2*xp)+1)*xt*PIE)/16.0)
                                 * cos((((2*yp)+1)*yt*PIE)/16.0);
          }
          }
        }
      }
          if (xfcn_jpeg->val[1] == 1.0) {
              eob[0] = 11;
              eob[1] = 11;
              eob[2] = 11; }
          if (xfcn_jpeg->val[1] == 2.0) {
              eob[0] = 10;
              eob[1] = 11;
              eob[2] = 10; }
          if (xfcn_jpeg->val[1] == 0.0) {
              eob[0] = 11;
              eob[1] = 10;
              eob[2] = 10; }
```

```
        for (xc = 0; xc < 12; ++xc) {
            if (xfcn_jpeg->val[1] == 1.0) {
                  DCbitlut [xc][0] = DCLbitlut [xc];
                  DCbitlut [xc][1] = DCLbitlut [xc];
                  DCbitlut [xc][2] = DCLbitlut [xc];
                  /* Set up Huffman DC codes for RGB space.*/
            }
            else if (xfcn_jpeg->val[1] == 2.0) {
                  DCbitlut [xc][0] = DCCbitlut [xc];
                  DCbitlut [xc][1] = DCLbitlut [xc];
                  DCbitlut [xc][2] = DCCbitlut [xc];
                  /* Set up Huffman DC codes for XYZ. */
            }
            else {
                  DCbitlut [xc][0] = DCLbitlut [xc];
                  DCbitlut [xc][1] = DCCbitlut [xc];
                  DCbitlut [xc][2] = DCCbitlut [xc];
                  /* Set up Huffman DC codes for L C C spaces.*/
            }
         }

      for (xc = 0; xc < 16;++xc) {
            for (yc = 0; yc < 10;++yc) {
            if (xfcn_jpeg->val[1] == 1.0) {
                  ACbitlut [xc][yc][0] = ACLbitlut [xc][yc];
                  ACbitlut [xc][yc][1] = ACLbitlut [xc][yc];
                  ACbitlut [xc][yc][2] = ACLbitlut [xc][yc];
                  /* Set up Huffman AC codes for RGB space. */
            }
            else if (xfcn_jpeg->val[1] == 2.0) {
                  ACbitlut [xc][yc][0] = ACCbitlut [xc][yc];
                  ACbitlut [xc][yc][1] = ACLbitlut [xc][yc];
                  ACbitlut [xc][yc][2] = ACCbitlut [xc][yc];
                  /* Set up Huffman AC codes for XYZ space. */
            }
            else {
                  ACbitlut [xc][yc][0] = ACLbitlut [xc][yc];
                  ACbitlut [xc][yc][1] = ACCbitlut [xc][yc];
                  ACbitlut [xc][yc][2] = ACCbitlut [xc][yc];
                  /* Set up Huffman AC codes for L C C spaces. */
            }
            }
      }



      /* And here begins the actual compression/decompression...     */
      for (xb = 0; xb < width; xb += 8) {
/* printf("rows %4d\r", xb); fflush(stdout); */
        for (yb = 0; yb < height; yb += 8) {
          for (xp = 0; xp < 8; xp++) {
            for (yp = 0; yp < 8; yp++) {
            xtmp = xb + xp; ytmp = yb + yp;
            index = (yb+yp)*11 + (xb+xp)*crs->hdr.color_planes;
              source[xp][yp][0] = crs->cdata[index+0] - HALFSCL;
              source[xp][yp][1] = crs->cdata[index+1] - HALFSCL;
              source[xp][yp][2] = crs->cdata[index+2] - HALFSCL;
            }
```

```
        }
        /* This section of code computes the FDCT coefficients.    */
      for (xp = 0; xp < 8; xp++) {
        for (yp = 0; yp < 8; yp++) {
            sum[0] = 0.0;
          sum[1] = 0.0;
          sum[2] = 0.0;

            if ((yp == 0) && (xp == 0))
              CC = 0.5;
            else if ((yp == 0) && (xp != 0))
              CC = 0.707107;
            else if ((yp != 0) && (xp == 0))
              CC = 0.707107;
            else
              CC = 1.0;

          for (xt = 0; xt < 8; xt++) {
            for (yt = 0; yt < 8; yt++) {
            basis = dctbasis[xp*8+xt][yp*8+yt];
                sum[0] += (source[xt][yt][0] * basis);
                sum[1] += (source[xt][yt][1] * basis);
                sum[2] += (source[xt][yt][2] * basis);
              }
            }
            dctcof[xp][yp][0] = (0.25*CC*sum[0]);
            dctcof[xp][yp][1] = (0.25*CC*sum[1]);
            dctcof[xp][yp][2] = (0.25*CC*sum[2]);
          }
        }
        /*=========================================================*/
        /*  This portion of code quantizes the coefficients.     */
      for (xp = 0; xp < 8; xp++) {
        for (yp = 0; yp < 8; yp++) {
   qtcof[xp][yp][0] =
(int)((int)(dctcof[xp][yp][0]/qtab1_fact[xp][yp]));
            dqtcof[xp][yp][0] =
((int)(dctcof[xp][yp][0]/qtab1_fact[xp][yp]))
                                * qtab1_fact[xp][yp];
   qtcof[xp][yp][1] =
(int)((int)(dctcof[xp][yp][1]/qtab2_fact[xp][yp]));
            dqtcof[xp][yp][1] =
((int)(dctcof[xp][yp][1]/qtab2_fact[xp][yp]))
                                * qtab2_fact[xp][yp];
   qtcof[xp][yp][2] =
(int)((int)(dctcof[xp][yp][2]/qtab3_fact[xp][yp]));
            dqtcof[xp][yp][2] =
((int)(dctcof[xp][yp][2]/qtab3_fact[xp][yp]))
                                * qtab3_fact[xp][yp];
          }
        }


        /*---------------------------------------------------------*/
      /*  Compute the BIT RATE.                                  */
      /************* D C   E N C O D I N G ***********************/
      DCcur[0] =  qtcof[0][0][0] - DCprev[0];
      DCcur[1] =  qtcof[0][0][1] - DCprev[1];
```

```
DCcur[2] =  qtcof[0][0][2] - DCprev[2];
for (xc = 0; xc<3; ++xc) {
    twopow = 0;
    DCflag = 0;
    while (DCflag == 0){
          rangemax = (int) pow ((double)2.0, (double)twopow);
          absDCcur[xc] = (int) fabs ((double)DCcur[xc]);
          if (rangemax > absDCcur[xc]) DCflag = 1;
          if (twopow > 16) DCflag = 1;
          ++twopow;
          }
    category[xc] = twopow - 1;
    numbits[xc] += DCbitlut[category[xc]][xc] + category[xc];
    DCprev[xc] =  qtcof[0][0][xc];
}
/*********************************************************/
/*        A C   E N C O D I N G                          */
for (zc = 0; zc < 3; ++zc) {
count = 0;
numzero[zc] = 0;
 for (xc = 0; xc < 8; ++xc) {
  for (yc = 0; yc < 8; ++yc) {
    quantcof[count][zc] = qtcof[xc][yc][zc];
    ++count;
    }
  }
}

for (zc = 0; zc < 3; ++zc) {
 for (xc = 1; xc < 64; ++xc) {
    order = zagorder[xc];
    zzag[xc][zc] = quantcof[order][zc];
    if (zzag[xc][zc] == 0) ++ numzero[zc];
    else numzero[zc] = 0;
 }
}

for (zc = 0; zc < 3; ++zc) {
zcount = 0;
stop[zc] = 64 - numzero[zc];
for (yc = 1; yc < stop[zc]; ++yc){
    if ((zzag[yc][zc] == 0)&&(zcount < 16)) ++zcount;
    else {
          twopow = 0;
          ACflag = 0;
          if (zzag[yc][zc] == 0) {
                ACflag = 1;
                category[zc] = 0; }
          while (ACflag == 0) {
                rangemax = (int) pow(2,twopow);
                abszzag[0] = (int) fabs((double)zzag[yc][zc]);
                if (rangemax > abszzag[zc]) ACflag = 1;
                if (twopow > 17) ACflag = 1;
                ++twopow; }
          if (twopow != 0) category[zc] = twopow - 1;
          if (category[zc] == 0) numbits[zc] += 12;
          if (category[zc] != 0) {
             cat = category[zc] - 1;
```

```
                    numbits[zc] += category[zc];
                    numbits[zc] += ACbitlut[zcount][cat][0]; }
                  zcount = 0;
                  }
              }
        }
  totalbits = numbits[0] + numbits[1] + numbits[2] + eob[0] + eob[1] +
eob[2];
        /* ------------------------------------------------------- */


          /* This section of code computes the IDCT coefficients.    */
        for (xp = 0; xp < 8; xp++) {
          for (yp = 0; yp < 8; yp++) {
              sum[0] = sum[1] = sum[2] = 0.0;
            for (xt = 0; xt < 8; xt++) {
              for (yt = 0; yt < 8; yt++) {

                  if ((yt == 0) && (xt == 0))
                    CC = 0.5;
                  else if ((yt == 0) && (xt != 0))
                    CC = 0.707107;
                  else if ((yt != 0) && (xt == 0))
                    CC = 0.707107;
                  else
                    CC = 1.0;

              basis = idctbasis[xp*8+xt][yp*8+yt];
                  sum[0] += CC*dqtcof[xt][yt][0]*basis;
                  sum[1] += CC*dqtcof[xt][yt][1]*basis;
                  sum[2] += CC*dqtcof[xt][yt][2]*basis;
                }
              }
            index = (yb+yp)*11 + (xb+xp)*crs->hdr.color_planes;

            temp = (0.25*sum[0]) + HALFSCL;
            if (temp < 0) temp = 0;
              else if (temp > 255) temp = 255;
            crs->cdata[index+0] = temp;

            temp = (0.25*sum[1]) + HALFSCL;
            if (temp < 0) temp = 0;
              else if (temp > 255) temp = 255;
            crs->cdata[index+1] = temp;

            temp = (0.25*sum[2]) + HALFSCL;
            if (temp < 0) temp = 0;
              else if (temp > 255) temp = 255;
            crs->cdata[index+2] = temp;
            }
          }
        }
      }
      /*printf("\n");*/
/*printf ("%d+%d+%d=%d
bits\n",numbits[0],numbits[1],numbits[2],totalbits);*/
      printf ("%d\n",totalbits);
      fflush(stdout);
```

```
        break;
    }
    return(OK);
}
```

## Appendix C. Inverse Color Space Transformations

The section on color spaces in the Background section of the this thesis provides the forward color space transformations for the color spaces used in this research. This appendix provides the inverse color space transformations for each of the color spaces. The XYZ color space is the only color space in which a direct inverse transformation to the RGB color space was established. Consequently, all of the other inverse color space transforms, except for CIELAB LCh, are to XYZ space. The inverse transformation for the CIELCh space is shown from CIELAB LCh to LAB coordinates. The inverse transform from LAB to XYZ can then be used to get to tristimulus space.

1. XYZ to RGB

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 0.0354 & -0.0335 & -0.0106 \\ -0.0097 & 0.0188 & 0.0004 \\ 0.0004 & -0.0014 & 0.0079 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \text{(H.1)}$$

2. YIQ to XYZ

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.9806 & 0.3113 & 0.6052 \\ 1.0 & 0.0 & 0.0 \\ 1.1811 & -1.2531 & 1.8556 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} \quad \text{(H.2)}$$

3. CIELAB to XYZ

if $(L^*) <= 7.9996$

then { $\text{Temp1} = (7.787 / 903.3) * (L^*) + (16 / 116)$ (H.3)

$Y = (Y_n/903.3) * (L^*)$ } (H.4)

else { $\text{Temp1} = (1 / 116) * ((L^*) + 16)$ (H.5)

$Y = (Y_n/116^3)*((L^*)+16)^3$ (H.6)

$\text{Temp2} = (1 / 500) * (a^*) + \text{Temp1}$ (H.7)

if $\text{Temp2} <= 0.206893$

then $X = (X_n / 7.787) * (\text{Temp2} - (16 / 116))$ (H.8)

else $X = X_n * (\text{Temp2})^3$ (H.9)

$\text{Temp3} = \text{Temp1} - (1/200) * (b^*)$

then $Z = (Z_n / 7.787) * (\text{Temp3} - (16 / 116))$ (H.10)

else $Z = Z_n * (\text{Temp3})^3$ (H.11)

4. CIELAB LCh to CIELAB

$a^* = \cos(h_{ab}) * (C^*)$ (H.12)

$b^* = \sin(h_{ab}) * (C^*)$ (H.13)

5. CIELUV to XYZ

if $(L^*) <= 7.996$

then $Y = (Y_n / 903.3) * (L^*)$ (H.14)

else $Y = (Y_n / 116^3) * ((L^*) + 16)^3$ (H.15)

$Temp1 = 0.076923 / (L^*)$ (H.16)

$u' = (u^*) * Temp1 + u'n$ (H.17)

$v' = (v^*) * Temp1 + v'n$ (H.18)

$Temp2 = 1.5 * u' - 4v' + 3$ (H.19)

$x = 2.25 * u' / Temp2$

(H.20)

$y' = Temp2 / v'$ (H.21)

$Temp3 = Y * y'$ (H.22)

$X = x * Temp3$ (H.23)

$Z = Temp3 - X - Y$ (H.24)

## Appendix D. CRT Calibration

The monitor was characterized using a technique previously developed at the Munsell Color Science Laboratory (Motta 1991). This method is based on a physical model and consists of two basic steps. First, a 3x3 matrix is derived to transform from monitor corrected RGB to XYZ and vice versa. Next, the specific characteristics of the CRT; the gain, offset, and gamma of each channel, are estimated in order to convert from RGB digital counts values to monitor corrected RGB values.

Before these calculations were performed, the monitor was first set to a desired white point. For this research, a white point of D90 was chosen and, consequently, the chromaticities of all of the achromatics were about 0.285 for x and 0.299 for y. This white point was choosen because the red gun of the CRT was unusually weak. As a result it was not possible to set the white point to a warmer correlated color temperature, D65 for example, and still keep the neutrals tracking.

Once these adjustments were made, the tristimulus values for a neutral ramp were measured. The LMT C1200 colorimeter was used to perform the calibration measurements. In addition to measuring a neutral ramp, the tristimulus values for the maximum red, green, and blue channels were measured one at a time. The tristimulus values for each of the three channels independently were then used to create a 3x3 matrix to convert from monitor corrected RGB to XYZ. The rows in this 3x3 are simply the tristimulus values for each of the three channels one at a time. This 3x3 was inverted to produce a 3x3 to convert from XYZ to RGB.

The inverse 3x3 matrix, shown in Appendix C, was then used to convert all of the tristimulus data from the neutral ramp to RGB values. The next step was to perform three non-linear optimizations to estimate the gain, offset and gamma for the three channels. These three values are the three parameters in the physical model which can be written as follows:

$$R = ((kg * dr) + ko)^{\gamma} \quad (D.1)$$

$$G = ((kg * dg) + ko)^{\gamma} \quad (D.2)$$

$$B = ((kg * db) + ko)^{\gamma} \quad (D.3)$$

where dr, dg, and db are the raw red, green, and blue digital counts, R, G, and B are the monitor corrected RGB values, kg is the gain, ko is the offset and $\gamma$ is gamma. The gain, offset and gamma parameters are different for each of the three channels. Eqns. D.1 to D.3 can then be used to convert from raw RGB digital counts to linear RGB values. The resulting linear RGB values can the be entered into Eqn. 2.8 to compute tristimulus values.

The statistical software package JMP was used to optimize the model parameters for each of the three channels. The estimates for the gain, offset, and gamma values are listed in Table D.1.

Table D.1 JMP Estimates for CRT Model Parameters.

| | Red | Green | Blue |
|---|---|---|---|
| Gain | 1.08 | 1.07 | 1.07 |
| Offset | -0.08 | -0.07 | -0.07 |
| Gamma | 1.33 | 1.29 | 1.28 |

Finally, these model values and the 3x3 matrix shown in Eqn. 2.8 were tested using an independent data set. This independent data set consisted of cyan, magenta, and yellow ramps and also 14 randomly generated color patches. Once again the LMT C1200 was used to measure the tristimulus values of these samples. The color difference was then computed between the measured tristimulus values and the tristimulus values predicted by the model. The average $\Delta E$ was 3.3 and the maximum $\Delta E$ was 7.1.

As was mentioned previously, this value is unusually large compared to the accuracy usually attained with this model. Generally, the average $\Delta E$ is around one but some unexpected deviation in the CRT luminance inflated the error. This average error is for the CRT after the deviation and the average is not available for before but it is not likely

that it was larger than 3.3. Nevertheless, this calibration and the additional monitor adjustments resulted in a reasonable calibration of the CRT for the purposes of this research.

## Appendix E. Preliminary Psychophysics Experiment

The preliminary experiment consisted of two parts, one to test the perceptibility of the 24 bit/min$^2$ viewing conditions and the other to select the range of quantization levels to use for the threshold experiment. Nine observers participated in the preliminary experiment and the viewing conditions were identical to those outlined in the Experimental section. Two additional images were included and the range of compressions was more extreme than those listed in Table 3.1.

The first part was forced-choice multiple-comparison experiment. The observer was presented with four grey patches of approximately the same luminance. One of the patches was made of one pixel thick altemating white and black horizontal lines. The other three patches were composed of pixels all of one grey level. The location of the lined patch was randomly determined. The observer was then asked to select the lined patch. The objective of this part of the experiment was to determine how visible the CRT pixels were when viewed at one pixel/min$^2$.

The second portion of the experiment was also a forced-choice multiple-comparison experiment. In this case, a 3 by 3 array of images was presented to the observer. In the center of that array was the original image. The surrounding images were all the original compressed and decompressed to various bit levels. These images were randomly placed around the central image for each of the observers. This experiment was designed to help select which images and quantization levels should selected for the two primary psychophysics experiments.

Altogether, there were six different images and six different color spaces selected for a total of 36 arrays of images. For each image-color space combination, there were eight levels of compression and decompression tested. The exact instructions given to the observers are as follows:

"This experiment requires careful examination of images presented on a monitor. The entire experiment will be conducted with the lights off and the observer seated three feet from the screen. The edge of the table at which you are seated is exactly three feet from the monitor. Please keep your eyes at a distance corresponding to this edge. Under no condition should you move closer to or further from the screen.

Part One

There are four achromatic squares displayed on the monitor. Three of these squares are continuous grey patches. The remaining patch is composed of alternating black and white lines. Please select the square that you believe to be the one made of horizontal lines. Circle the corresponding number on your answer sheet and hit *return* on the keyboard in front of you.

Part Two

A three by three array of images is now on the screen. The original image is in the center. Examine the images surrounding the original and determine if any of these eight images are identical to the center. If an image is identical to the center then circle that image's number on your worksheet. There may be more that identical image. When you are satisfied with your answers, enter *return* on the keyboard to see the next set of images. There are 36 three by three arrays of images to be evaluated. There is no pattern to the number of identical images."

The results of the preliminary experiment indicated that the viewing conditions were reasonable. In addition, the results also implied that there were in fact differences in the color compression spaces. The specifics were uncertain, however, because of the limited number of observers and the considerable noise in the data. These results were used to select the compression levels for the first experiment. The image set was also reduced to the final four shown in Images 3.1 to 3.4.

## Appendix F. Observer Instructions

The following text is the instructions for the observers who participated in experiment one.

"This experiment requires careful examinantion of pairs of images presented on a monitor. The entire experiment will be conducted with the lights off and the observer seated three feet from the screen. The edge of the table at which you are seated is exactly three feet from the monitor. Please keep your eyes at a distance corresponding to this edge. Under no condition should you move closer to or further from the screen.

The experiment will begin with a gray field with a single image in the center. You will then be able to toggle back and forth between two images by entering either a '1' or a '2' on the keyboard in front of you. One of the two images is the original. The other image is a copy of lower image quality. For this experiment, lower image quality is any perceptible deterioration in the resolution or color of the image. Closely examine both images and determine which image is NOT the original. Stop on this image and hit the space bar on the keyboard to enter this image as your selection. It may be difficult to distinguish between some pairs but you must still select one of the two images. After hitting the space bar, there will be a beep and a brief pause before the next image is presented. This will continue until a series of beeps and a black screen indicate that the experiment is over. Thank you for your participation."

The text for the second experiment was identical to that of the first except for the two portions in italics. The first two italicized sentences were replaced with: "The two images will have different image quality." The last portion of italicized text was replaced with: "you believe has the best image quality for that pair."

## Appendix G. Sample Probit Analysis SAS Code

The following lines are the actual SAS code used to perform the probit analysis for the CIELAB compression sequence of the birds image. The first section simply enters the experimental data in SAS format. The second portion of the code performs the probit analysis.

```
options nocenter nodate pagesize=60 linesize=78;

data lab;
  input bits totobs obsfail;
  ratio =  bits / 1179648;
  fail = (totobs / 2) + (totobs - obsfail);
  phat = fail / totobs;
  output;
  cards;
1083486 4 2
1108622 20 13
1138969 20 12
1173982 20 12
1291040 20 15
1585919 20 19
1779689 17 14
2097060 17 14
;

proc probit data = lab C=0.5;
  model fail/totobs = ratio / lackfit inversecl;
  output out=b p=prob std=std xbeta=xbeta;
  title 'bi.lab: Probit Output';
  run;
```

The following pages are the SAS ouput for the above code. A considerable volume of output is produced for such a few lines of code. Of special interest are the Chi-Square measures of the goodness-of-fit for the cummulative normal curve. In all 24 cases, the chi-square tests indicated that the probit analysis was a valid procedure to apply to the data. Lastly, the values of MU and SIGMA are the thresholds and the uncertainty for those thresholds respectively.

Probit Procedure

Data Set =WORK.LAB
Dependent Variable=FAIL
Dependent Variable=TOTOBS
Number of Observations= 8
Number of Events = 106 Number of Trials = 138
Log Likelihood for NORMAL -70.50934445

Goodness-of-Fit Tests

| Statistic | Value | DF | Prob>Chi-Sq |
|---|---|---|---|
| Pearson Chi-Square | 4.8250 | 6 | 0.5665 |
| L.R. Chi-Square | 5.2786 | 6 | 0.5086 |

Response Levels: 2 Number of Covariate Values: 8

NOTE: Since the chi-square is small (p > 0.1000), fiducial limits will be calculated using a t value of 1.96.



Probit Procedure

| Variable | DF | Estimate | Std Err | ChiSquare | Pr>Chi | Label/Value |
|---|---|---|---|---|---|---|
| INTERCPT | 1 | 3.39009309 | 2.27351 | 2.22346 | 0.1359 | Intercept |
| RATIO | 1 | -2.8491667 | 2.159843 | 1.740168 | 0.1871 | |

C=0.5000

Probit Model in Terms of Tolerance Distribution

| MU | SIGMA |
|---|---|
| 1.189854 | 0.35098 |

Estimated Covariance Matrix for Tolerance Parameters

| | MU | SIGMA |
|---|---|---|
| MU | 0.016800 | 0.028559 |
| SIGMA | 0.028559 | 0.070790 |

# Appendix H. Bits/Min$^2$ Calculations

The formula for computing the compression measure bits/min$^2$ was introduced in Eqn. 2.7. This appendix will provides additional details for how bits/min$^2$ was computed for this research. All of the experimental images were 512 pixels high by 768 pixels wide and the subjects viewed the images at a distance of three feet. The 512 pixel height translated to a screen size of about 5.37 inches. This arrangement is shown in Fig. H.1 below.

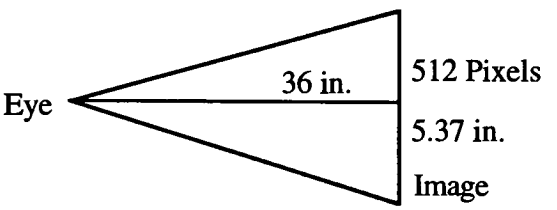



Fig. H.1 Basic Experimental Viewing Geometry.

The viewing geometry in Fig. H.1 can be further modified by bisecting the angle of view. It is assumed that bisecting this angle also approximately bisects the line representing the image plane. The tangent of half of the image size divided by the viewing distance will then yield the angle of view. In this case, the angle of view equals 4.265 degrees or 255.9 minutes. This reduced viewing geometry is presented in Fig. H.2.

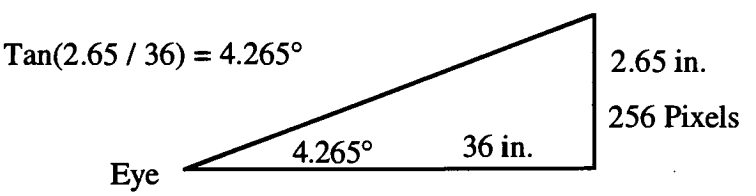



Fig. H.2 Simplified Experimental Viewing Geometry.

Dividing the number of pixels by the number of minutes results in the number of pixels per minute. The preceding values of 256 pixels divided by 255.9 minutes means

that there is about one RGB pixel per minute. Furthermore, assuming that the pixel is approximately square then there is about one RGB pixel per minute squared.

The original image was stored with 8 bits per channel or 24 bits per RGB pixel. Consequently, the image was originally in a 24 bits/min$^2$ form based on the resolution of the PIXAR and the viewing conditions. A simpler way of computing the bits/min$^2$ is simply the bits per channel, called bits per pixel in Eqn 2.6., multiplied by three.